INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

ROBSON CÉRGOLI Nº 1580965

UMA ANÁLISE DA APLICAÇÃO DE TESTES NO DESENVOLVIMENTO DE UM SISTEMA ERP

SÃO PAULO

2017

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

ROBSON CÉRGOLI Nº 1580965

UMA ANÁLISE DA APLICAÇÃO DE TESTES NO DESENVOLVIMENTO DE UM SISTEMA ERP

Dissertação apresentada ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, como exigência para a obtenção do título de Especialista em Gestão da Tecnologia da Informação.

ORIENTADOR: PROF. IVAN FRANCOLIN MARTINEZ

SÃO PAULO

2017

C414a Cérgoli, Robson
Uma análise da aplicação de testes no desenvolvimento de um sistema ERP / Robson Cérgoli. São Paulo: [s.n.], 2017.
93 f.: il.

Orientador: Prof. Ivan Francolin Martinez.

Monografia (Especialização Lato Sensu em Gestão da Tecnologia da Informação) - Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, IFSP, 2017.

1. Testes de Software 2. ERP 3. Qualidade I. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo II. Título

CDD 658.404

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

ROBSON CÉRGOLI Nº 1580965

UMA ANÁLISE DA APLICAÇÃO DE TESTES NO DESENVOLVIMENTO DE UM SISTEMA ERP

ORIENTADOR: PROF. IVAN FRANCOLIN MARTINEZ

BANCA:

Prof. Ivan Francolin Martinez - IFSP

Prof. Me. Francisco Supino Marcondes - IFSP

Prof. Me. Antonio Airton Palladino - IFSP

Data de aprovação: 28 / 06 / 2017.

Dedico este trabalho especialmente à minha mãe que, com muito carinho e apoio, não mediu esforços para que eu chegasse até esta etapa da minha vida.

E também à minha esposa, pessoa que amo partilhar a vida. Obrigado pelo carinho, pela paciência e por sempre acreditar em mim.

AGRADECIMENTOS

Agradeço primeiramente a Deus pela minha vida. Ele que me ilumina, me protege e me guia em todos os meus passos.

Aos meus pais e ao meu irmão, por todos os ensinamentos.

Agradeço à minha esposa, pela paciência, pelo incentivo e por todo o suporte para que pudesse chegar até aqui.

Aos meus parentes e amigos que entenderam algumas das minhas ausências nestes últimos semestres.

Agradeço também à empresa onde trabalho, por incentivar e me apoiar nesta oportunidade.

E finalmente, agradeço ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, aos funcionários e especialmente aos professores do curso de Especialização em Gestão de TI por me propiciarem o conhecimento.

“Não há vitórias sem lutas, muito menos conquistas sem batalhas. A vida é feita de oportunidades e o mais importante talvez não seja saber somente aproveitá-las, mas também criá-las...”

(Leonardo Luis Gomes)

RESUMO

Os sistemas *Enterprise Resource Planning* (ERP) são sistemas de informação integrados, adquiridos na forma de pacotes de software, que visam dar sustentação à maioria das atividades da empresa: suprimentos, manufatura, manutenção, administração financeira, contabilidade, recursos humanos, etc., seja de qualquer segmento em que ela atue: logística, agronegócio, administração, serviços, terceiro setor, entre outros. Porém, por se tratar de um sistema complexo e abrangente, o seu desenvolvimento até se chegar à produção de um software de qualidade, torna-se um desafio. Neste cenário, a atividade de teste torna-se uma das principais tarefas neste processo. O teste de software visa verificar se o sistema está funcionando de acordo com o que foi especificado, além de ser o responsável por identificar erros no processo de desenvolvimento, para que estes não cheguem ao cliente final. Possui muitas etapas, com diferentes tipos de ferramentas, aplicações e metodologias para realizá-lo. Este trabalho teve por objetivo principal demonstrar como um software ERP, no seu processo de desenvolvimento, pode ser testado utilizando-se as mais variadas técnicas existentes no mercado. E, como objetivos secundários, listar os principais fundamentos dos testes de software, seus conceitos, ferramentas, técnicas e metodologias. O estudo de caso foi desenvolvido em uma empresa de software nacional, na qual foram identificados os processos utilizados na constituição do seu ERP focando principalmente nas etapas de testes e identificando as particularidades dela, comparando ao estudo referencial teórico.

Palavras-chave: testes de software, ERP, qualidade.

ABSTRACT

The Enterprise Resource Planning (ERP) systems are integrated information systems, acquired in the form of software packages that have for purpose to support most of the company's activities: supplies, manufacturing, maintenance, financial management, accounting, human resources, etc., be what segment in which it operates: logistics, agribusiness, management, services, third sector, among others. However, because it is a complex and comprehensive system, its development even goes to the production of quality software, it becomes a challenge. In this scenario, a test activity becomes one of the main tasks in the process. The software test aims to verify that the system is working according to what was specified, as well as being responsible for identifying errors in the development process, so that they do not reach the final customer. It has many stages, with different types of tools, applications and methodologies to accomplish it. This work aimed to demonstrate how ERP software, its development process, can be tested using the most varied techniques available in the market. And, as secondary objectives, list the main fundamentals of software testing, its concepts, tools, techniques and methodologies. The case study was developed in a national software company, which identified the processes used in the constitution of its ERP focusing mainly on the stages of testing and identifying as peculiarities of it, comparing to the theoretical reference study.

Keywords: software testing, ERP, quality.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BDD	Behaviour Driven Development (Desenvolvimento Guiado por Comportamento)
BVA	Bondary Value Analysis (Análise do Valor Limite)
CMMI	Capability Maturity Model – Integration (Modelo de Maturidade em Capacitação – Integração)
ERP	Enterprise Resource Planning (Planejamento de Recursos Empresariais)
MRP	Material Requirements Planning (Planejamento das Necessidades de Materiais)
MRPII	Manufacturing Resource Planning (Planejamento de Recursos de Fabricação)
ROI	Return On Investment (Retorno sobre Investimento)
RUP	Rational Unified Process (Processo Unificado da Rational)
SGE	Sistemas de Gerenciamento Empresarial
SQA	Software Quality Assurance (Garantia da Qualidade de Software)
TDD	Test Driven Development (Desenvolvimento Guiado por Testes)
TI	Tecnologia da Informação

SUMÁRIO

1. INTRODUÇÃO

Nos dias atuais a tecnologia evolui rapidamente. Surgem constantemente poderosas ferramentas para criação de software e também de variadas e poderosas plataformas de hardware para comportá-los. De posse destas ferramentas e do hardware adequado, somos capazes de criar sistemas complexos e abrangentes. O sistema *Enterprise Resource Planning* (ERP), responsável por controlar todos os processos realizados numa grande empresa, é um bom exemplo.

De acordo com PRESSMAN (2002), os testes são responsáveis por grande percentual do esforço num projeto de desenvolvimento de software.

Deste modo, no caso dos sistemas ERP, os testes se tornam parte fundamental para o sucesso do projeto. Dada a sua importância, várias técnicas podem ser adotadas, utilizando as mais variadas ferramentas. Equipes geralmente são especializadas neste tipo de atividade, responsáveis por planejar, executar e garantir os testes e a qualidade do produto final.

Segundo OLIVEIRA (2003), “o teste de software é uma atividade crítica na garantia da qualidade. Os seres humanos são passíveis de erros e a atividade de testes é essencial para diminuir os possíveis erros que possam ser encontrados”.

BATISTA (2014), aponta que “qualidade é um conjunto de atributos que se refere ao atendimento das necessidades dos clientes e ao padrão de produtos e serviços disponibilizados por uma empresa”. Estes atributos podem ser relacionados da seguinte forma: moral, qualidade intrínseca, entrega, custo e segurança.

1.1. OBJETIVOS

O objetivo principal deste trabalho é demonstrar a importância dos testes nos processos de desenvolvimento de um sistema ERP, conceitos de testes, técnicas, metodologias, suas vantagens e desvantagens, custos e ganhos.

Estes testes visam minimizar a quantidade de possíveis erros, defeitos ou falhas que poderão ser encontradas pela empresa que adquire este tipo de software.

Como demais objetivos ou objetivos secundários, temos: demonstrar as vantagens e desvantagens dos testes de software quanto à demanda, recursos e

resultados indiretos quanto aos custos, além de ilustrar as diferenças entre as etapas de testes no desenvolvimento de um ERP e de outros tipos de software.

1.2. JUSTIFICATIVA

No projeto de desenvolvimento de software, de um sistema, a atividade de teste pode estar presente em todas as fases. Segundo as fases do *Rational Unified Process* (RUP), por exemplo, os testes estão presentes na Iniciação, Elaboração, Construção e Transição (LUIZ, 2011).

Conforme podemos verificar na Figura 1 – Fases do RUP, o final da fase de Construção é onde a atividade de Teste é mais presente. Justamente nesta fase, as funcionalidades devem ser validadas, a fim de encontrar o maior número de erros, defeitos e falhas possíveis, evitando assim retrabalhos e aumento de custo no projeto.

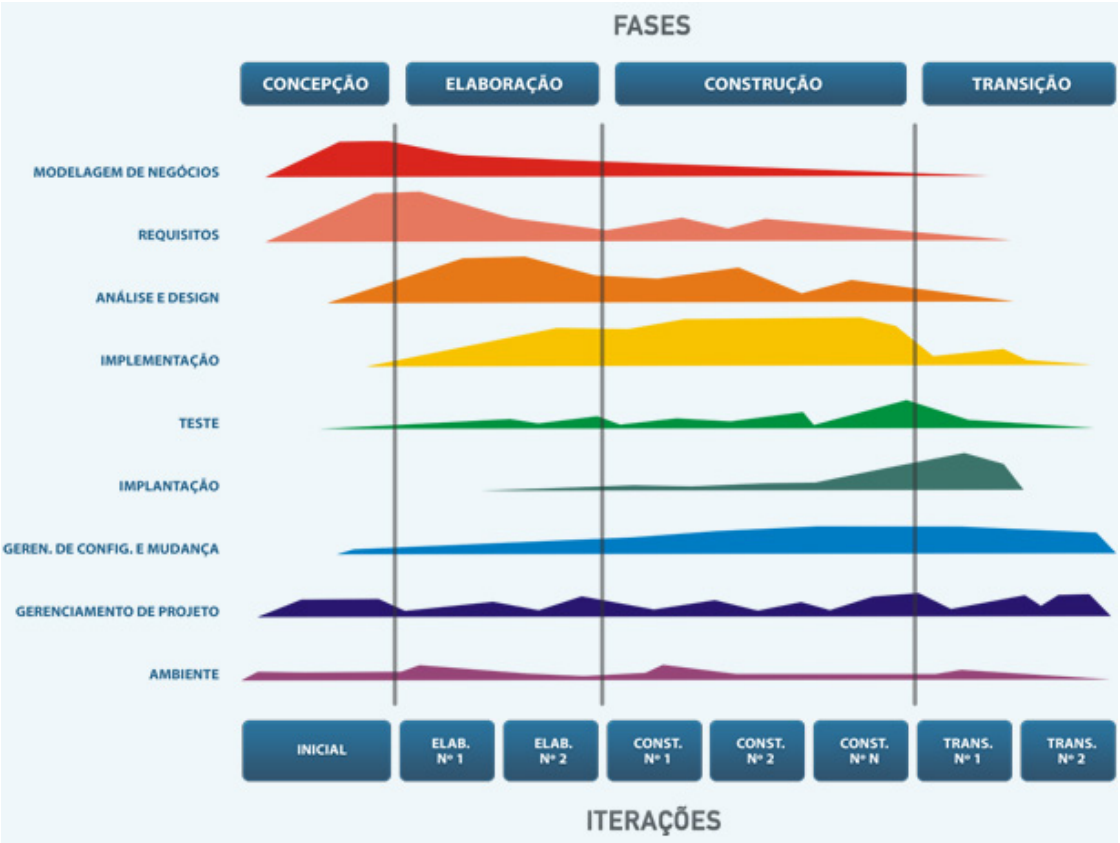

Figura 1 – Fases do RUP
Fonte: LUIZ, 2011.

Já no *Test Driven Development* (TDD), por exemplo, item que também será detalhado no decorrer deste estudo, os testes unitários são definidos antes do próprio desenvolvimento, garantindo uma melhor qualidade a cada etapa do processo de fabricação do software.

1.3. ESTRUTURA DO ESTUDO

A estrutura desta pesquisa começa com a revisão da literatura, descrevendo os principais pontos sobre os processos que envolvem os testes de software, sendo importante destacar: os conceitos de testes, testes positivos e negativos, as diferenças entre defeitos, erros e falhas, as atividades e artefatos de testes, critérios e níveis de testes, as técnicas e o ciclo de vida dos testes, os papéis dos profissionais que trabalham nesta área, além de novas tecnologias, como a Automação de Testes e o surgimento de metodologias ágeis, como por exemplo, o *Behaviour Driven Development* (BDD) e TDD.

Ainda na revisão bibliográfica, os conceitos dos sistemas ERP são ilustrados, trazendo suas características, seus benefícios, vantagens e desvantagens de sua utilização, bem como o desenvolvimento, testes e implantação deste tipo de software nas empresas.

A metodologia utilizada é baseada numa pesquisa qualitativa, através do estudo de caso em uma empresa que desenvolve ERP, focando principalmente nas etapas inerentes aos testes, suas metodologias, características e peculiaridades. Além de aferir, na prática, a real importância e contribuição dos testes para a qualidade do produto final desenvolvido por ela.

Ao final do estudo, são apresentados os resultados obtidos através destas pesquisas, a contribuição acadêmica e a indicação para trabalhos futuros.

2. REVISÃO DA LITERATURA

Segundo SANTOS (2012), a revisão da literatura ou bibliográfica comprova a relevância acadêmica do trabalho realizado pelo pesquisador. Este estudo utilizou o tipo de revisão sistemática.

Esta revisão abordou dois temas principais: os conceitos de teste e os sistemas ERP.

"Teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado" (NETO, 2009a).

O ERP é um sistema de gestão empresarial que visa organizar e gerenciar melhor as informações de uma empresa, funcionando de forma integrada, diminuindo o tempo e os custos dos processos.

2.1. CONCEITO DE TESTES DE SOFTWARE

Segundo NETO (2009a), "atualmente existem várias definições para este conceito, porém ele pode ser compreendido através de uma visão intuitiva ou de maneira formal". Ainda segundo ele, "testar um software significa verificar se o seu comportamento é executado de acordo com o especificado, através de uma execução controlada". O principal objetivo dos testes é buscar o maior número de falhas possíveis, no menor tempo e esforço, demonstrando aos que desenvolvem que os resultados não estão de acordo com o especificado.

> "O primeiro relato sobre *bugs*[1] em computadores aconteceu em 1947, quando engenheiros trabalhavam com a Harvard Mark I[2]. Eles encontraram um inseto nos circuitos. Este inseto estava causando um erro nos cálculos da máquina. O inseto foi retirado e colado no livro de registro". (GONÇALVES, 2013).

O teste de software é uma atividade crítica, porém fundamental na garantia da qualidade do software. Os seres humanos são passíveis a erros, que podem ser encontrados desde a especificação do sistema até a entrega final do

[1] **Bug**: tradução de inseto em inglês.
[2] **Harvard Mark I**: computador totalmente eletromecânico, construído em 1944 pelo professor Howard Aiken, da Universidade de Harvard, em Cambridge (EUA).

produto. Dada a sua importância, os testes podem ser escalonados em paralelo ou vinculados a todas as etapas do projeto.

Neste sentido, os testes de software, que possuem grande representatividade em relação ao tempo total gasto num projeto, tornam-se fundamentais para a garantia da entrega daquilo que foi acordado e especificado e, também, da qualidade do produto que está sendo entregue.

MYERS (2004) lista um conjunto de regras que podem ser utilizadas para elencar os objetivos de testar um software:

> - “Testar é o processo de executar um programa com intenção de encontrar um erro;
> - Um bom caso de teste é aquele que possui uma alta probabilidade de encontrar um erro não descoberto;
> - Um teste de sucesso é aquele que encontra erros não identificados anteriormente”. MYERS (2004).

2.1.1. TESTES POSITIVOS E NEGATIVOS

O teste positivo é aquele em que se verifica se e quanto o sistema atende aos requisitos funcionais e não funcionais especificados. Já o teste negativo é a execução do programa com o objetivo de encontrar erros/defeitos.

De acordo com FARIAS (2014), podemos traduzir os testes positivos e negativos da seguinte forma:

- Positivo: é o “caminho feliz” do software, ou seja, as ações previstas. É a confirmação que o teste faz aquilo que deve fazer;
- Negativo: é o caminho “infeliz” do software, ou seja, as ações imprevistas. O software não faz aquilo que não deve fazer.

2.1.2. DIFERENÇAS ENTRE DEFEITOS, ERROS E FALHAS

Em linhas gerais, defeitos são imperfeições que estão de alguma maneira, ligados a estrutura do produto. Erros são problemas que impedem a continuação da operação ou que são diferentes do resultado esperado. E falhas são ocorrências “observáveis” dos defeitos. (NETO, 2009a).

Uma das metas das atividades de testes é identificar e conduzir à correção de defeitos antes que ocorram falhas, se possível o quanto antes durante o ciclo de vida do projeto.

Por definição, defeito é a diferença em relação a um atributo desejado do produto, seja em relação às especificações ou em relação às expectativas do usuário ou cliente. Podemos dividir os defeitos em dois tipos:

- Defeitos de lógica de programação: erros em algoritmos, normalmente de baixo impacto no projeto, mas que podem consumir tempo considerável para sua correção;

- Defeitos de sintaxe: erros de programação, detectados na compilação do código fonte. As ferramentas atuais de desenvolvimento facilitam muito a remoção deste tipo de erro, a ponto de tornar o esforço necessário para isso mínimo, porém devem ser levados em consideração, principalmente quando não há disponíveis tais ferramentas ou para ambientes de grande porte (NETO, 2009a).

Por outro lado, qualquer incorreção ou resultado diferente do esperado na execução do programa são considerados erros. É a manifestação concreta do defeito ou a diferença entre o valor obtido e aquele que realmente seria (NETO, 2009a).

Já as falhas são as conseqüências dos defeitos. Ou seja, é a manifestação do defeito, impactando diretamente o usuário. Uma falha pode ter sido causada por diversos erros, porém alguns erros podem nunca causar uma falha (NETO, 2009a).

2.1.3. ATIVIDADES

No planejamento ou no ciclo de construção de desenvolvimento, devem ser definidas também quais etapas serão destinadas aos testes. Para isso, uma série de atividades podem ser utilizadas:

- Casos de teste;
- Planejamento de teste;
- Procedimento de teste;
- Critérios de teste;
- Artefatos;

- Ferramentas de testes.

2.1.3.1. CASOS DE TESTE

De acordo com NETO (2009b), o caso de teste descreve uma condição particular a ser testada e é composto por valores de entrada, restrições para a sua execução e um resultado, comportamento esperado ou requisitos especificados do sistema.

Eles são os primeiros passos das atividades de teste em um projeto de software. São eles que definem as entradas a serem informadas pelo testador, manualmente ou com apoio de ferramentas, além dos resultados esperados a partir desta ação.

“Os casos de teste estabelecem quais informações serão empregadas durante os testes dos cenários e quais serão os resultados esperados, estabelecendo a massa crítica de teste necessária para validar todos os requisitos do software” (BASTOS et al., 2007).

Para elaborar os casos de testes, deve-se saber:

- O que se pretende testar?
- Quando irá testar?
- Como irá testar?
- Onde irá testar?

2.1.3.2. PLANEJAMENTO DE TESTE

Segundo TRAVASSOS et al., (2006), alguns itens são fundamentais para o planejamento de testes. Dentre eles, podemos citar:

- Preocupação com a definição dos objetivos e escopo dos testes;
- Identificação dos itens que serão testados;
- Identificação das características dos itens que serão avaliadas;
- Escolha das abordagens de teste a serem utilizadas;
- Definição dos recursos humanos e físicos, e custos para os testes;
- Identificação dos casos e procedimentos de teste, e sua priorização;

- Identificação de um cronograma para as atividades de teste e seus responsáveis.

2.1.3.3. PROCEDIMENTO DE TESTE

Um procedimento de teste define os passos ou sequência necessários para executar os casos. Estes visam apoiar na adequação do esforço de teste em um projeto. “É uma descrição dos passos necessários para executar um caso ou um grupo de casos de teste” (NETO, 2009b).

A descrição pode conter:

- A identificação dos procedimentos de testes;
- Os objetivos, como por exemplo, a cobertura dos testes;
- Os requisitos especiais para a execução do procedimento;
- O fluxo do procedimento, detalhando o passo a passo para que possa ser executado manualmente por um testador ou convertido em um script de teste.

2.1.4. CRITÉRIOS DE TESTE

“Os critérios de testes servem para selecionar e avaliar casos de teste de forma a aumentar as possibilidades de provocar falhas ou, quando isso não ocorre, estabelecer um nível elevado de confiança na correção do produto” (ROCHA et al., 2001). Os critérios de teste podem ser utilizados como: critério de cobertura de testes, critério de adequação de casos de teste e critério de geração de casos de teste.

- Critério de cobertura de testes: “permitem a identificação de partes do programa que devem ser executadas para garantir a qualidade do software e indicar quando o mesmo foi suficientemente testado” (RAPPS e WEYUKER, 1982). Ou seja, “determinam o percentual de elementos necessários por um critério de teste que foram executados pelo conjunto de casos de teste” (NETO, 2009b);

- Critério de adequação de casos de teste: quando, a partir de um conjunto de casos de teste qualquer, ele é utilizado para verificar se satisfaz os requisitos de teste estabelecidos pelo critério. Ou seja, “este critério avalia se os

casos de teste definidos são suficientes ou não para avaliação de um produto ou uma função” (ROCHA et al., 2001);

- Critério de geração de casos de teste: “quando o critério é utilizado para gerar um conjunto de casos de teste T adequado para um produto ou função, ou seja, este critério define as regras e diretrizes para geração dos casos de teste de um produto que esteja de acordo com o critério de adequação definido anteriormente” (ROCHA et al., 2001).

2.1.5. ARTEFATOS

Os artefatos descrevem os documentos que deverão ser desenvolvidos ao longo das atividades de teste. A Figura 2 – Relacionamento entre os documentos de testes - demonstra os documentos produzidos ao longo do processo de teste e outros documentos. (SILVA, 2011).

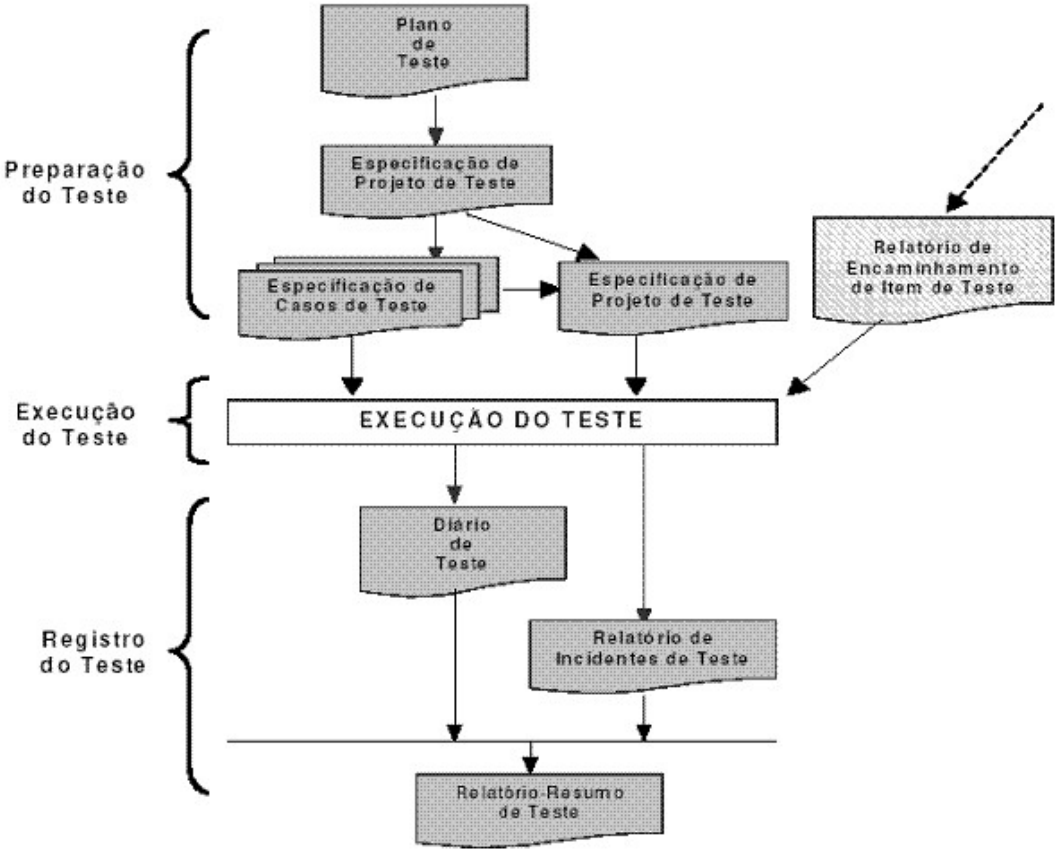

Figura 2 – Relacionamento entre os documentos de testes

Fonte: SILVA, 2011

O Plano de Testes é o primeiro documento a ser criado e estabelece o planejamento para a execução dos testes, incluindo objetivo, escopo, abordagem de teste adotada, recursos físicos e humanos, além do cronograma das atividades de teste. Também contempla a identificação dos itens e funcionalidades a serem testadas, as características a serem avaliadas em cada um dos itens, as tarefas a serem realizadas e os riscos associados às atividades de teste. (SILVA, 2011).

Além do Plano de testes, a fase de preparação contempla a atividade de especificação de testes que inclui a elaboração de 3 (três) documentos, conforme (SILVA, 2011) descreve a seguir:

- Especificação de Projeto de Teste: inclui o detalhamento da abordagem definida no Plano de Testes e identifica as funcionalidades que serão avaliadas com base nas características definidas no Plano. Este documento também identifica os casos e os procedimentos de teste, quando aplicável. Além disso, a especificação de projeto de teste estabelece os critérios para aprovação das funcionalidades avaliadas durante este projeto de teste;

- Especificação de Caso de Teste: trata da definição dos dados de entrada, resultados esperados, ações e condições gerais para a execução do teste;

- Especificação de Procedimento de Teste: consiste na definição dos passos para executar um ou mais casos de teste.

2.1.6. CICLO DE VIDA DOS TESTES

Segundo publicação da Funpar (2016), “o ciclo de vida dos testes faz parte do ciclo de vida do software. Eles devem ser iniciados em um período de tempo equivalente”. O processo de design e execução dos testes pode ser tão complexo quanto o processo de codificação.

O ciclo de vida dos testes pode ser dividido em 5 (cinco) fases: Planejamento, Preparação, Especificação, Execução e Entrega.

No Planejamento são elaboradas: a Estratégia de Teste, o Plano de Testes, e a análise de risco do projeto de testes. “Essa 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” (FARIAS, 2014).

Na etapa de Preparação, o ambiente de testes para a execução é preparado. “Entende-se por ambiente de testes os equipamentos, hardware e software, ferramentas para automação de testes, “massa” de dados, entre outros. Ou seja, toda a estrutura necessária para execução dos testes deverá estar disponível nessa etapa” (FARIAS, 2014).

A Especificação é a etapa onde são elaborados e revisados 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 liberação por parte da equipe de desenvolvimento dos módulos a serem testados” (FARIAS, 2014).

Na etapa de Execução é que são efetivamente executados os testes, conforme planejamento nas etapas anteriores e, registrados os resultados obtidos durante a execução. (FARIAS, 2014).

“A Entrega é responsável por 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” (FARIAS, 2014).

2.1.7. NÍVEIS DE TESTE DE SOFTWARE

“O planejamento dos testes deve ocorrer em diferentes níveis e em paralelo ao desenvolvimento do software” (NETO, 2009b). Os principais níveis de teste de software são: Teste de Unidade, Teste de Integração, Teste de Sistema, Teste de Aceitação e Teste de Regressão.

- Teste de unidade: “também conhecido como teste unitário. Tem por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente. O universo alvo desse tipo de teste são os métodos dos objetos ou mesmo pequenos trechos de código” (NETO, 2009a);

- Teste de integração: o teste de integração tem por objetivo garantir que as interfaces entre os módulos funcionem satisfatoriamente, assegurando que os dados estão sendo processados de forma correta. Tendo em vista que a execução do teste de integração requer o conhecimento sobre as estruturas internas e interações existentes entre as diversas partes do sistema, este teste tende a ser realizado pela equipe de desenvolvimento. Além disso, a realização dos testes de

integração deve ser feita, preferencialmente, de forma incremental, iniciando com a integração de algumas unidades e resultando no sistema integrado. Entretanto, em muitas empresas, o teste de integração só é realizado após a junção de todas as unidades do sistema. Essa prática não é aconselhável, pois dessa forma é muito mais difícil detectar os defeitos e corrigi-los (RAKITIN, 2001);

- Teste de sistema: “avalia o software em busca de falhas por meio da utilização do mesmo, como se fosse um usuário final. Dessa maneira, os testes são executados nos mesmos ambientes, com as mesmas condições e com os mesmos dados de entrada que um usuário utilizaria no seu dia-a-dia de manipulação do software. Verifica se o produto satisfaz seus requisitos” (NETO, 2009a);

- Teste de aceitação: “são realizados geralmente por um restrito grupo de usuários finais do sistema. Esses simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado” (NETO, 2009a);

- Teste de regressão: “não corresponde a um nível de teste, mas é uma estratégia importante para redução de “efeitos colaterais”. Consiste em se aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do sistema. Pode ser aplicado em qualquer nível de teste” (NETO, 2009a).

A Figura 3 - Modelo V – Paralelo entre desenvolvimento e testes – demonstra este modelo muito utilizado para descrever o paralelismo entre as atividades de desenvolvimento e teste de software.

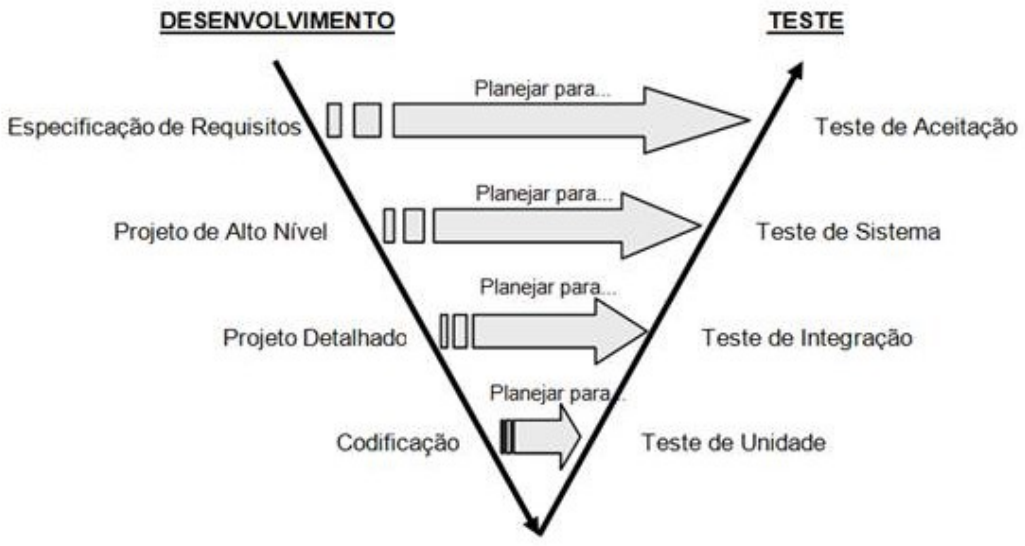

Figura 3 - Modelo V – Paralelo entre desenvolvimento e testes

Fonte: NETO, 2009a.

2.1.8. DEFEITOS NO DESENVOLVIMENTO DE SOFTWARE

Para evitar os defeitos, diminuir os riscos torna-se fundamental. Afinal, quanto menor o risco, menor a probabilidade de encontrar erros. Essa afirmação deve ser aplicada tanto para o projeto de desenvolvimento de software, como para o de Teste.

> "O processo de desenvolvimento de software cria produtos com defeitos. Na maioria das vezes, os defeitos são considerados riscos para a imagem da empresa e ao negócio. Por isso, apenas descobri-los não é suficiente. É fundamental adotar o gerenciamento de defeitos no processo de teste para que os riscos sejam minimizados, controlados, evidenciados e o seu impacto não seja grande". LAGARES, ELIZA (2011).

Conforme LAGARES, ELIZA (2011), "seria perfeito se os defeitos não existissem, e os *bugs* jamais impedissem o bom funcionamento de um software. No entanto, enquanto não chegamos a essa situação ideal, gerenciar os defeitos produzidos torna-se essencial".

Mudanças no processo de desenvolvimento de software ocorrem a todo o momento, por inúmeras razões, como restrições de tempo e custo, novas possibilidades de negócios e alteração nas necessidades de clientes. Em função disso, saber identificar a importância dos defeitos é fundamental para entender o impacto que eles causarão no sistema e nos negócios da empresa.

Por isso, é importante que a gestão de defeitos seja realizada, pois a mesma possibilita uma visão geral e consequentemente um melhor acompanhamento do andamento do projeto, através da verificação dos *bugs* registrados.

Neste contexto, a qualidade do sistema pode ser medida a partir dos *bugs* encontrados durante todo o seu ciclo de vida, desde a fase de projeto, até ser colocado efetivamente em produção. E para que os *bugs* sejam gerenciados com sucesso, é necessário que a gestão de defeitos seja utilizada de maneira simples, tornando-se de fundamental importância dentro de um processo de Teste de Software. (LAGARES, ELIZA, 2011).

2.1.9. TÉCNICAS DE TESTES DE SOFTWARE

As técnicas de teste podem ser: técnica estrutural, também conhecida como teste caixa branca, teste funcional, também chamada de teste caixa preta e uma terceira, que seria a mescla das duas primeiras, que é chamada de teste caixa cinza.

> “Atualmente existem muitas maneiras de se testar um software. Mesmo assim, existem as técnicas que sempre foram muito utilizadas em sistemas desenvolvidos sobre técnicas estruturadas que ainda hoje tem grande valia para os sistemas orientados a objeto.” (PERES, 2009).

Apesar destas técnicas possuírem paradigmas de desenvolvimento completamente diferentes, o objetivo principal delas continua sendo o mesmo, ou seja, encontrar falhas no software.

2.1.9.1. TÉCNICA ESTRUTURAL OU TESTE CAIXA BRANCA

Os testes de caixa branca são aqueles que garantem avaliar a estrutura de um software, já que o analista possui acesso ao código fonte. Nesse caso, as condicionantes e situações específicas podem ser detalhadas em cada caso de teste.

Os passos básicos para se aplicar um Teste Caixa Branca são:

- Analisa-se o código fonte da aplicação;
- Escolhe-se caminhos através da implementação;
- Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados;
- As saídas esperadas para as entradas escolhidas são determinadas;
- Os casos de testes são construídos;
- As saídas obtidas são comparadas com as saídas esperadas;
- Um relatório é gerado para avaliar o resultado dos testes.

A técnica de Teste de Caixa Branca trabalha diretamente sobre o código fonte do software para avaliar aspectos tais como:

- Teste de Fluxo de Controle;
- Teste de Condição;

- Teste de Fluxo de Dados;
- Teste de Ciclos.

No Fluxo de Controle utilizam-se apenas características de controle de execução do programa, como comandos ou desvios, para determinar quais estruturas são necessárias. Os critérios mais conhecidos são: todos os nós, todos os arcos ou todos os caminhos. Um nó representa uma ou mais instruções as quais são sempre executadas em sequência. Um arco representa o fluxo de controle entre blocos de comandos (nós).

No Teste de Condição, o teste exercita as condições booleanas de um módulo de programa, focalizando em cada condição para garantir que o programa não contém erros. Pode ser uma condição simples ou condição composta (E, OU e NÃO).

O Fluxo de dados utiliza informações do programa para determinar os requisitos de teste. Esses critérios exploram as interações que envolvem definições de variáveis e referências a tais definições para estabelecerem os requisitos de teste. Os critérios mais conhecidos são os de Rapps e Weyuker e os critérios Potenciais-Usos. (SANTOS, 2011).

O Teste de Ciclos focaliza a validade das construções dos ciclos. Os ciclos podem ser: simples, concatenados, aninhados ou desestruturados.

2.1.9.2. TESTE FUNCIONAL OU TESTE CAIXA PRETA

Os testes são derivados da especificação do programa. Este tipo de teste é chamado de teste de caixa preta, pois o componente é uma caixa preta cujo comportamento só pode ser determinado pela análise das entradas e saída relacionadas. É uma abordagem complementar para o teste de caixa branca. Este tipo de teste tende a ser aplicado nas últimas etapas da atividade de testes (SOMMERVILLE, 2011).

Algumas das técnicas de teste Caixa Preta são: Particionamento de Equivalência, Análise de Valor Limite e Teste de Comparação.

De acordo com OLIVEIRA (2003), o particionamento de equivalência é uma técnica que determina quais classes de dados de entrada tem propriedades comuns. O programa deve se comportar da mesma maneira para todos os

elementos de propriedades semelhantes. Existem partições de equivalência para entradas e saídas. A partição de equivalência pode ser identificada pela especificação do programa ou documentação do usuário e pela experiência do testador em prever quais classes de entradas de dados são possíveis de detectar erros. Como exemplo, se a especificação de uma entrada diz que algum valor de entrada deve ser um inteiro de 5 dígitos, isto é, entre 10.000 e 99.999, a partição de equivalência pode ser valores menores que 10.000, valores entre 10.000 e 99.999, e valores maiores que 99.999.

A análise de valor limite ou *Bondary Value Analysis* (BVA), segundo OLIVEIRA (2003), leva em consideração a escolha de casos de teste que põem à prova os valores fronteiriços, ou seja, aqueles próximos ao valor desejado, onde costumam ocorrer o maior número de erros. A análise do valor limite completa a técnica de particionamento de equivalência. A BVA também deriva casos de testes do domínio de saída. Devem ser exercitados os valores máximos e mínimos, os valores logo abaixo deles e logo acima respectivamente, relatórios de saída que produzam o número máximo de entradas permitidas numa tabela, casos de teste que exercitem o número máximo de entradas de um *array*, entre outros. Por exemplo, se o valor desejado deve estar entre 5 e 10, testa-se estes mais os valores logo abaixo (4 – teste negativo e 9 – teste positivo) e os logo acima (6 – teste positivo e 11 – teste negativo).

Em casos críticos, há a elaboração de software paralelamente. Neste caso, os programas são testados com a mesma base de teste e em paralelo. Esses testes são chamados de Teste de Comparação.

2.1.9.3. TESTE CAIXA CINZA

A técnica de caixa-cinza é um ponto de equilíbrio entre os testes de caixa-preta e de caixa-branca.

Segundo JUNIOR (2017), em tese, o testador não tem acesso ao código fonte, porém tem conhecimento dos algoritmos que foram implementados. Além disso, pode executar manipulações em arquivos de entrada e saída, acessar o banco de dados para efetuar simples conferências ou até mesmo alterar parâmetros descritos nos casos de teste.

Nestes casos, as saídas são avaliadas através de processos externos, tendo como referencial os processos internos que foram usados para gerá-los.

2.1.10. PAPÉIS

De acordo com ALMEIDA (2010), os principais participantes no processo de testes são:

- Gerente de Teste: tem como papel defender a qualidade dos testes, planejar e gerenciar os recursos e resolver os problemas que representam obstáculos ao esforço de teste;

- Líder de Teste: pessoa responsável pela liderança de um projeto de teste específico, normalmente relacionado a um projeto de desenvolvimento, sejam um projeto novo ou uma manutenção;

- Analista de Teste: elabora e modela os casos e roteiros de testes. Deve focar seu trabalho nas técnicas de teste adequadas à fase de teste trabalhada;

- Arquiteto de Teste: É responsável por montar a infra-estrutura de testes como: ambiente, ferramentas, capacitação da equipe, entre outros;

- Testador: executa os testes, o mesmo deve observar as condições de teste e respectivos passos de teste documentados pelo analista de teste e evidenciar os resultados de execução;

- Automatizador: tem como papel automatizar as situações de teste em ferramentas observando as condições de teste documentadas pelo analista de teste e automatizar a execução desses testes na ferramenta utilizada. Normalmente são gerados scripts de teste que permitam a execução de ciclos de teste sempre que se julgar necessário.

Uma pessoa pode assumir mais de um dos papéis citados acima, como por exemplo, um testador pode exercer o papel de um automatizador de testes também.

2.1.11. AUTOMAÇÃO DE TESTES

Nos dias atuais, as empresas estão cada vez mais dependentes da tecnologia, tanto para sustentar suas operações internas quanto para efetivar

negociações. Assim, qualquer risco de mau funcionamento de sistemas ou aplicações podem gerar um impacto negativo na organização. Por isso, testar os softwares, sistemas e aplicações são procedimentos fundamentais para minimizar os riscos e manter a eficiência na gestão e na operação organizacionais, garantindo a qualidade do produto e sua conformidade com a necessidade do cliente.

Contudo, testar softwares e sistemas é sempre um desafio e um custo muito grande. Em uma aplicação grande e robusta, o esforço necessário para os testes de integração e regressão a cada novo *release* é enorme. Assim, a Automação de Testes aparece como uma solução para aumentar o número de testes em menor tempo e custo.

Segundo publicação do site da STEFANINI (2013), a Automação de Testes, além de permitir que a equipe de qualidade foque nas novas funcionalidades de uma aplicação, deixando o esforço de “reteste” a cargo dos scripts de automatização, proporciona redução de tempo e de custos em longo prazo, permite a procura de classes específicas de defeitos e, no caso de aplicações que possuem um longo ciclo de vida, permite também que incrementos ou pequenas correções no código não afetem funcionalidades que já estavam estabelecidas.

A Automação de Testes pode ser realizada em diversos tipos de testes:

- Teste de Regressão: avalia se uma nova funcionalidade impacta em funcionalidades preexistentes;

- Teste de Performance: avalia o tempo de resposta de uma aplicação.

- Teste Funcional: este tipo de teste avalia o comportamento da aplicação do ponto de vista do usuário, para assegurar que ela entregue o que se espera dela;

- Teste de Segurança: permite avaliar as vulnerabilidades em aplicações e serviços frente a diferentes tipos de ataques de segurança.

Em todos esses casos, a Automação de Testes pode avaliar vários atributos das aplicações, entre eles: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.

2.1.11.1. FERRAMENTAS DE AUTOMAÇÃO

Para se chegar à automação dos vários tipos de testes já descritos, seja funcional, de regressão ou desempenho, algumas etapas devem ser realizadas

primeiro, utilizando-se de ferramentas e técnicas de mercado para se chegar no objetivo que é o teste da rotina automático. Como exemplo, podemos citar o seguinte fluxo de automação de testes:

- Criação do Mapa Mental: segundo MARINS (2009), “o mapa mental é uma ferramenta que permite a memorização, organização e representação da informação com o propósito de facilitar os processos de aprendizagem, administração e planejamento organizacional, assim como a tomada de decisão”. Conforme podemos verificar na Figura 4 – Exemplo de Mapa Mental - trata-se de uma ferramenta para ilustrar ideias e conceitos, dar-lhes forma e contexto, traçar os relacionamentos de causa, efeito, simetria e/ou similaridade que existem entre elas e torná-las mais palpáveis e mensuráveis, sobre os quais seja possível planejar ações e estratégias para alcançar objetivos específicos.

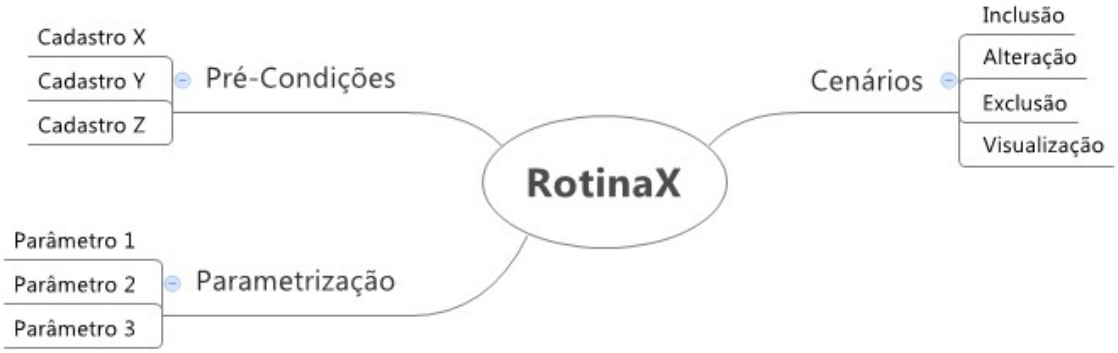

Figura 3 – Exemplo de Mapa Mental
Fonte: elaborado pelo autor, 2016.

- Criação dos casos de testes: para cada cenário do mapa mental, é criado um caso de teste. O caso de teste vêm a ser um algoritmo, um passo-a-passo simplificado daquilo que será descrito no fonte de automação. O mesmo caso de teste pode ser utilizado em um ou mais cenários. E também em uma ou mais rotinas. Tudo vai depender das suas interligações. Na Figura 5 – Exemplo de Casos de Teste - há dois exemplos de casos de testes, um que valida o *login* no sistema e outro, um cadastro de clientes:

- Script de teste: os scripts de teste automatizados podem ser criados (registrados) ou gerados automaticamente com ferramentas de automatização de teste. Eles utilizam alguma linguagem de programação ou uma combinação de elementos. Os scripts de teste podem ser modificados para incluir conceitos de

programação, como a referência a bibliotecas de funções comuns, para utilizar variáveis e *loops*, e para criar ramificações a fim de aumentar a eficiência, a eficácia e a resistência dos scripts.

ID	CT-001
Caso de Teste:	Efetuar login no sistema.
Funcionalidade:	Login
Pré - Condição:	1. Possuir usuário válido para efetuar o login.
Procedimento:	1. Acessar o sistema; 2. Inserir usuário e senha; 3. Clicar em "Login".
Resultado Esperado:	1. Login do usuário efetuado com sucesso.

ID	CT-002
Caso de Teste:	Cadastrar cliente no Sistema
Funcionalidade:	Cadastro de cliente
Pré - Condição:	Estar logado no sistema
Procedimento:	1. Entrar na tela "Cadastro" -> "Cliente" 2. Inserir dados do cliente. 3. Clicar em "Salvar".
Resultado Esperado:	1. Cliente é cadastrado com sucesso..

Figura 4 - Exemplo de Casos de Teste
Fonte: elaborado pelo autor, 2016.

Na Figura 6 – Exemplo de Script de Teste Automatizado - é demonstrado um exemplo simples de teste de *login*, desenvolvido em WebDriver por NOGUEIRA (2016).

```
11 public class TesteLogin {
12     @Test
13     public void testeLoginGrupo() {
14         WebDriver driver = new FirefoxDriver();
15         driver.get("http://exemplo.com/login");
16 
17         driver.findElement(By.id("usuario")).sendKeys("elias");
18         driver.findElement(By.id("senha")).sendKeys("123456");
19         driver.findElement(By.id("login")).click();
20 
21         assertEquals("Admin", driver.findElement(By.id("grupos")).getText());
22         driver.quit();
23     }
24 }
```

Figura 5 - Exemplo de Script de Teste Automatizado
Fonte: NOGUEIRA, 2016.

- Robô de teste: o "robô" de teste é a interface responsável pela comunicação entre o script de teste automatizado e a rotina ou sistema a ser testado. Em alguns casos, esta comunicação não é necessária, já que é utilizada

uma simulação do sistema e de seus dados. O robô executa o teste de acordo com o script e exibe o seu resultado ao final, sendo ele positivo ou negativo.

2.1.11.2. DIFERENÇAS ENTRE TESTE MANUAL E TESTE AUTOMATIZADO

De acordo com FILHO (2015), “o teste manual é uma atividade da ciência da sabedoria, uma atividade que requer julgamento humano. Quando um analista está testando, ele está usando conhecimento implícito para julgar se algo está ou não funcionando como esperado”. Através da sua expertize, experiências anteriores ou até mesmo intuição, o dá a chance de encontrar *bugs* extras, os quais testes automatizados nunca iriam encontrar. Manualmente, também permite que sejam seguidos “sinais”, não explícitos, para explorar áreas que podem não ter sido testadas ou exigidas.

Teste manual também é útil para encontrar problemas de *layout* e *bugs* triviais, os quais não seriam encontrados por testes automatizados, já que é realizada uma verificação do contexto geral, ou seja, observada a aplicação total enquanto a utiliza. Outro cenário diz respeito aos problemas de usabilidade, que também são identificáveis por testes manuais, não podendo ser descobertos através da escrita e execução de scripts automatizados de teste.

Em contrapartida, os testes automatizados bem feitos, não são facilmente enganados por um mero *pop-up* inesperado. Testes bem arquitetados e bem escritos evitam esse tipo de situação e trabalham em ambiente controlado.

Hoje em dia é possível fazer muito mais do que se pensa com testes automatizados, até mesmo testes de comparações de *print screens*, os quais reduzem bastante o tempo de testes manuais para verificar simples *bugs* de estilo e layouts quebrados.

Recomenda-se que os testes automatizados sejam escritos antes do código da funcionalidade em si. Desta maneira, eles guiam o desenvolvimento da funcionalidade em busca da simplicidade e arquitetura evolutiva.

Porém, mesmo com todas estas facilidades dos testes automatizados, obviamente algumas coisas ainda assim serão somente encontradas com testes manuais. Contudo, no mercado atual, profissionais que fazem só isso estão cada vez mais deixando de ser procurados e, por outro lado, cada vez mais busca-se não

só pessoal qualificado para trabalhar com automação de testes, mas profissionais completos, os quais são desenvolvedores que criam testes, arquitetam uma solução que agrega valor, desenvolvem e colocam software funcionando em produção continuamente. (FILHO, 2015).

2.1.11.3. VANTAGENS E DESVANTAGENS DOS TESTES AUTOMATIZADOS

Os testes automatizados são uma realidade nos dias atuais. Principalmente nas grandes empresas, as buscas por este tipo de teste vêm crescendo consideravelmente. Dentre as vantagens da automatização de testes, FILHO (2015) cita:

- São muito explícitos, “preto no branco”, então há uma chance maior de reproduzir um *bug*;

- Devido aos testes automatizados serem explícitos, eles também executam consistentemente, ou seja, eles não se cansam ou ficam com preguiça como os humanos, os quais são mais propensos ao erro;

- São mais rápidos de serem executados do que testes manuais, já que não há atraso entre o tempo de entrada e a verificação. Isso significa que você pode executar mais testes em menos tempo;

- Testes automatizados também o capacitam a testar coisas que são manualmente impossíveis. Por exemplo, responder uma pergunta como “e se eu tiver 2000 contas?”, ou “e se seu processasse 100 transações simultaneamente?” pode somente ser respondida eficientemente ao usar testes automatizados.

FILHO (2015), porém, também diz que, ao mesmo tempo em que os testes automatizados facilitam o trabalho, minimizam os erros e aumentam a produtividade, alguns “problemas” devem ser levados em conta, como por exemplo:

- Mesmo ao automatizar um cenário de teste, você tem que testá-lo manualmente ao menos uma vez de alguma forma para automatizá-lo. E você tem que, manualmente, verificar o resultado do teste;

- Testes automatizados podem parar de funcionar por algo como um *pop-up* inesperado, o qual pode ser rapidamente analisado e rejeitado quando testando manualmente.

2.1.12. IMPACTOS NEGATIVOS DOS ITENS COM ERROS (NÃO-TESTADOS)

Segundo JUNIOR (2010), é senso comum, além de comprovado na prática que, a identificação e resolução antecipada de problemas e a correção de defeitos, gera economia de recursos, em particular financeiros, nos projetos de software. Como verificado na Figura 7 – Custo da correção de defeitos - quanto mais cedo o problema é identificado, menor será o custo.

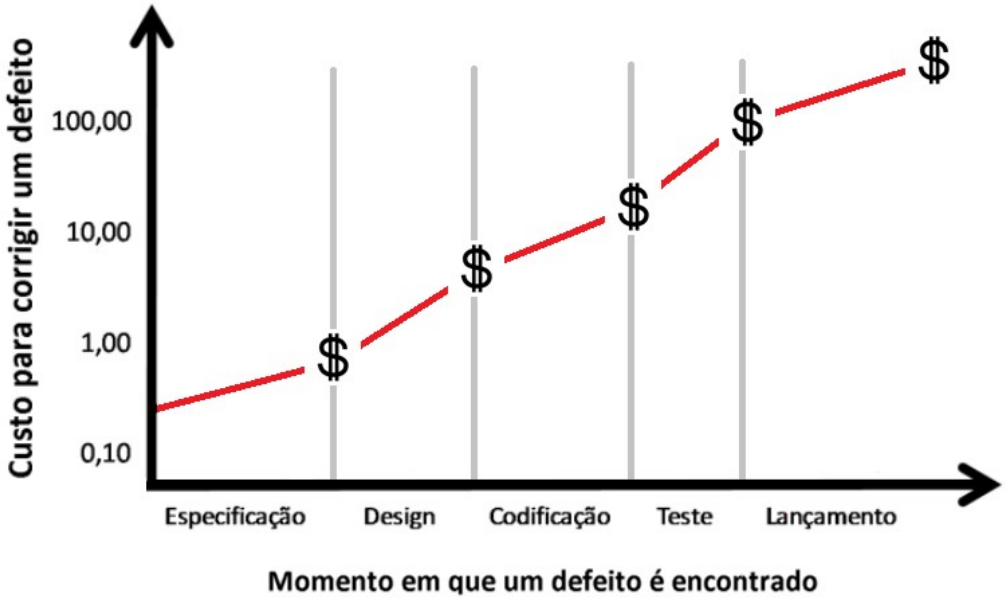

Figura 7 - Custo da correção de defeitos
Fonte: adaptado de VIEGAS, 2015

“Requisitos bem definidos colaboram para aumentar a capacidade de testá-los sob diversas perspectivas: unidade, integração, sistema e aceite”. (JUNIOR, 2010).

Além disso, outro ponto relevante a ser observado é que, na fase de manutenção do software, além do custo, a imagem da empresa fornecedora é afetada, dependendo da quantidade ou do tipo de erro encontrado pelo cliente.

Entretanto, sabe-se que entregar um software livre de erros é, na prática, uma tarefa quase impossível de ser alcançada. Os testes, sejam eles manuais ou automatizados, tem eficiência limitada e a remoção dos defeitos envolve custos.

Para cada defeito existente, deve-se avaliar se o benefício obtido com sua remoção justifica o esforço necessário para tal. Deve-se, portanto, avaliar qual o

nível, em cada um de seus fatores determinados, de qualidade atingível e, acordá-lo com o cliente.

A Figura 8 – Relação entre qualidade de software e extensão do teste - demonstra o que chamamos de teste “ótimo”. HIRAMA (2011) cita que “o teste ótimo é o ponto ideal para a maioria dos casos, porém é essencial que se trabalhe com prioridades e criticidades para se definir a cobertura dos testes”. Cada tipo de negócio requer um esforço específico. Por exemplo, um sistema de aviação para Controle de Vôo, deve ser “supertestado”. Já um sistema de portaria, que controla a catraca, é um exemplo de sistema que pode ser “subtestado”. Para os ERPs, o teste geralmente pára no ponto ótimo. Possuir base histórica, indicadores, etc. ajudam a definir a cobertura de testes com mais assertividade.

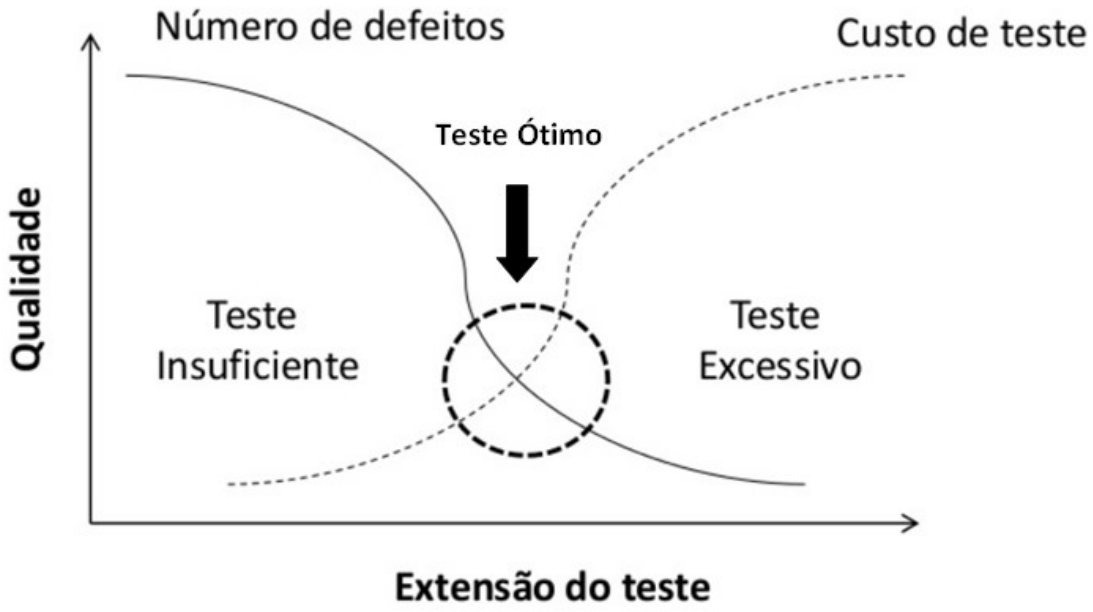

Figura 8 - Relação entre qualidade de software e extensão do teste
Fonte: HIRAMA, 2011.

2.1.13. METODOLOGIAS ÁGEIS (TDD E BDD)

Algumas metodologias, consideradas novas no mercado, estão ganhando força e começando a ser implantadas em empresas de desenvolvimento de software. O TDD e BDD são metodologias de desenvolvimento ágil, representadas através dos fluxos na Figura 9 – Diferença entre TDD e BDD.

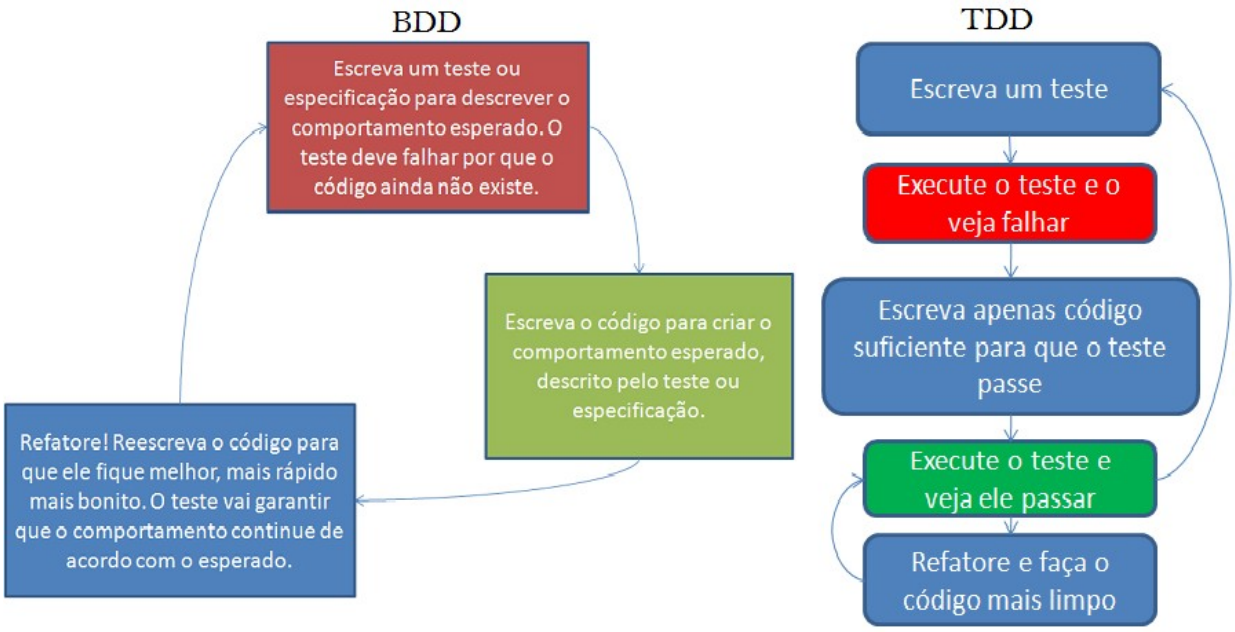

Figura 9 - Diferença entre TDD E BDD
Fonte: CALDAS, 2014.

Segundo CALDAS (2014), “no TDD, o desenvolvimento é guiado a testes, onde um teste unitário deve ser escrito antes que uma funcionalidade do sistema o seja. O objetivo, então, é fazer com que o teste passe com sucesso, significando que assim a funcionalidade está pronta e conta com garantia de qualidade”.

Já no BDD, CALDAS (2014) diz que “o desenvolvimento deve ser guiado aos comportamentos que o sistema deve apresentar. Desta forma, um comportamento (requisito/especificação) é priorizado em relação ao teste unitário, o que não exclui a execução do fluxo do TDD neste processo”.

Existem algumas ferramentas que oferecem apoio à adoção de uma metodologia ou outra ao processo de codificação de sistemas, das quais RSpec e Cucumber são exemplos. Em resumo elas permitem:

- escrever uma especificação em uma “pseudo-linguagem”;

- transformar a especificação em uma função de teste.

Isso porque esses *frameworks* implementam o TDD para guiar a realização do BDD.

> “Assim como TDD e BDD, existem outras metodologias de desenvolvimento ágil que pode ser aplicadas à etapa de codificação do software (DDD, FDD, ATDD, etc). Cada uma delas evidencia uma etapa do ciclo de vida do processo de desenvolvimento do software ou um artefato que faça parte

> dele como forma de guiar a construção do sistema. Para saber qual deles utilizar é necessário realizar uma análise sobre o projeto, equipe e outros recursos envolvidos no processo". (CALDAS, 2014).

2.2. SISTEMAS ERP

PRESSMAN (2002) afirma que "os ERPs ou Sistemas de Gerenciamento Empresarial (SGEs) são um exemplo de como as ferramentas de software estão se tornando cada vez mais abrangentes e complexas". Além disso, os ERPs são extremamente genéricos, ou seja, são concebidos para atender às necessidades de empresas das mais variadas áreas de negócios.

Segundo HABERKORN (2003), "uma empresa adquire um sistema ERP visando à automação dos seus processos". O ERP abrange: planejamento, execução e controle, sob o ponto de vista financeiro e econômico, objetivando a execução das tarefas de forma mais rápida e eficiente, fornecendo mobilidade para toda e empresa, independente da sua área de atuação no mercado.

Este tipo de software possui características diferenciadas, desde o seu planejamento até a sua implantação e manutenção. Geralmente uma empresa, seja ela de pequeno, médio ou grande porte, possui um sistema ERP. Por isso esse mercado é abrangente e ao mesmo tempo especializado.

2.2.1. O QUE É UM SISTEMA ERP

De acordo com OLIVEIRA (2003), "o ERP é um conceito administrativo que evoluiu do *Material Requirements Planning* (MRP) e *Manufacturing Resource Planning* (MRPII). Atualmente a maioria dos sistemas de gestão empresarial são baseados neste conceito e, portanto, são chamados de sistemas ERP".

Antigamente, em meio à necessidade de reduzir os níveis de estoque, as empresas começaram a adotar os primeiros sistemas de MRP. Estes sistemas ofereciam uma visão integrada dos bens, matéria prima e estoque, baseada no inventário realizado e nos períodos de reabastecimento. Sendo assim, o MRP

realizava o planejamento de todos os materiais necessários para o atendimento de um pedido de cliente.

Ainda de acordo com OLIVEIRA (2003), o cálculo do MRP está diretamente ligado à estrutura de produtos, saldos de materiais em estoque, ordens em andamento e políticas de planejamento, objetivando definir as quantidades e momentos em que cada item deve ser produzido ou comprado. Para o atendimento destas necessidades, são geradas ordens de compra e fabricação considerando as datas das necessidades. O resultado deste cálculo é um plano de compra e um plano de fabricação para que o pedido do cliente seja atendido no prazo desejado.

O aperfeiçoamento do MRP e do MRPII é o sistema ERP que, não só permite gerir a manufatura, como também controla toda a empresa, da produção às finanças, passando pelo gerenciamento dos recursos humanos, de forma integrada e sincronizada. OLIVEIRA (2003) afirma que, o ERP automatiza os processos de uma empresa, como pode ser verificado na Figura 10 – Estrutura típica do funcionamento de um sistema ERP. Conseqüentemente, elimina interfaces complexas e caras entre sistemas não projetados para conversarem, ou seja, não integrados, gerando uma base de dados única, sem redundâncias encontradas nos antigos sistemas.

Figura 10 - Estrutura típica de funcionamento de um sistema ERP
Fonte: OLIVEIRA, 2003.

2.2.2. CARACTERÍSTICAS

Os sistemas ERP passaram nos últimos anos por um processo de rápida evolução. Suas características, do mesmo modo, também foram sendo redefinidas. Atualmente, os fornecedores de ERPs continuam expandindo seus sistemas com novos recursos e soluções, a fim de sempre estarem presentes na chamada “corrida tecnológica”.

Para serem considerados com tal, ou seja, indispensáveis para as empresas, os sistemas ERP possuem características próprias, sendo estas apresentadas abaixo conforme publicação do site da TERAWARE[3] (2014):

- Flexibilidade: o sistema ERP é flexível de forma a responder às constantes transformações das empresas, podendo operar sob diferentes bases de dados, já que as informações podem mudar de área durante os processos;

- Modularidade: módulos são os menores conjuntos de funções que podem ser adquiridos e implementados, separadamente em um sistema ERP. Ele se utiliza do chamado sistema de arquitetura aberta, isto é, pode ser utilizado um módulo livremente, sem que este afete os restantes. Normalmente cada módulo representa um setor da empresa. Facilita também a expansão e/ou adaptabilidade de mais módulos posteriormente. Isso possibilita que uma empresa implemente apenas aquelas partes do sistema que sejam de seu interesse, e, mesmo que esta deseje implementar todo o sistema, possa fazê-lo por etapas, através de um processo escalonado. Além disso, a divisão conceitual de um sistema ERP em módulos facilita a compreensão de seu funcionamento e a divisão de responsabilidades entre os usuários, ou seja, quem faz o quê;

- Multiplataforma: geralmente suporta múltiplas plataformas de software e hardware, pois muitas empresas possuem sistemas heterogêneos;

- Compreensivo: se adapta a um vasto repertório de áreas de negócio e também suporta diferentes estruturas organizacionais das empresas;

- Conectividade: não precisa estar confinado ao espaço físico da empresa, permitindo a ligação com outras entidades ou filiais pertencentes ao mesmo grupo empresarial;

---

[3] **TERAWARE:** portal de soluções em software e internet para gestão de pequenas e médias empresas.

- Melhores Práticas: em geral, possui em suas bases a seleção das melhores práticas de negócio do mercado;

- Simulação da Realidade: permite a simulação da realidade da empresa com prospecções para os diversos módulos existentes, gerando sólidos relatórios e gráficos para auxiliar na tomada de decisões;

A TERAWARE (2014) também ilustra, além das características apresentadas, outros conceitos importantes relativos aos sistemas ERP: funcionalidade, parametrização, configuração, customização, localização e atualização. Abaixo, seguem as definições de cada um desses conceitos:

- Funcionalidade: é o conjunto total de funções embutidas em sistema ERP, suas características e suas diferentes possibilidades de uso;

- Parametrização: é o processo de adequação da funcionalidade de um sistema ERP a uma determinada empresa através da definição dos valores de parâmetros já disponibilizados nele próprio;

- Configuração: é o nome dado ao conjunto total de parâmetros após sua definição, representando o conjunto das opções de funcionamento das diversas funções de um sistema ERP;

- Customização: é a modificação de partes de um sistema ERP para que este possa se adequar a uma determinada situação empresarial impossível de ser reproduzida através dos parâmetros já existentes;

- Localização: é a adaptação, através de parametrizações ou adequações, de sistemas ERP desenvolvidos em determinado país para utilização em outro, considerando aspectos como impostos, taxas, leis, procedimentos comerciais, práticas de negócio, etc.;

- Atualização: são as novas versões disponibilizadas pelo fornecedor a fim de aumentar a funcionalidade, corrigir erros/problemas existentes, atender às mudanças nas legislações, etc.

2.2.3. BENEFÍCIOS DE UTILIZAÇÃO DE UM SISTEMA ERP

Dentre os muitos benefícios na utilização de um sistema ERP, OLIVEIRA (2003) cita principalmente a agilidade nos processos, muito pelo fato de que as informações são colocadas no sistema somente uma única vez, eliminando a

repetição de tarefas. Exemplificando essa integração entre as áreas, vamos dar o exemplo da entrada de um pedido no sistema: no instante em que um pedido é gerado, imediatamente atualiza-se o planejamento de vendas, aciona-se a fábrica, solicita-se a matéria-prima necessária e automaticamente já se torna possível dar ao cliente uma estimativa precisa do prazo de entrega. O fechamento contábil do mês, fase em que antigamente os relatórios ficavam pelo menos 20 dias sendo passados de um departamento a outro, é realizado de forma muito mais rápida e acertiva.

Outra grande vantagem é que as operações manuais são minimizadas. Consequentemente existe uma redução de horas extras e também o ajuste do quadro de funcionários a esta realidade. Exceto quando uma empresa utiliza diferentes sistemas, coexistindo através de uma interface ou não, as operações manuais são na maioria das vezes necessárias, pois existe a necessidade de se realizar uma constante conferência, conciliação ou compatibilização de informações entre sistemas e ajustes ou correções de informações nestes sistemas.

O fato de todos os módulos do ERP acessarem uma única base de dados evita a redundância. Desta forma, as informações são consistentes e a tomada de decisão na empresa também é beneficiada. Conseqüentemente, obtém-se uma melhor satisfação dos clientes internos e externos.

“Além disso, existem outras vantagens geradas pelo próprio conceito administrativo ERP, como redução de estoque, maior controle do processo produtivo, entre outros”. (OLIVEIRA, 2003).

2.2.4. DESVANTAGENS DE UM SISTEMA ERP

OLIVEIRA (2003) descreve que o problema da maioria dos softwares dessa categoria é o fato de serem flexíveis demais, isso mais pelo lado da fornecedora, que é quem implanta. Em contrapartida, com relação aos clientes, espera-se de um negócio mais disciplina do que a maioria tem. Os ERPs geralmente são feitos para acomodar todo tipo de processo e de negócio, da refinação de petróleo à revenda de automóveis usados. Isso acaba gerando dois problemas. O primeiro é encontrar, nesse emaranhado, a configuração que melhor se adapta a cada empresa. Em seguida, uma vez instalado o programa, é preciso que os dados

sejam fornecidos corretamente por quem os usa, senão os erros também se propagam em tempo real.

> “Outro fator que, dependendo dos processos de trabalho, pode ser considerado como uma desvantagem é que o ERP exige da empresa que se adapte ao sistema, ou seja, os sistemas ERP levam as empresas a modificar seus processos para se adequar aos descritos em seus módulos. O problema é maior ainda quando as empresas não têm seus processos mapeados. Entretanto, empresas que possuem bons processos de negócios não irão ser beneficiadas com adaptações aos modelos do sistema. Já aquelas que possuem processos ultrapassados, com mau funcionamento, terão um grande benefício com tal adaptação. Este fato pode ser amenizado com as customizações. A customização é uma forma de manutenção adaptativa, que geralmente visa incrementar funcionalidades aos pacotes ou mesmo simplificar determinada funcionalidade. Entretanto, customizações não são fortemente recomendadas, uma vez que torna significativamente difícil a manutenção do sistema e a atualização de versões”. (OLIVEIRA, 2013).

Outro problema que as empresas enfrentam, sendo na maioria das vezes as multinacionais, é a adaptação às leis e práticas de negócios de outros países. Como se trata de um programa de escopo geral e padrão, especificidades locais são seu ponto fraco. Em alguns casos, são adquiridos outros sistemas que complementam a funcionalidade não existente no ERP. Um exemplo: cada país tem suas próprias leis e tributações, com isso, se um sistema foi concebido para funcionar na Alemanha, quando implantado no Brasil, diversas configurações, modificações e adaptações devem ser realizadas para atender essas particularidades, principalmente no que diz respeito a impostos.

> “A própria integração gerada pelo sistema também pode ser vista como uma vantagem ou como uma desvantagem, pois apesar da integração ser benéfica, ela faz com que qualquer dado colocado no sistema por uma área específica tenha imediato impacto em todas as demais áreas em que esse dado será utilizado de alguma forma”. (OLIVEIRA, 2003).

2.2.5. IMPLANTAÇÃO DE UM SISTEMA ERP

Segundo OLIVEIRA (2003), a implantação de um sistema ERP é um projeto bastante complexo, já que afeta a organização como um todo, inclusive nas

operações do dia-a-dia. Para muitas empresas, talvez esse seja o maior projeto em que já se envolveram. Os projetos são na maior parte dos casos de longa duração. Entretanto, o tempo não é exato, já que depende do tamanho da empresa, da área de negócio em que atuam e do sistema contratado. Porém, em média, os projetos de implantação costumam levar de 6 (seis) meses a um ano. A implantação pode ser conduzida através de várias metodologias elaboradas pelas diversas consultorias atuantes neste campo. Entretanto, as metodologias existentes são significativamente similares sendo divididas em fases que proporcionam os mesmos resultados.

As fases mais comuns na implantação do sistema ERP são: planejamento do projeto, treinamento da equipe, levantamento de processos, configuração e parametrização, testes, treinamento de usuários, migração de dados, até finalmente o sistema entrar em produção, o chamado de *Go-live*. Após isso, ainda torna-se necessário o período de adaptação, fase em que a fornecedora acompanha os primeiros processos da empresa e realiza os últimos ajustes, até os usuários conseguirem “caminhar” sozinhos.

2.2.6. DESENVOLVIMENTO DE SISTEMA ERP

O desenvolvimento de um sistema ERP ocorre em várias etapas. Para tal, torna-se fundamental a adoção de uma metodologia para basear-se e mapear os processos.

Um exemplo de metodologia para desenvolvimento é o RUP, que se utiliza de quatro fases principais, definidas por (SILVA, 2006) como:

- Início: entendimento da necessidade e visão do projeto;
- Elaboração: especificação e abordagem dos pontos de maior risco;
- Construção: desenvolvimento principal do sistema;
- Transição: ajustes, implantação e transferência de propriedade do sistema.

Porém o RUP na íntegra é muito mais complexo, abordando os papéis em um maior nível de detalhes. Para cada fase, há uma série de disciplinas que são seguidas. Dentre elas, SILVA (2006) cita como as principais:

- Análise: desde o momento em que foi decidido o desenvolvimento do software, baseado em critérios de levantados junto ao cliente;

- Especificação de Objetivos: após a análise, são estabelecidos os objetivos do sistema a ser desenvolvido;

- Planejamento: feito todo o levantamento, é necessário fazer a modelagem do sistema;

- Desenvolvimento e Programação: codificação. Nesta etapa são usadas as ferramentas para o desenvolvimento do projeto;

- Validação e Revisão: antes de ser considerado efetivamente concluído, o sistema proposto ou o produto, deve ser submetido a testes;

- Implantação: produção e entrega do software para os usuários finais;

- Gerenciamento de Configuração e Mudança: responsável pela estruturação sistemática e solicitações de mudança;

- Gerenciamento de Projeto: concentra-se principalmente sobre aspectos importantes de um desenvolvimento iterativo, como Gestão de Riscos, por exemplo;

- Ambiente: enfoca as atividades necessárias para configurar o processo para um projeto, descrevendo as atividades para desenvolver as diretrizes de apoio.

2.2.7. CUSTOMIZAÇÃO

Por se tratar de um sistema complexo, o ERP geralmente deve ser flexível no que diz respeito às customizações, ou seja, personalização de acordo com os processos da empresa. Às vezes, o mesmo processo difere de uma empresa para outra.

> “Basicamente, um sistema ERP customizável é uma solução tecnológica pré-formatada que permite adequações diversas para fazer com que os recursos e funcionalidades disponíveis estejam de acordo com as necessidades e processos operacionais do seu negócio”. (LEITE, 2014).

A contratação, o desenvolvimento, os testes e todas as outras etapas da customização são apartadas do projeto original. A empresa fornecedora geralmente possui uma equipe de “Fábrica de software” para oferecer este tipo de serviço. Já a outra ponta, ou seja, o cliente contrata a empresa de acordo com a sua necessidade ou, em alguns casos, possui em sua própria equipe de TI, pessoas especializadas para tal.

2.2.8. FORNECEDORES DE SISTEMAS ERP

De acordo com PORTALERP [4] (2014), com dados de 2013, no mundo, as principais fornecedoras de sistemas ERP são: SAP, Oracle e Microsoft. No Brasil, a líder é a TOTVS, com quase 37% do mercado, seguida da SAP e da Oracle.

Na Figura 11 – Mercado de ERP no mundo - é possível identificar essas empresas no cenário mundial.

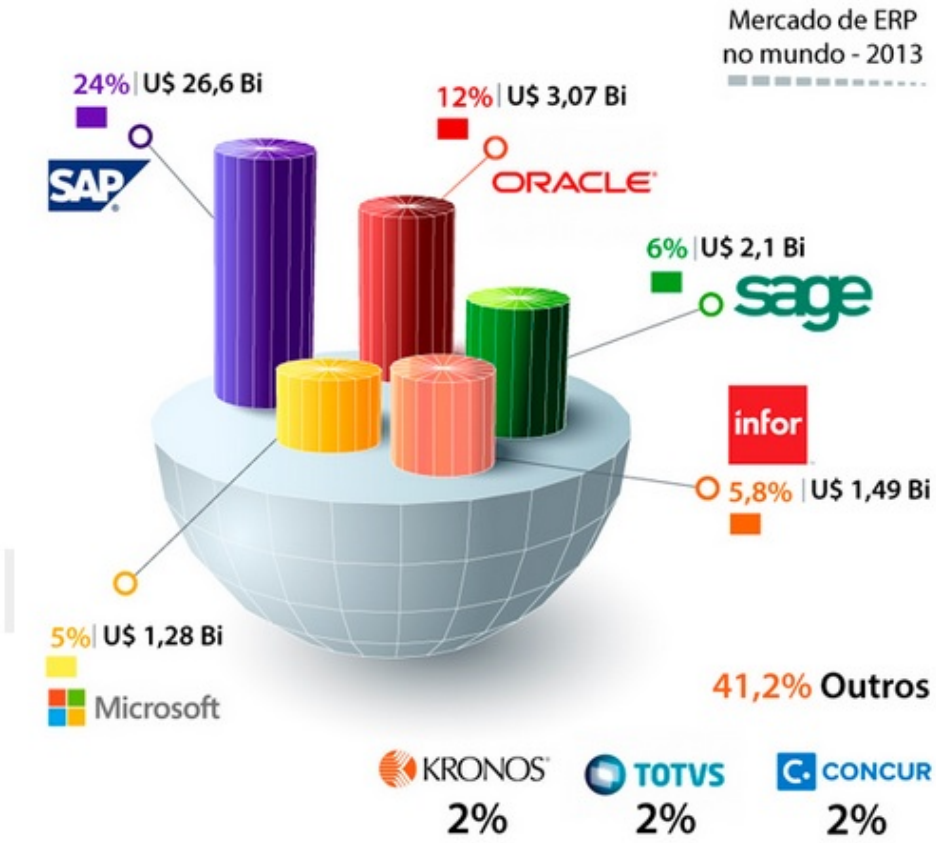

Figura 11 - Mercado de ERP no mundo

Fonte: PortalERP, 2014

Já na Figura 12 – Infográfico Mercado ERP no Brasil - podemos verificar que o cenário muda, quando se trata somente do mercado aqui do Brasil.

[4] **Portal ERP:** portal de conteúdo, serviços e relacionamento, entre pessoas e empresas envolvidas na área de Tecnologia da Informação com enfoque em sistemas de informação.

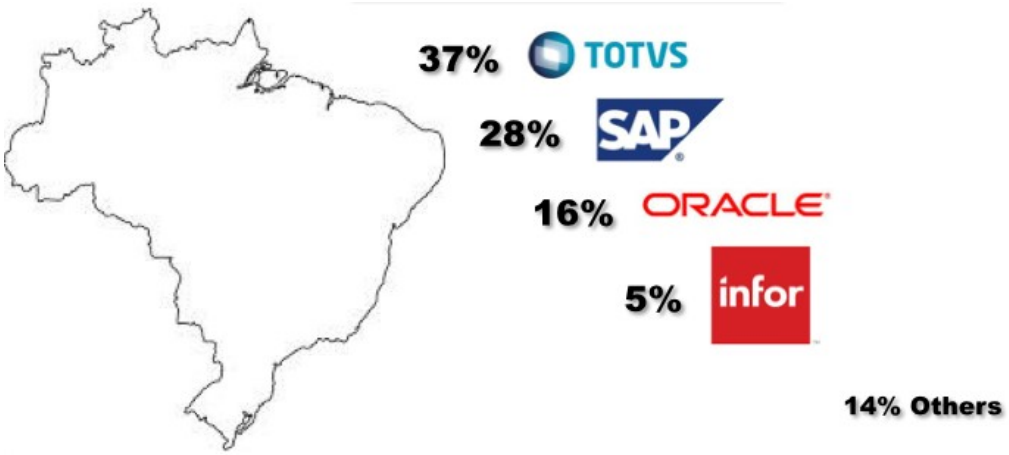

Figura 12 - Infográfico Mercado ERP no Brasil
Fonte: PortalERP, 2014

O PORTALERP (2014) apresenta também alguns dados importantes sobre essas principais fornecedoras de ERP no Brasil:

- TOTVS: fundada com o nome de Microsiga, a TOTVS é uma empresa de software, inovação, relacionamento e suporte à gestão. Líder absoluta no Brasil, com 48,6% de participação de mercado, e também na América Latina, com 34,5%. É a maior empresa de softwares aplicativos sediada em países emergentes e a 6ª maior do mundo no setor. Tem mais de 26 mil clientes ativos e conta com o apoio de aproximadamente 10 mil participantes em unidades próprias e franqueadas. Possui unidades próprias no México, Argentina e Portugal e está presente em 23 países;

- SAP: é uma empresa de origem alemã, criadora de softwares de gestão de empresas. Ao longo de quatro décadas, a SAP evoluiu de uma empresa pequena e regional a uma organização de alcance mundial. Hoje, a SAP é a líder global de mercado em soluções de negócios colaborativas e multiempresas. O principal produto da empresa, que emprega 74.497 pessoas em 2015, é o sistema integrado de gestão empresarial SAP ERP;

- Oracle: fundada em 1977, é fornecedora dos sistemas de software e hardware mais completos, abertos e integrados do mundo. Sua liderança no mercado de Tecnologia da Informação (TI) é resultado de um histórico constante de inovações tecnológicas. A companhia oferece soluções completas e integradas de TI, incluindo banco de dados, servidores de aplicação, aplicativos empresariais, soluções de colaboração, ferramentas para desenvolvimento de aplicações, bem como serviços

de consultoria, treinamento e suporte em mais de 145 países. Atende a cerca de 390 mil organizações em todo o mundo, entre elas, todas as empresas que figuram na lista Fortune 100 (dados de 2013);

- **Microsoft:** o Dynamics é o ERP da Microsoft. Fruto das aquisições e fusões que a empresa fez no começo dos anos 2000 de produtos voltados para gestão financeira, gestão de relacionamento com clientes e gestão de cadeia de suprimentos. Em 2009, somente com a venda dos produtos deste tipo de segmento a empresa faturou em torno de US$ 1,25 bilhões.

3. MODELO TEÓRICO E PRESSUPOSTOS DA PESQUISA

Um dos elementos mais importantes no processo de desenvolvimento, ou codificação, de um sistema ERP é o teste. Ele está presente não só no desenvolvimento, mas também nos processos anteriores, como planejamento e especificação e nos posteriores, como implantação e manutenção.

Segundo PRESSMAN (2002), “o teste de software é responsável por grande percentual de esforço técnico no processo de desenvolvimento de software”. O objetivo do teste de software é descobrir erros. Porém, para atingir esse objetivo, uma série de passos, etapas de testes devem ser realizadas, alinhados a cada etapa do processo de construção do software. Para cada passo de teste, o nível de abstração com a qual o software é submetido, é ampliado.

Ainda de acordo com PRESSMAN (2002), a distribuição recomendada para realização de testes com relação ao esforço total de um projeto é de 40%. Esta regra, a qual pode ser visualizada na Figura 13 – Distribuição do esforço num projeto de desenvolvimento - é chamada de “40-20-40”, sendo: 40% reservado para análise e projeto, 20% para codificação e o restante para testes. Porém, ressalta-se que a regra deve ser utilizada apenas como diretriz, pois pode variar de acordo com cada projeto.

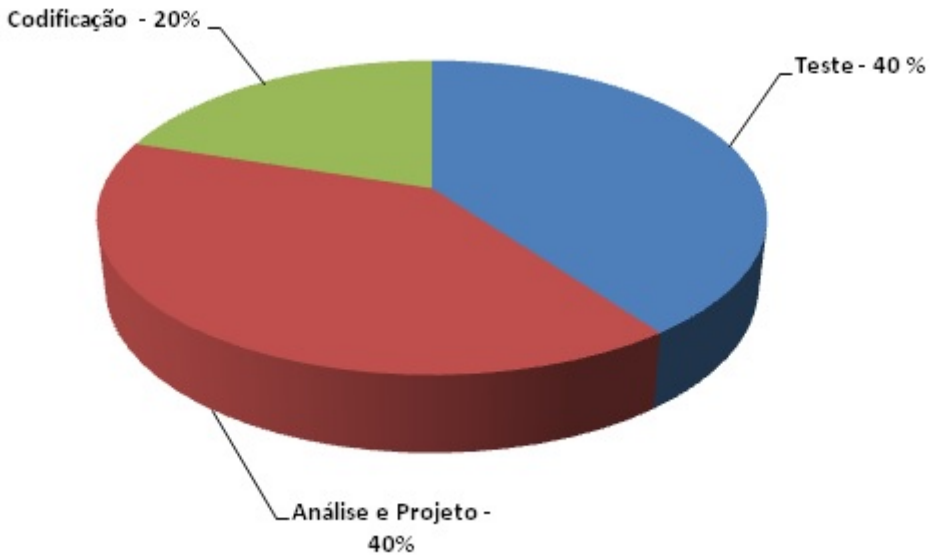

Figura 13 - Distribuição do esforço num projeto de desenvolvimento
Fonte: PRESSMAN, 2002

Para os sistemas integrados ou ERPs, onde o grau de complexidade é grande, devido à grande quantidade de elementos dos mais diferentes conceitos, o teste é exigência em todas as etapas, da especificação até a entrega final. Uma das ferramentas que está crescendo nos últimos anos, no que diz respeito a testes de software, é a Automação de Testes, também apresentada neste trabalho.

Os testes, seja em sistemas ERP ou quaisquer outros softwares, são baseados em premissas, documentos, técnicas e metodologias. Tanto na revisão bibliográfica quanto na análise de estudo de caso, são ilustrados tais elementos.

3.1. TESTES NO DESENVOLVIMENTO DE SISTEMAS ERP

Num desenvolvimento de sistemas ERP, os testes do sistema são longos e complexos. Inicialmente, são realizados os testes das funcionalidades dos módulos, passando depois pelos testes entre os vários módulos (teste dos processos), testes de integração, caso seja mantido alguma sistema atual da empresa ou adquirido algum sistema complementar. (OLIVEIRA, 2003).

Ainda dentro do processo de desenvolvimento de um software, existe a etapa de validação, com o intuito de reduzir os erros que serão liberados para o cliente.

> “O projeto de teste de software deve começar paralelamente ao projeto de desenvolvimento. As correções de defeitos encontrados em requisitos ou modelos custam menos. Ao reduzir a incidência de defeitos encontrados nos testes propriamente ditos, estamos otimizando essa atividade”. BASTOS et al. (2007).

Entende-se que os testes realizados nos softwares são considerados um projeto e ao serem desenvolvidos juntamente com o projeto de desenvolvimento do software, terá mais sucesso na sua execução.

> “Quem poderia garantir que um software testado pelos próprios desenvolvedores está corretamente testado? Com toda certeza, existem exceções, mas a melhor maneira de testar um software é ter um processo de teste claramente definido”. BASTOS et al. (2007).

A garantia de que um software seja bem testado pelo próprio desenvolvedor é pequena, já que sua especialidade não é a auditoria de software e sim seu desenvolvimento.

Garantindo a qualidade do produto com um processo de teste implementado corretamente, a empresa consegue reduzir o custo de manutenção dos softwares já que os erros encontrados antes do produto sair para o mercado são menores. Entretanto, é importante que essa tarefa seja executada por um profissional capacitado que possua o conhecimento do processo e técnicas dos testes.

> “As razões são muito simples, pois essa atividade, cada vez mais passa a ser executada por técnicos especialistas, e não mais por desenvolvedores cuja preocupação sempre foi de mostrar que os sistemas funcionam, ou usuários, que apenas querem saber se o produto que pediram foi entregue da forma definida por eles”. BASTOS et al. (2007).

Aumentando a cobertura dos testes, a probabilidade do usuário ou cliente encontrar erros é menor, sendo assim é possível reduzir os custos de correções.

Desta maneira é possivel identificar que a melhor forma de garantir a qualidade do produto e manter o custo de correção de defeitos em um nivel aceitavel, é realizar o projeto de teste junto ao projeto de desenvolvimento do software. Assim caso seja identificado algum erro no inicio do projeto, o custo de correção será muito baixo comparado à uma situação que o erro foi encontrado após o período de testes ou mesmo já em cliente. (BASTOS, 2007).

MYERS (2004) afirma o seguinte sobre defeitos:

> “- Os testes unitários podem remover entre 30% e 50% dos defeitos dos programas;
> - Os testes de sistemas podem remover entre 30% e 50% dos defeitos remanescentes;
> - Desse modo, os sistemas podem ir para produção ainda com aproximadamente 49% de defeitos;
> - Por último, as revisões de códigos podem reduzir entre 20% a 30% desses defeitos”. MYERS (2004).

A Figura 14 – Regra 10 de Myers - mostra que os custos de correção dos defeitos tendem a aumentar de acordo com tempo.

Um dos principais erros cometidos no processo de teste em sistemas ERPs refere-se às integrações. Os testes geralmente são realizados separadamente, já que há especialistas em um produto ou módulo e no outro.

Quando há integração entre eles, o teste deveria ser realizado à “quatro mãos”, o que muitas vezes não acontece.

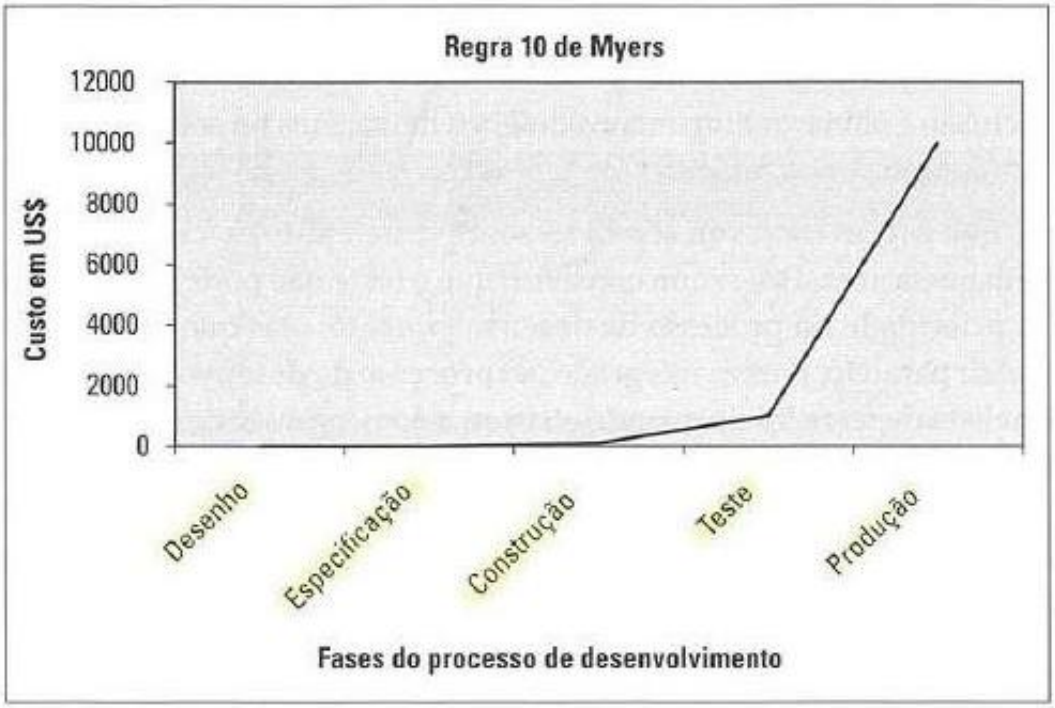

Figura 14 - Regra 10 de Myers
Fonte: BASTOS et al. (2007)

4. MÉTODOS DE PESQUISA

A metodologia aplicada na realização deste trabalho seguiu as seguintes etapas:

- Pesquisa qualitativa, através da revisão bibliográfica, que buscou identificar os fundamentos de testes de software, seus conceitos, ferramentas, documentações e novas metodologias. Além disso, também foi identificado o perfil dos sistemas ERP, suas características, benefícios e principalmente, os testes na fase de desenvolvimento;

- Estudo de caso em uma empresa que desenvolve ERP, trazendo a importância dos testes nos seus processos de desenvolvimento para itens de manutenção (correção do produto padrão) e inovação (novas funcionalidades).

Porém este estudo delimitou-se somente aos testes na etapa de desenvolvimento. Os testes, por exemplo, nas etapas de implantação no cliente ou em customizações, não foram detalhados.

4.1. PESQUISA QUALITATIVA

A elaboração deste trabalho foi conduzida pela pesquisa qualitativa, pois ela não se preocupa com representatividade numérica ou estática, mas sim com o aprofundamento da compreensão de um grupo social ou de uma organização, por exemplo.

“Ela busca basicamente entender um fenômeno específico em profundidade, considerando a existência de uma relação dinâmica entre realidade e sujeito investigado” (GOLDENBERG, 1999).

Por estarmos amparados pela pesquisa qualitativa, podemos classificá-la como sendo do tipo descritiva. Esta tem por desígnio: observar, registrar, analisar, classificar, interpretar e correlacionar fatos ou fenômeno investigado, de forma fiel e imparcial, sem interferir neles.

4.2. ESTUDO DE CASO

Segundo YIN (1994) os estudos de caso representam uma estratégia que responda às perguntas “como?” e “por quê?”, visando utilizar proposições orientadoras do estudo, a partir de questões primárias e secundárias, onde o pesquisador tem pouco controle sobre os eventos e o foco se encontra em fenômenos contemporâneos inseridos em algum contexto da vida real.

O estudo de caso deste trabalho baseia-se nas seguintes etapas:

- Verificação da aplicabilidade do estudo ao trabalho proposto;
- Levantamento de dados e informações da empresa pesquisada;
- Demonstração das amostras dos dados coletados, realizando um paralelo entre o embasamento teórico e o levantamento da pesquisa na empresa;
- Demonstração de como a empresa realiza os testes em seus processos;
- Ilustração através de alguns números em um cenário específico para concretizar e embasar os dados qualitativos;
- Análise e apuração dos resultados.

4.3. COLETA DE DADOS

As informações para elaboração deste trabalho foram coletadas da seguinte forma:

- Pesquisa para revisão bibliográfica: realizada em livros, dicionários, repositórios e periódicos especializados, além de outras publicações, com dados relacionados ao assunto em estudo;
- Pesquisa documental: realizada principalmente nos documentos técnicos e publicações da empresa pesquisada, relativos ao ano de 2016, com foco no tema principal do trabalho;

Através dos dados coletados buscou-se identificar e comparar os métodos e ferramentas utilizadas pela empresa com os obtidos nas referências bibliográficas. Com isso, foi possível identificar se a empresa adota os modelos mais atuais e aderentes e se estes surtem o efeito necessário ao negócio, ou seja, à qualidade do produto.

Por questões de concorrência e dificuldade de acesso às informações, foi possível coletar dados de apenas uma empresa na área de sistemas ERP. Com isso, a comparação dos dados obtidos dar-se-á com relação aos conteúdos levantados via pesquisa bibliográfica.

5. RESULTADOS DA PESQUISA

Para mostrar na prática a importância dos testes em sistemas ERP, foi realizado o estudo de caso numa empresa fornecedora deste tipo de software. Nela, existem os setores de Engenharia e *Software Quality Assurance* (SQA), que traduzem e exemplificam analogamente o que foi referenciado teoricamente nos itens anteriores com a prática adotada pela empresa em seu dia-a-dia, no que diz respeito a testes de software.

Um termo de confidencialidade foi apresentado, no qual foi esclarecido que os dados da empresa e dos colaboradores não seriam divulgados em nenhum momento e que a pesquisa não tinha qualquer caráter comercial, sendo apenas para fins acadêmicos.

O modelo do termo está disponível no APÊNDICE A, ao final deste trabalho. Além disso, uma cópia, assinada em 15 de maio de 2017, encontra-se em poder do pesquisador.

5.1. PERFIL DA EMPRESA PESQUISADA

A pesquisa foi realizada em uma das maiores empresas de software, serviços e consultoria do país que, ao longo dos anos, se tornou uma das principais marcas no ramo de sistemas ERP a nível nacional e até mesmo internacional.

Provedora de soluções de negócios para empresas de todos os portes, porém com principal foco em pequenas e médias empresas, atuando com softwares de gestão, plataformas de produtividade e colaboração, hardware, consultoria e *Cloud Computing*.

Apresenta soluções de TI para diversos segmentos, sejam eles dos setores primário, secundário e terciário, como por exemplo: Agroindústria, Construção e Projetos, Distribuição e Logística, Educacional, Serviços Financeiros, Jurídico, Manufatura, Saúde, Varejo, Serviços, entre outros.

5.2. RELEVÂNCIA DOS TESTES DE SOFTWARE E DA QUALIDADE

A busca pela qualidade é um dos fatores chave para o sucesso do negócio. Ela interfere diretamente na satisfação dos clientes e, consequentemente, nos lucros da empresa.

Neste sentido, a empresa analisada preocupada com as questões voltadas à qualidade, possui diversas formas de gerir, controlar e manter o nível de excelência em seus produtos e serviços. Para isso conta com ferramentas importantes, como por exemplo:

- Política da qualidade: a Alta Direção deve assegurar que os objetivos da qualidade, incluindo aqueles necessários para atender aos requisitos do produto, sejam estabelecidos nas funções e nos níveis pertinentes da organização. Os objetivos da qualidade devem ser mensuráveis e consistentes com a política da qualidade. Além disso, todos os colaboradores devem ter conhecimento sobre a Política da Qualidade.

- Procedimento do sistema de Gestão da Qualidade: têm por objetivo documentar os processos generalizados e vinculados aos demais processos do escopo de certificação NBR ISO 9001:2008.

- Manual da Qualidade: É de responsabilidade da equipe de *Quality Assurance* a emissão deste manual, que tem por objetivo:

  - Demonstrar os processos do Sistema de Gestão da Qualidade;

  - Demonstrar a sequência e a inter-relação dos processos;

  - Demonstrar os critérios e métodos para assegurar que a operação e o controle dos processos sejam eficazes;

  - Demonstrar a disponibilidade de recursos e informações necessárias, para apoiar a operação e o monitoramento dos processos.

Com relação aos testes, para cada planejamento e especificação de itens de codificação, há uma dependência obrigatória com a atividade de teste, seja manual, automatizado ou sistêmico. O percentual dedicado a esta atividade, com relação ao total de horas, seja do requisito ou de um chamado, equivale geralmente de 30 a 40% do tempo.

5.3. ESTRUTURA DA EQUIPES DE TESTE

A Equipe de Teste, também chamada de Engenharia ou SQA, é subdividida de acordo com os módulos do próprio ERP. Então, dentro do time de SQA temos, por exemplo, a Controladoria, o Faturamento, o Fiscal, RH, etc. Junto a estas, há também uma equipe especializada em Automação. Para cada equipe, há um Líder de Testes e, abaixo dele, os Analistas de Testes.

A seguir, seguem as características e responsabilidades de cada função:

Líder de testes:

- Responsável pelo planejamento da capacidade e demanda da equipe (balanceamento da Inovação e Manutenção);
- Gestão do portfólio de testes (projetos, chamados, atualizações e versões);
- Aprovação das execuções e/ou plano de testes;
- Gestão da qualidade do produto;
- Acompanhar e controlar os indicadores de qualidade e produção;
- Promover a melhoria contínua dos processos e ferramentas;
- Apoio técnico de primeiro nível aos analistas de testes e testadores;
- Responsável pela escalação de primeiro nível.

Analista de testes: este papel também pode ser denominado Designer de Teste. É responsável por:

- Verificar se o ambiente onde será realizado o teste se encontra atualizado;
- Identificar os itens de teste a serem avaliados pelo esforço de teste;
- Definir os testes apropriados e os dados de testes associados;
- Coletar e gerenciar os dados de teste;
- Avaliar o resultado de cada ciclo;
- Também pode apoiar o Testador nos testes integrados e sistêmicos.
- Garantir que a metodologia de desenvolvimento é seguida.

Testador: é responsável pelas atividades de condução dos testes necessários e registro dos resultados desses testes. Isso inclui:

- Implementar testes;
- Configurar e executar os testes;
- Verificar a execução dos testes;

- Verificar e registrar os resultados gerados pela execução;
- Analisar os erros de execução;
- Garantir que a metodologia de desenvolvimento é seguida.

Analista de testes de Automação: executar/criar os testes conforme especificado, para isso deve:

- Atualizar os scripts existentes afetados;
- Criar os novos scripts/suítes conforme casos de testes planejados;
- Executar os testes automatizados e, enviar *logs* e resultados para o analista de testes homologar;
- Reprovar a etapa de testes automatizados quando existir erros que impeçam o teste automatizado de ser criado;
- Quando a reprovação impactar os testes integrados, informar o analista de testes responsável.

5.4. METODOLOGIA DE TESTES UTILIZADA

Toda a empresa, mesmo em filiais diferentes, segue a mesma metodologia no que diz respeito aos fluxos de testes. A diferença está entre as etapas de Inovação (novos produtos, funcionalidades e ferramentas) e Manutenção (correção dos sistemas e funcionalidades já existentes). Nos itens a seguir serão demonstrados estes fluxos e as diferenças entre eles.

Vale ressaltar que os produtos podem ter, numa mesma versão, vários *releases* [5]. Nestes casos, os processos de testes são a última etapa antes da liberação de cada *release*.

Uma preocupação relevante da empresa é com relação às horas gastas com manutenção. O ideal, para empresas deste ramo, é sempre investir em Inovação. O processo de manutenção é tratado como custo, sempre. Já a inovação é tratada como investimento. Por isso, quanto mais estável o produto padrão estiver, menos se gastará com manutenção e estes recursos poderão ser melhor distribuídos em inovação.

[5] **Release:** liberação ou lançamento de novas funcionalidades do produto. Cada versão pode conter vários releases.

5.4.1. PROCESSO DE INOVAÇÃO

O processo de inovação é responsável pela promoção da evolução dos produtos em conformidade com as diretrizes estratégicas estabelecidas pela organização, visando as necessidades do mercado. Este processo, conforme demonstrado na Figura 15 – Processo de Inovação - inicia-se a partir de uma demanda originada por um Cliente ou pelos próprios participantes (sugestão de melhoria), alta direção, Arquiteto de Relacionamento, ou uma tendência de mercado que gera um planejamento da versão/*release*.

Os subprocessos de Gestão de Projetos, Análise de Requisitos, Codificação, Testes, Capacitação de Especialista e Documentação / Treinamento / Certificação / Tradução são iterativos e incrementais, ou seja, podem ser executados mais de uma vez durante o ciclo de desenvolvimento do projeto.

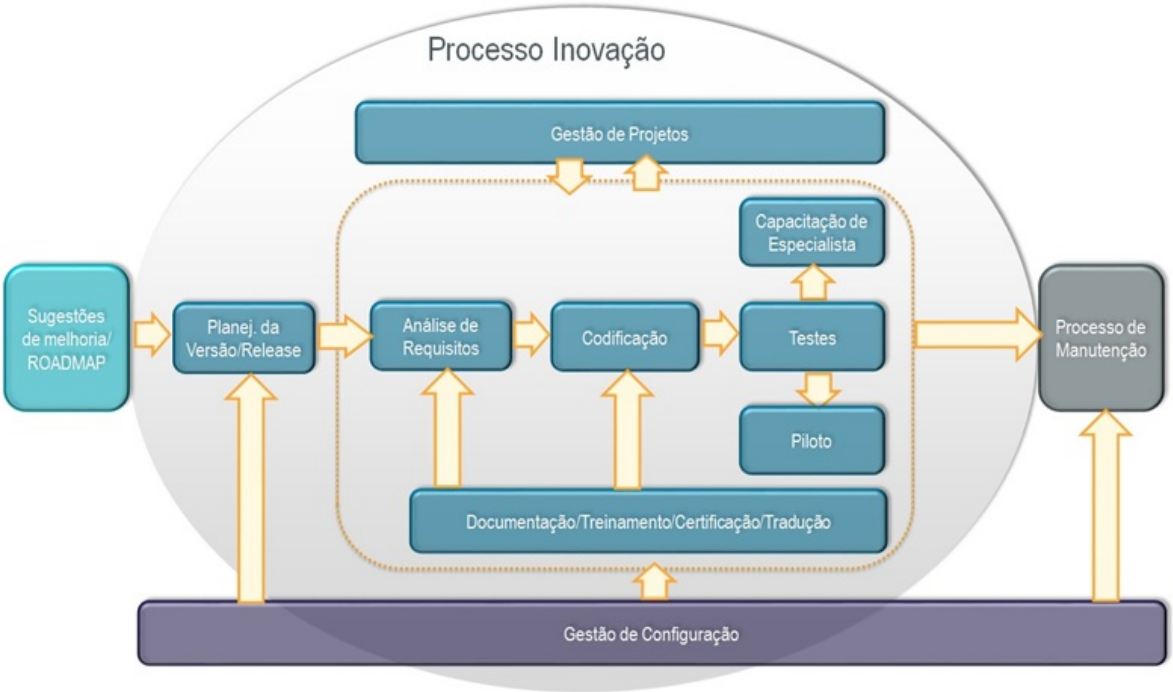

Figura 15 - Processo de Inovação
Fonte: documento da empresa pesquisada

O subprocesso de Testes, ilustrado na Figura 16 – Etapas de Testes (Processo de Inovação) - compreende a etapa de validação do sistema, é composto pelas atividades:

- Planejar Testes;
- Especificar Testes;
- Executar Teste Integrado;
- Executar Teste Sistêmico;
- Monitoramento e Controle.

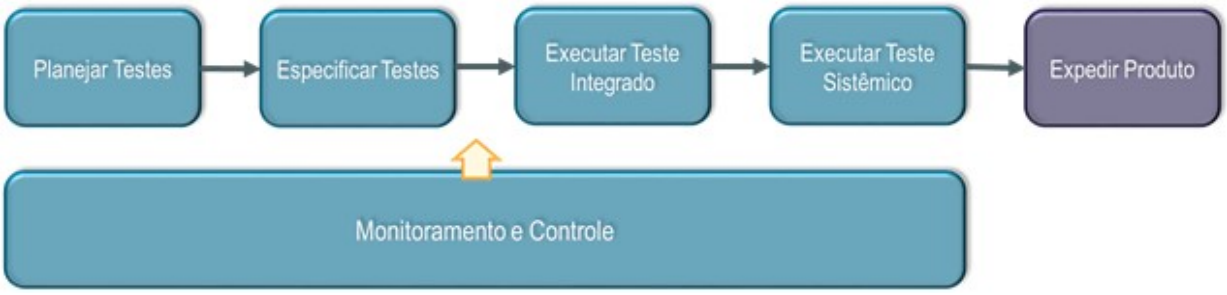

Figura 16 - Etapas de Testes - Processo de Inovação
Fonte: documento da empresa pesquisada

5.4.1.1. PLANEJAMENTO DE TESTES

Planejar testes tem por objetivo dimensionar o esforço, os recursos necessários e definir as diretrizes do que será testado. Registra o que será testado e como os testes serão realizados, além dos aspectos globais relacionados.

O plano deve ser elaborado até o início da execução da etapa de Teste Integrado e Sistêmico. Obrigatoriamente, para cada uma destas etapas, deve ser gerado um Plano de Testes por *release.* O Plano de Testes Integrado deve conter pelo menos uma execução para cada requisito e o Plano de Testes Sistêmico deve conter pelo menos uma execução. As execuções (agrupadores de casos de teste a serem testados) poderão ser associadas ao plano após a criação do mesmo.

O plano pode ser criado por participantes que possuam a permissão específica para criação do Plano de Testes, conforme definido internamente.

Os status do Plano de Testes podem ser: Aprovado, Rascunho, Aguardando Aprovação ou Concluído.

5.4.1.2. ESPECIFICAÇÃO DE TESTES

Especificar Casos de Testes compreende a identificação e detalhamento das condições de testes necessárias para validação do sistema. Identificar Casos de Testes é verificar as condições básicas de testes para validação do sistema.

Os casos de testes devem ser identificados e registrados por participantes que possuam a permissão na ferramenta de criar novos casos de testes. Na criação destes Casos de Testes deve ser selecionado o status identificado. Após identificação realizada, pode ser iniciado o detalhamento ou revisão desses casos de testes. O detalhamento dos Casos de Testes deve conter informações necessárias para a execução de um teste.

A Especificação dos Casos de Testes, transcrita em ferramenta interna, com acesso disponibilizado à todos os analistas, seja de codificação ou teste, contém informações detalhadas quanto à execução dos testes, baseados na especificação do requisito, a fim de assegurar que o teste esteja voltado ao que foi especificado.

Os status dos Casos de Teste podem ser:

- Liberado;
- Rascunho;
- Aguardando Aprovação;
- Obsoleto;
- Identificado;
- Revisar.

Após o detalhamento dos Casos de Testes, devem ser criadas as execuções de testes, e essas execuções obrigatoriamente devem ser associadas aos devidos planos de Teste Integrado ou Teste Sistêmico.

- A execução do Teste Integrado deve ser criada por requisito;
- A execução do Teste Sistêmico deve ser criada por *release*.

A aprovação da execução de teste é opcional. Porém é obrigatória uma aprovação no processo, podendo ser da Execução de Testes ou do Plano de Testes, conforme internamente.

O registro da rastreabilidade dessa atividade deve ser feito por uma das seguintes opções:

- Indicando o link ou documento (anexo) do Relatório de Resultados de Teste da execução criada;

- Associando cada Caso de Teste criado ao seu respectivo requisito.

5.4.1.3. EXECUÇÃO DE TESTE INTEGRADO

Teste Integrado é a fase do teste de software em que módulos são combinados e testados em grupo, tem como propósito verificar os requisitos funcionais, de desempenho e de confiabilidade, podendo detectar defeitos entre os componentes do sistema. Ele sucede o Teste Unitário e antecede Teste Sistêmico.

Nesta etapa o testador deve consultar as execuções direcionadas a ele, realizar a análise da evidência do teste de unidade, validação funcional e executar os testes.

As execuções criadas e associadas aos planos de testes na atividade de Especificar são executadas nesta atividade. No teste integrado esta execução ocorre por requisito. A evidência de execução será gerada automaticamente conforme os testes realizados, contendo as informações produto, versão, ambiente utilizado, sequência dos testes realizados e resultados obtidos. Para complementar as evidências geradas, podem ser anexados *print screen* de telas e/ou *logs*.

O status da Execução de Testes pode ser:

- Não Executado;

- Em Progresso;

- Pronto para Execução;

- Concluído.

Para projetos que envolvem integração entre produtos, são considerados os seguintes pontos:

- Deve ser evidenciado o produto e versão de ambos os softwares;

- Cada linha de produto será responsável por gerar a evidência de teste do seu produto;

- Os testes devem ser realizados nos ambientes corporativos disponíveis, a fim de garantir a funcionalidade completa das partes envolvidas, exatamente como é realizado pelo cliente;

- Recomenda-se que este teste seja planejado para ocorrer simultaneamente;

- Devem ser criados diferentes casos de teste para cada etapa da integração entre os produtos, identificando no campo de “Pré-condição” quais são os casos de teste que precedem o caso de teste atual. Recomenda-se que o caso de teste seja atribuído ao responsável por sua execução.

Caso haja necessidade de automatizar algum teste funcional são considerados os pontos destacados abaixo:

- O próprio testador poderá criá-lo, atualizá-lo e executá-lo ou solicitar a criação para a área responsável pela automação;

- Devem ser registradas quais foram as *suites* executadas e os resultados atingidos;

Com relação aos defeitos encontrados, são seguidas as seguintes diretrizes:

- Durante a execução dos testes, se encontrados defeitos, estes devem ser registrados no Projeto da Equipe. Os defeitos que não forem corrigidos no *release* atual devem ser avaliados pelo Analista de Testes e podem ser movidos para serem corrigidos no próximo *release*;

- Sugere-se que não sejam movidos para correção em outro *release*, defeitos de criticidade Crítica e Alta;

- Inconsistências referentes à ortografia, padrão e/ou arquitetura textual deve ser registrado com um defeito para a equipe de Documentação;

- Referente à falta ou distorção de informação no conteúdo deve ser registrado um defeito para a equipe de Inovação;

- O Testador/Analista de Testes deve aprovar o Documento Técnico e o Documento de Referência via *workflow*, caso estes artefatos ainda não tenham sido aprovados pela área de Inovação;

- Após a atividade de Executar Teste Integrado e antes da expedição do produto, o Analista de Documentação deve oficializar a publicação de todos os

documentos do produto. É de sua responsabilidade consultar o calendário corporativo e acompanhar a *release* de entrega planejada dos requisitos para realizar esta atividade.

A segregação de funções na execução das atividades de Codificação, Aprovação do Teste de Unidade e Realização do Teste Integrado é respeitada, ou seja, o recurso que executa o Teste Integrado não pode executar a atividade Codificar e nem a atividade Aprovar Teste de Unidade.

5.4.1.4. DEFEITOS

Um defeito pode ocorrer devido à omissão de informações, definições de dados ou comandos/instruções incorretas dentre outros fatores. Se um determinado defeito não for encontrado, pode causar uma falha no funcionamento do software.

São considerados “defeitos” no processo de desenvolvimento todos os erros, *bugs*, inconformidades encontradas no decorrer do desenvolvimento dos projetos.

O defeito pode ser identificado e gerado em qualquer uma das etapas do processo de desenvolvimento, sendo que independente da etapa que está sendo executada pode ser aberto um defeito para alguma etapa anterior que pode ter causado ou originado o defeito.

Para defeitos encontrados até o Teste Integrado, os mesmos deverão ser vinculados requisito originador do projeto de inovação, para os defeitos encontrados durante o Teste Sistêmico, estes deverão ser vinculados projeto de Teste Sistêmico.

A abertura de defeitos deve ser pautada baseada nos tipos descritos nos quadros de 5 (cinco) a 12 (doze), encontrados no ANEXO A deste documento.

5.4.1.5. MONITORAMENTO E CONTROLE DE TESTES

A etapa de Monitoramento e Controle tem como objetivo acompanhar o progresso das atividades de teste, desde o Planejamento até as Execuções.

Para fazer esse acompanhamento, podem ser utilizados os filtros e modos de visualização dentro de cada área no sistema, e também podem ser

gerados diversos relatórios por meio da área “Relatórios” que permitem visualizar os dados de teste sob várias perspectivas.

Os dados indicados para acompanhamento são:

- Execuções de Teste associadas a um Plano de Testes: Verificar se existe uma (ou mais) Execução de Teste para cada requisito do Plano, ou uma (ou mais) Execução de Testes quando for Plano de Testes Sistêmico;

- Verificar se os Planos de Teste possuem informações válidas nos campos obrigatórios;

- Verificar se a atividade Executar Testes possui uma ou mais Execuções associadas;

- Verificar se as Execuções de Teste existentes possuem somente Casos de Teste com o Status “Liberado”;

Também é possível acompanhar o progresso das Execuções de Teste por meio dos relatórios disponíveis, os quais disponibilizam diversas opções de filtros e visualizações. São eles:

- Relatório de Execução dos Testes – Sumário: este relatório apresenta o sumário de uma execução de testes (como por exemplo, uma visão geral do status e do progresso);

- Relatório de Execução dos Testes – Esforço: este relatório apresenta as estatísticas detalhadas de uma execução de testes por Esforço;

- Relatório de Execução dos Testes – Cobertura: este relatório apresenta as estatísticas detalhadas de uma execução de testes por Cobertura;

- *Scorecard* de Execução de Teste por Testador: o relatório de *Scorecard* fornece uma visão geral do progresso e esforço de uma execução de testes agrupado por Testador;

- *Scorecard* de Execução de Teste por Pasta: o relatório de *Scorecard* fornece uma visão geral do progresso e esforço de uma execução de testes agrupado por pasta;

- Resultados de Teste: este relatório apresenta uma lista de resultados de testes;

- Resultados de Testes Impactados por *Issues*: este relatório apresenta uma lista de resultados de testes que foram impactados por *Issues* durante uma execução de teste.

Neste momento não são definidos indicadores e metas para o processo de testes. As informações são disponibilizadas numa ferramenta para gestão e acompanhamento das equipes.

5.4.2. REALIZAÇÃO DE TESTES - PROCESSO DE MANUTENÇÃO

A Manutenção de Software é uma etapa que consiste na correção de erros que não foram previamente detectados e envolve mudanças no software com a finalidade de corrigir defeitos e deficiências encontradas durante o uso do software no cliente. Na Figura 17 – Fluxo do processo de manutenção - podemos observar como funciona o fluxo das etapas do processo de manutenção.

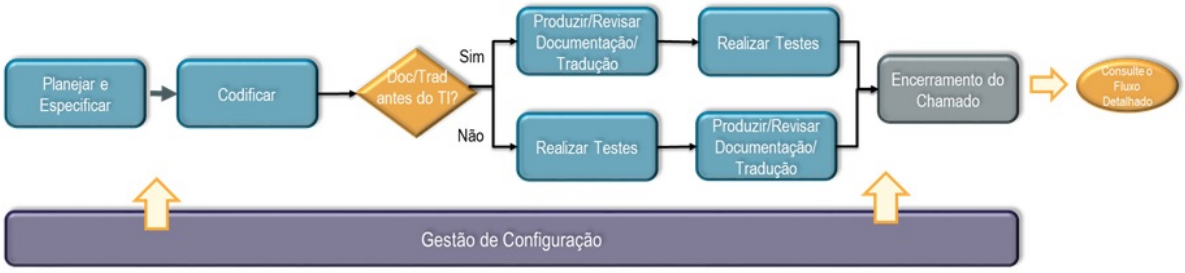

Figura 6 - Fluxo do processo de manutenção
Fonte: documento da empresa pesquisada

A primeira etapa consiste em Planejar e especificar. O objetivo desta atividade é planejar as atividades e recursos disponíveis para realizar a manutenção, conforme solicitação e detalhar uma solução que atenda os requisitos especificados, definindo uma estrutura implementável para um produto de software. Deve-se observar nesta atividade alguns padrões de mercado, o qual objetivo é garantir que a solução proposta é a mais adequada.

Dentro da etapa de codificação, ao final do seu processo, o analista líder deve realizar a aprovação do teste de unidade. Neste momento o analista de sistemas/líder de desenvolvimento, além de verificar as evidências geradas pelo desenvolvedor na execução do teste de unidade e avaliar no fonte a existência de

códigos maliciosos que possam oferecer riscos ao cliente quanto à confiabilidade das informações geradas pela solução, pode realizar novos testes para complementar os testes já realizados pelo desenvolvedor. Caso sejam realizados novos testes, deve-se anexar a evidência ou descrever na etapa os testes realizados e o resultado obtido.

Além da aprovação do teste de unidade, o Analista de Sistemas/Líder de Desenvolvimento deve:

- Aprovar as alterações realizadas em Dicionário de Dados testando o script em conjunto com as alterações do fonte;

- Aprovar o Documento Técnico, o Documento de Integração e revisar/aprovar a Base de Conhecimento. Caso o Documento Técnico não seja aprovado pelo Analista de Sistemas nesta atividade, pode ser aprovado pelo Testador/Analista de Testes na etapa Realizar Teste Integrado.

Caso sejam encontrados defeitos durante a aprovação do teste de unidade, estes deverão ser registrados na ferramenta e direcionados a equipe responsável para devida correção.

5.4.2.1. CRIAÇÃO E REVISÃO DE CASOS DE TESTES

Nesta atividade devem ser identificados e detalhados os Casos de Testes e verificadas as rotinas críticas para automatização.

No processo de Manutenção, a realização da identificação e do detalhamento dos casos de testes é opcional, caso já tenham sido indicados os casos de testes ou registrada uma breve descrição do que deve ser testado para evidenciar a correção do problema reportado, na atividade Planejar e Especificar.

Identificar Casos de Testes é verificar quais são as condições de testes necessárias para validação do sistema. Caso seja realizada a identificação e o detalhamento dos casos de testes no processo de manutenção, esta atividade pode ser executada nas etapas Criar/Rever Casos de Testes ou Realizar Teste Integrado.

5.4.2.2. REALIZAÇÃO DE TESTE INTEGRADO

O teste integrado é a fase do teste de software em que módulos são combinados e testados em grupo. Ele sucede o teste de unidade e antecede o teste sistêmico. Tem como propósito verificar os requisitos funcionais, de desempenho e de confiabilidade do sistema, podendo detectar defeitos entre os componentes do sistema.

Nesta etapa o testador deve realizar a análise da evidência do teste de unidade, a validação funcional e evidenciar o teste.

O teste deve ser evidenciado com informações: do produto, da versão, do banco de dados em que este componente será testado, da sequência dos testes realizados e dos resultados atingidos. Esta evidência pode ser gerada através do *template* proposto ou outros meios, desde que observados os itens mínimos citados acima. Para complementar as evidências geradas poderão ser anexados *print screen* de telas e/ou *logs*.

Para solicitações que envolvem integração entre produtos também deverá ser evidenciado o produto e versão de ambos os softwares. Cada linha de produto será responsável por gerar a evidência de teste do seu produto. Estes testes devem ser realizados nos ambientes corporativos disponíveis, a fim de garantir a funcionalidade completa das partes envolvidas, exatamente como é realizado pelo cliente. Recomenda-se que este teste seja planejado para ocorrer simultaneamente, ou seja, as equipes de teste responsáveis por cada produto realizam o teste de integração ao mesmo tempo.

Caso seja identificada a necessidade de automatizar algum teste funcional, o próprio testador poderá criá-lo, atualizá-lo e executá-lo ou solicitar a criação para a área responsável pela automação. Neste caso também deverá ser registrado quais foram as suítes executadas e os resultados atingidos.

Nesta atividade também pode ser efetuada a validação do Documento Técnico e do Documento de Referência caso não tenha sido realizada pela equipe de manutenção.

Caso sejam encontrados defeitos durante a realização dos testes, estes deverão ser registrados na ferramenta e direcionados a equipe responsável para devida correção.

Inconsistências referentes à ortografia, padrão e/ou arquitetura textual devem retornadas para a equipe de Documentação. Referentes à falta ou distorção de informação no conteúdo devem ser retornadas para a equipe responsável pela elaboração do documento.

Sendo necessário replicar o chamado para versões superiores ou inferiores, as atividades a serem executadas devem seguir o mesmo processo de manutenção, a partir da atividade de codificação.

Outras informações importantes do fluxo de teste integrado:

- A atividade de Criar e Rever Casos de Testes pode ser executada em conjunto com esta atividade, porém recomenda-se que a criação e revisão dos casos de testes sejam finalizados antes do início dos testes integrados e somente pequenos ajustes sejam realizados durante a execução dos testes;

- A segregação de funções na execução das atividades de Codificação, Aprovação do Teste de Unidade e Realização do Teste Integrado deve ser respeitada, ou seja, o recurso que Realizar o Teste Integrado não pode ter executado a etapa Codificar Componentes / Programas e nem a etapa Aprovar Teste de Unidade;

- A liberação especial pode ser realizada ao cliente em casos críticos, onde não há possibilidade de aguardar por uma liberação oficial. Esta liberação pode ser feita a partir da atividade de Codificação, onde deve ser comunicado ao cliente que liberações feitas no nível de codificação não contemplam a aprovação do teste feito pelo desenvolvedor, e caso ele aceite, todos os riscos são de responsabilidade do cliente. A liberação especial deve ocorrer via iteração de chamado, com a possibilidade de anexar o pacote ou informar o *link* para o repositório oficial de liberação ao cliente. É importante que após a liberação realizada, seja dado continuidade ao fluxo até sua etapa final;

- Após a realização desta etapa ou da etapa de Produzir/Revisar Documentação o Analista de Testes/Testador ou o Analista de Documentação deve disponibilizar os documentos, oficializando a publicação antes da expedição do produto. Sugere-se que seja consultado o calendário corporativo e acompanhado o planejamento para entrega do *update*;

- Em datas planejadas também ocorrem os testes sistêmicos, conforme atividade Realização de Teste Sistêmico.

5.4.2.3. REALIZAÇÃO DE TESTE SISTÊMICO

O teste sistêmico ou de sistema é uma fase do processo de teste de software em que o sistema já completamente integrado é verificado quanto a seus requisitos num ambiente correspondente o máximo possível ao objetivo final, ou o ambiente de produção. Está no escopo da técnica de teste de caixa-preta, e dessa forma não requer conhecimento da estrutura lógica (interna) do sistema.

Testes de sistemas podem ser baseados em especificação de riscos e/ou de requisitos, processos de negócios, casos de uso, dentre outras descrições de alto nível do comportamento, interações e recursos do sistema, e devem também tratar requisitos funcionais e não funcionais.

Após aprovação de todos os testes integrados, o testador deve receber o ambiente de configuração e documentação necessária para realização do teste sistêmico do produto que contempla: análise de evidência de teste integrado, validação funcional e teste de instalação.

Este nível de testes ocorre em períodos planejados de acordo com o calendário corporativo, tomando como base cada uma das versões planejadas onde são liberadas todas as manutenções efetuadas no período.

Recomenda-se neste nível de teste realizar os tipos de teste: instalação, performance, conversão e integração entre sistemas.

Em caso de execução de testes de conversão um novo ambiente (vazio) de testes sistêmicos deve ser gerado com base nos programas de conversão e mídia de instalação.

Caso sejam encontrados defeitos durante a realização dos testes, estes deverão ser registrados na ferramenta e direcionados a equipe responsável para devida correção.

Inconsistências referentes à ortografia, padrão e/ou arquitetura textual devem reportadas para a equipe de Documentação. Referente à falta ou distorção de informação no conteúdo, deve ser reportada para a equipe responsável pela elaboração do documento.

Após aprovação do teste sistêmico, o produto e suas documentações são liberados para expedição.

5.4.3. AUTOMAÇÃO DE TESTES

O teste automatizado trata-se de programa ou script executável que roda a rotina “de negócio” a ser testado e faz verificações automáticas em cima de efeitos colaterais obtidos.

O teste automatizado visa otimizar o processo de testes, evitando o retrabalho, aumentando a confiabilidade do produto.

Na execução do processo de teste automatizado, são percorridas as seguintes etapas:

- demanda (chamado ou requisito);
- análise da demanda (FNC – Ficha de Não Conformidade ou Requisito);
- desenvolvimento: manutenção ou inovação;
- elaboração ou alteração do Mapa Mental;
- criação ou alteração dos Casos de testes;
- criação ou atualização dos Scripts de teste Automatizados;
- execução do teste pelo Robô de testes;
- aprovação e expedição do pacote.

O fluxo destas etapas está ilustrado na Figura 18 – Fluxo de automação de casos de testes.

5.5. TESTES EM NÚMEROS

Para ilustrar o que foi descrito anteriormente, sobre como a empresa pesquisada realiza os seus testes, foram coletados alguns números que podem ser verificados nos Quadros 1, 2, 3 e 4.

Em tempo, cabe dizer que no transcorrer deste estudo de caso, a empresa adotou uma nova metodologia e também uma nova ferramenta de controle. Por este motivo, foram extraídos apenas os dados de um determinado período.

Ainda vale ressaltar que os dados obtidos foram retirados de um único módulo, versão / release e produto em específico e que o período pesquisado equivale apenas aos meses de Janeiro e Fevereiro de 2017.

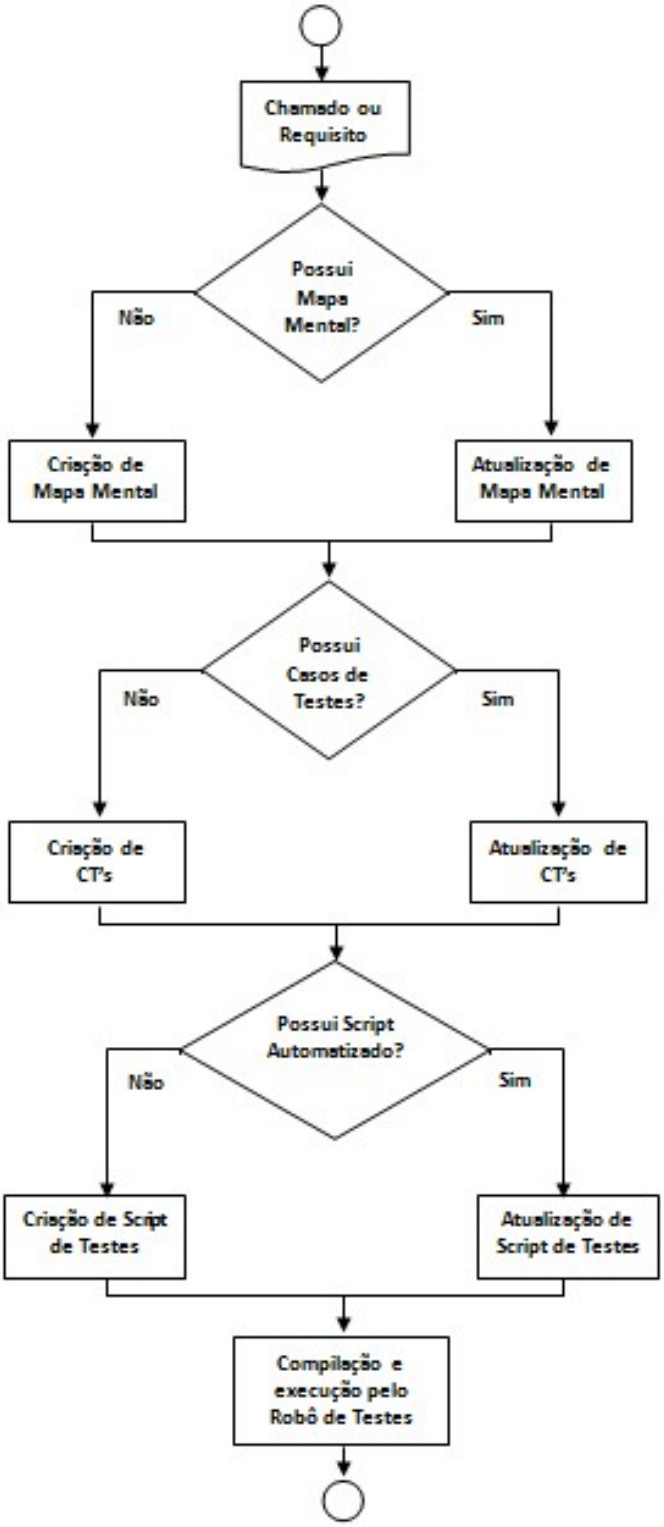

Figura 7 – Fluxo de automação de casos de testes
Fonte: elaborado pelo autor, 2016

No Quadro 1 podemos verificar os números de Manutenção, ou seja, aqueles erros que são encontrados no produto padrão, confrontando os chamados

abertos internamente (“dentro de casa”) com os abertos por clientes (chamados externos). O número de rejeições corresponde aos erros recorrentes, ou seja, aqueles que foram codificados, porém persistiram, sendo identificados pela equipe responsável pelos testes.

Quadro 1 – Chamados de Manutenção

Chamados de Manutenção			
Chamados clientes	**Chamados internos**	**Nº rejeições**	**Total**
26	133	13	**172**

Pelos números apresentados no Quadro 1, percebe-se que os testes internos são extremamente importantes, já que somados os chamados internos e o número de rejeições, estes representam 60% a mais do que os erros encontrados pelos clientes.

Já no Quadro 2 estão expostos os números dos itens de Inovação, ou seja, aqueles requisitos que foram desenvolvidos no período informado, mas que entram no produto padrão somente no release seguinte.

Os defeitos correspondem aos erros encontrados nos testes dos requisitos. Estes defeitos retornam para a equipe de desenvolvimento e são novamente testados para validar a sua correção.

Quadro 2 – Itens de Inovação

Inovação			
Nº requisitos	**Qtde horas codificação**	**Nº defeitos**	**Qtde horas testes**
6	462,5	48	160,5

Pelos números apresentados no Quadro 2 identificamos o quão é difícil, seja por falta de mão-de-obra, seja por prazo ou priorização, adotar a proporção do “40-20-40” proposta por PRESSMAN.

Na prática, a etapa de testes costuma ser uma das primeiras a sofrer “cortes” quando se percebe que o prazo está apertado. Entretanto, somente pelo quadro apresentado, não é possível determinar se este foi o real motivo, mas

identificou-se que as horas reservadas para testes foram quase 3 vezes menos do que as horas destinadas à codificação destes 6 requisitos.

Para consolidar o produto, ao final de cada release e antes da expedição, são realizados Testes Sistêmicos. No Quadro 3, estão os números de erros encontrados no módulo *versus* esforço realizado.

Quadro 3 – Testes Sistêmicos

Teste Sistêmico	
Nº defeitos	**Qtde horas testes**
15	80

No período do Sistêmico, tanto as equipes de desenvolvimento quanto as equipes de testes são focadas em corrigir os defeitos remanescentes para que a nova versão chegue os clientes com o menor número de erros possível.

Paralelamente aos testes demonstrados nos quadros anteriores, todos eles realizados manualmente, estão os testes automatizados. O robô de testes executa a rotina diariamente e abre o chamado automaticamente em caso de erro encontrado.

No Quadro 4 são demonstrados os erros encontrados no período citado com base nas rotinas automatizadas para aquele módulo:

Quadro 4 – Testes Automatizados

Automação	
Nº rotinas automatizadas	**Qtde defeitos**
45	15

As rotinas de testes automatizados são implementadas constantemente, buscando abranger o maior número de linhas de código possíveis. Com isso, a tendência é que a quantidade de defeitos encontrados por elas aumente e o número de defeitos encontrados por testes manuais diminua.

Com base nos quadros e a partir dos números levantados, mesmo que simplificados e retratando um cenário específico, podemos verificar como os testes são importantes em todos os processos, seja Manutenção ou Inovação, e quanto de esforço é gasto comparando-se desenvolvimento e testes.

No geral, percebe-se a preocupação da empresa em desenvolver e entregar produtos com qualidade. Porém, conforme citado anteriormente, um software livre de erros é uma utopia. Neste sentido, a empresa, ciente disso, busca minimizar os impactos que estes erros podem causar se encontrados pelos clientes.

5.6. RETORNO SOBRE INVESTIMENTO (ROI)

Em tecnologia, o *Return On Investment* (ROI) segundo LEAL (2002), "é o benefício que se obtém por cada unidade investida em tecnologia durante certo período de tempo e costuma utilizar-se para analisar a viabilidade de um projeto". O cálculo do ROI é representado pela seguinte fórmula:

ROI = (Ganho obtido – Investimento) / Investimento

> "Considera-se ROI como sendo a razão entre os valores economizados (em relação à abordagem sem testes) e o valor dos investimentos em cada uma das opções de testes avaliadas neste caso". (VIEGAS, 2015).

Na Figura 19 – Resultados do ROI - como exemplo para o ROI, num cenário hipotético, utilizou-se como comparativo de dados: entregas sem teste, testes internos e testes automatizados.

Tomando como premissa para análise deste cenário, assumindo que o defeito na fase de desenvolvimento tenha o custo de R$ 1,00, na fase de testes tenha o custo de R$ 10,00 e que os mesmos defeitos, porém encontrados na fase de produção, custem R$ 100,00, temos os seguintes números:

	Tipos de testes		
	Sem testes	Testes internos	Testes automatizados
Testes			
Pessoal: Salário de 1 analista (R$)	-	5.000,00	-
Gestão: Salário de 1 gerente de testes (R$)	-	15.000,00	-
Investimento em Automação de Testes	-	-	30.000,00
Total do investimento (R$)	-	**20.000,00**	**30.000,00**
Desenvolvimento			
Defeitos encontrados	100	100	100
Custo dos defeitos (R$)	100,00	100,00	100,00
Testes			
Defeitos encontrados	-	400	850
Custo dos defeitos (R$)	-	4.000,00	8.500,00
Produção			
Defeitos encontrados	900	500	50
Custo dos defeitos (R$)	90.000,00	50.000,00	5.000,00
Totalização			
Investimentos (R$)	-	20.000,00	30.000,00
Custo dos defeitos (R$)	90.100,00	54.100,00	13.500,00
Total de custos (R$)	**90.100,00**	**74.100,00**	**43.600,00**
ROI (em relação ao processo sem testes)	n/a	80%	155%
Economia (em relação ao processo sem testes)	n/a	16.000,00 17,75%	46.500,00 51,60%

Figura 8 - Resultados do ROI
Fonte: adaptado de VIEGAS, 2015.

Baseando-se nos dados referentes ao ROI, pode-se ilustrar através do gráfico da Figura 20 – Economia com teste de software - os ganhos ou a economia, na realização de testes, gerando reduções importantes dos custos no processo de desenvolvimento.

Neste sentido, destaca-se também a importância do investimento em Testes Automatizados. O retorno pode ser medido através da diminuição dos defeitos e, consequentemente, do retrabalho, além da economia que pode proporcionar à empresa o investimento em outras frentes, como Inovação por exemplo. Por isso, a adoção deste tipo de tecnologia vem se tornando cada vez mais presente nas empresas, inclusive na empresa pesquisada.

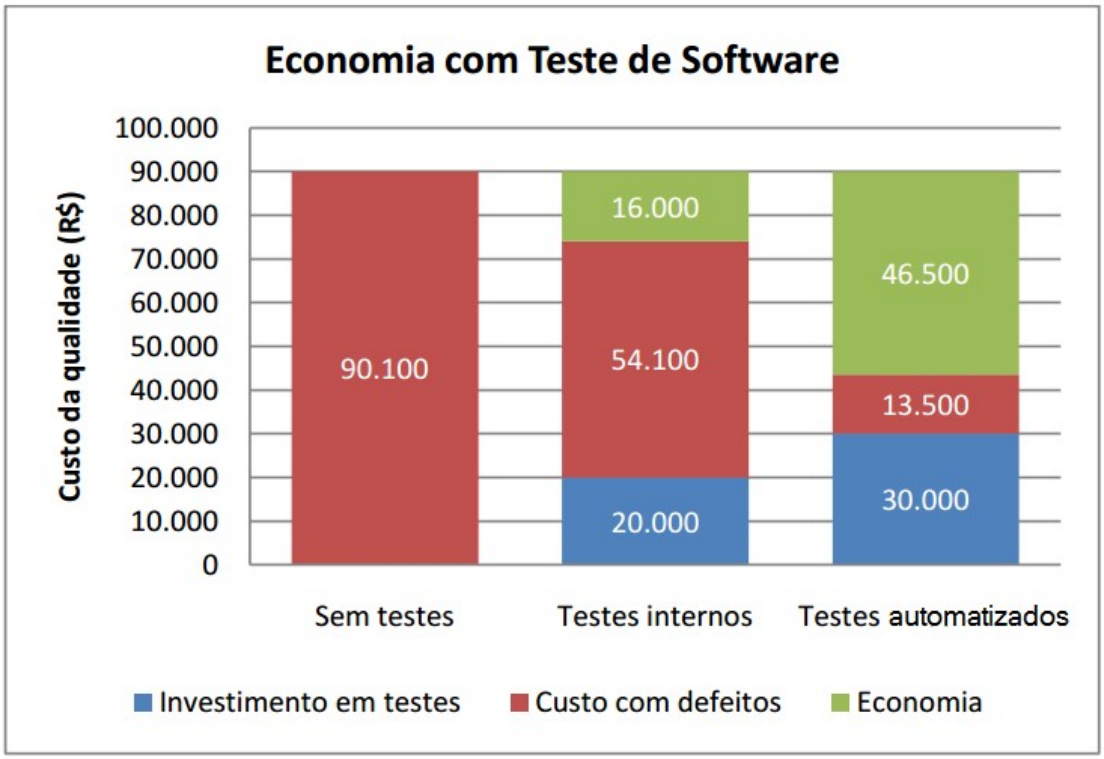

Figura 20 - Economia com teste de software
Fonte: adaptado de VIEGAS, 2015.

6. CONCLUSÃO

Destaca-se neste estudo a importância dos testes no ciclo de desenvolvimento, principalmente nos sistemas ERP, aprimorando assim a qualidade no software. Quanto mais se testa, mais confiável será o produto final. Com menos erros, os retrabalhos são minimizados nas etapas de desenvolvimento e, após a entrega do produto, reduz a insatisfação do cliente. Em contrapartida, uma das principais desvantagens dos testes refere-se ao custo. O teste de software é um processo caro, já que não é incomum uma companhia de desenvolvimento de software gastar entre 30% e 40% do total do esforço do projeto em teste.

O momento em que o erro é encontrado está diretamente ligado ao custo. Por exemplo, na etapa de especificação, o custo do erro é baixo, pois ainda está no início do projeto. Já na etapa de produção, o custo é alto, pois além de todos os custos anteriores, implicam os custos de atendimento ao cliente, replicação dos defeitos pela equipe de testes, e o *recall*, ou seja, a correção do defeito pela equipe de desenvolvimento.

Neste caso, dizemos que o software foi “subtestado”. Por outro lado, se um software for “supertestado”, o custo para tal torna-se desnecessário.

Mesmo com todos os levantamentos, métodos e conceitos, nenhum sistema está livre de erros ou defeitos. Porém, utilizando vários deles combinados, os efeitos podem ser minimizados.

Os clientes estão mais exigentes e o mercado mais concorrido. Por conta disso, quando um erro chega ao cliente, como impactos negativos podemos citar:

- Aumento do custo do projeto por causa da correção do erro já em produção;

- Insatisfação do cliente por conta dos problemas que este erro pode causar ao seu processo;

- Em muitos casos, perda da confiança por parte do cliente com relação ao produto e até mesmo futuros projetos.

7. CONSIDERAÇÕES FINAIS

No desenvolvimento de um sistema ERP, seja na fase de construção ou nas etapas de inovação e manutenção, a etapa de testes ocupa papel importante, sendo responsável por grande parte do esforço total de cada projeto.

Em função disso, as metodologias, documentos e ferramentas disponíveis são inúmeras, porém, cabe à empresa e principalmente aos gestores das áreas de TI, Desenvolvimento e/ou Qualidade identificar quais destes são mais bem aplicados ao seu dia-a-dia e ao seu produto.

Por outro lado, diversas novas formas e ferramentas surgem no mercado a fim de ampliar, melhorar e dinamizar os testes. Um exemplo disso é a Automação de Testes.

7.1. LIMITAÇÕES DA PESQUISA

Este estudo apresentou limitações quanto ao universo pesquisado. O estudo de caso foi realizado em apenas uma empresa de desenvolvimento de software ERP.

Por se tratar de uma pesquisa qualitativa, o foco foi em demonstrar como as técnicas, metodologias e ferramentas podem auxiliar na busca da qualidade do produto “software” nesta empresa estudada.

A limitação também se deveu à questão de confidencialidade e concorrência, já que dependendo dos tipos e métodos utilizados, podem se tornar um diferencial no mercado.

7.2. CONTRIBUIÇÕES ACADÊMICAS PARA ESTUDOS FUTUROS

Ao longo do desenvolvimento deste estudo foram identificadas algumas questões que permitem ampliar o entendimento do assunto estudado.

Entre eles podemos citar o estudo quantitativo, já citado no item anterior, realizando a comparação entre diversas empresas que desenvolvem software ERP, já que esta pesquisa foi realizada com uma amostra de dados retirada de uma única

empresa. Sugere-se então, como trabalho futuro, que a pesquisa seja aplicada em diferentes empresas, aumentando e diversificando a amostra de dados, tendo assim uma pesquisa geral sobre os processos de testes existentes e adotados no mercado atualmente.

Além disso, através das pesquisas realizadas e até mesmo do estudo de caso, percebeu-se uma crescente na utilização de ferramentas de Automatização de testes. Estudos específicos sobre este assunto, inclusive equiparando-os com os testes realizados em sistemas ERP e em outros sistemas, seriam de grande contribuição e complementaria o presente trabalho.

Outro trabalho futuro bastante relevante poderia focar nas ferramentas e metodologias ágeis de testes aplicadas já nas etapas de especificação e codificação, como supracitadas neste estudo, por exemplo, o TDD e o BDD.

8. REFERÊNCIAS

ALMEIDA, C., Introdução ao Teste de Software. 2010. Disponível em: http://web.archive.org/web/20170506140652/http://www.linhadecodigo.com.br/artigo/2775/introducao-ao-teste-de-software.aspx. Acesso em 14 de maio de 2016.

BASTOS, A., RIOS, E., CRISTALLI, R., MOREIRA, T., Base de conhecimento em teste de software. 2.ed. SÃO PAULO: Martins, 2007.

BATISTA, F. de M., O que é mesmo Qualidade? Blog da Qualidade. 2014. Disponível em: http://web.archive.org/web/20170506140818/http://www.blogdaqualidade.com.br/o-que-e-qualidade/. Acesso em 14 de abril de 2017.

CALDAS, L. Diferença entre TDD e BDD. 2014. Disponível em: http://web.archive.org/web/20170506140940/https://ciclosw.wordpress.com/2014/09/04/diferenca-entre-tdd-e-bdd/. Acesso em 15 de julho de 2016.

FARIAS, E. A arte do teste de software. 2014. Disponível em: http://web.archive.org/web/20170506141111/https://artedotestedesoftware.wordpress.com/. Acesso em 10 de julho de 2016.

FILHO, W. L. S., O que é melhor: Teste manual ou automatizado? 2015. Disponível em: http://web.archive.org/web/20170506135910/https://talkingabouttesting.com/2015/03/16/o-que-e-melhor-teste-manual-ou-automatizado/. Acesso em 10 de julho de 2016.

FUNPAR. Fundação de Apoio da Universidade Federal do Paraná. O ciclo de vida do teste. 2016. Disponível em: http://web.archive.org/web/20170509000810/http://www.funpar.ufpr.br:8080/rup/process/workflow/test/co_lifet.htm. Acesso em 15 de julho de 2016.

GOLDENBERG, M. A arte de pesquisar: como fazer pesquisa qualitativa em ciências sociais. Rio de Janeiro: Record, 1999.

GONÇALVES, M. R. Redução de custos de manutenção aplicando rotinas de testes. 2013. Disponível em: http://web.archive.org/web/20170511015835/http://monografias.brasilescola.uol.com.br/computacao/reducao-custos-manutencao-aplicando-rotinas-testes.htm. Acesso em 10 de maio de 2017.

HABERKORN, E., Gestão Empresarial com ERP. São Paulo: Microsiga, 2003.

HIRAMA, K., Engenharia de Software: qualidade e produtividade com tecnologia. Rio de Janeiro: Elsevier, 2011.

JUNIOR, H. E., Engenharia de Software na Prática. São Paulo: Novatec, 2010.

JUNIOR, W. Qualidade em Teste de Software. 2017. Disponível em: http://web.archive.org/web/20170509002710/https://qtsw.wordpress.com/testes-de-caixa-branca-preta-cinza-alpha-beta-hammm/. Acesso em 14 de abril de 2017.

LAGARES, V., ELIZA, R., Gestão de defeitos no teste de software. Revista Java Magazine 94. Rio de Janeiro: DevMedia. 2011.

LEAL, J. A usabilidade e o ROI (return on investiment). 2002. Lisboa, 2002.

LEITE, M. O que significa utilizar um sistema ERP customizável? 2014. Disponível em: http://web.archive.org/web/20170715135203/http://www.artsoftsistemas.com.br/blog/o-que-significa-utilizar-um-sistema-erp-customizavel. Acesso em 15 de julho de 2017.

LUIZ, R. R. V., Obtendo qualidade de software com o RUP. 2011. Disponível em: http://web.archive.org/web/20170509004258/http://javafree.uol.com.br/artigo/871455/. Acesso em 17 de abril de 2016.

MARINS, A., O que são Mapas Mentais? Comunidade ADM. 2009. Disponível em: http://web.archive.org/web/20170506141325/http://www.administradores.com.br/artigos/economia-e-financas/o-que-sao-mapas-mentais/28259/. Acesso em 14 de abril de 2017.

MYERS, G. J.,The Art of Software Testing, Second Edition. Word Association: 2004.

NETO, A. C. D., Artigo Engenharia de Software - Introdução a Teste de Software, Engenharia de Software Magazine, Edição Especial, Rio de Janeiro: DevMedia, 2009a.

NETO, A. C. D., Casos de Teste: Aprimore seus casos e procedimentos de teste, Engenharia de Software Magazine, Revista 67, Rio de Janeiro: DevMedia, 2009b.

NOGUEIRA, E. Data Driven com WebDriver. Blog Qualister. 2016. Disponível em: http://web.archive.org/web/20170506142131/http://www.qualister.com.br/blog/data-driven-com-webdriver. Acesso em 15 de julho de 2016.

OLIVEIRA, A. C. de., ETIERP – Estratégia de Testes para Implantação de Sistemas ERP. Universidade de Campinas – Mestre em Ciência de Computação, Dezembro, 2003.

PERES, P., Testes de Software – Uma Visão Geral. 2009. Disponível em: http://web.archive.org/web/20170506142229/https://pt.slideshare.net/pauloperes2009/testes-de-software-uma-viso-geral. Acesso em 10 de julho de 2016.

PRESSMAN, R. S.. Engenharia de Software. [S.l.]: McGraw Hill, 2002.

PORTALERP. Infográfico Mercado de ERP 2013. 2014. Disponível em: http://web.archive.org/web/20170506142338/http://portalerp.com/destaques/1299-infografico-mercado-de-erp-2013. Acesso em 17 de abril de 2016.

RAKITIN, S.R., Software Verification and Validation for Practitioners and Managers, Second Edition, Artech House Publishers, Norwood, USA, 2001.

RAPPS, S.; WEYUKER, E. J. Data flow analysis techniques for program test data selection. In: 6th International Conference on Software Engineering, Tokio, Japan, 1982.

ROCHA, A. R. C., MALDONADO, J. C., WEBER, K. C. , “Qualidade de software – Teoria e prática”, Prentice Hall, São Paulo, 2001.

SANTOS, L. R. dos, Técnicas de Testes de Software. FAES – UFPR. 2011. Disponível em: http://web.archive.org/web/20170506142543/http://www.inf.ufpr.br/lmperes/ci221/aula_tecnicas_teste_Luis_Renato.pdf. Acesso em 24 de abril de 2016.

SANTOS, V., O que é e como fazer “Revisão da Literatura” na pesquisa Teológica. Mackenzie. 2012. Disponível em: http://web.archive.org/web/20170506142648/http://mackenzie.br/fileadmin/Mantenedora/CPAJ/Fides_Reformata/17/17_1artigo6.pdf. Acesso em 21 de janeiro de 2017.

SILVA, A. R. da., Uma metodologia de testes em software para micro e pequenas empresas estruturada em multicritério. Universidade de Fortaleza. Pós Graduação em Informática Aplicada. 2011.

SILVA, C. E., Desenvolvimento de um sistema ERP com foco nas tecnologias de software livre / código aberto”. UNISUL. 2006. Disponível em: http://web.archive.org/web/20170506142747/https://projetos.inf.ufsc.br/arquivos_projetos/projeto_731/resumo_TCC.pdf. Acesso em 13 de março de 2016.

SOMMERVILLE, Engenharia de Software, 9ª edição. Pearson, 2011.

STEFANINI. Automação de Testes. 2013. Disponível em: http://web.archive.org/web/20170506142831/https://stefanini.com/br/2013/08/automacao-de-testes/. Acesso em 17 de abril de 2016.

TERAWARE. Sistemas ERP e suas características. 2014. Disponível em: http://web.archive.org/web/20170506142933/http://portal.teraware.com.br/sistemas-erp-e-suas-caracteristicas/. Acesso em 17 de abril de 2016.

TRAVASSOS, G. H., LIMA, G. M. P. S., NETO, A. C. D., Metodologia para Planejamento, Execução e Controle de Teste de Software. UFRJ. 2006. Disponível em: http://web.archive.org/web/20170506143213/http://webx.sefaz.al.gov.br/posengsoft/documentos/vvt/2-teste_gladys_travassos_planejamento.pdf. Acesso em 10 de julho de 2016.

VIEGAS, J. Teste de Software: Indrodução, Conceitos Básicos e Tipos de testes. OneDayTesting Blog. 2015. Disponível em:

http://web.archive.org/web/20170506143312/http://blog.onedaytesting.com.br/teste-de-software/. Acesso em 21 de janeiro de 2017.

YIN, R. K. Pesquisa Estudo de Caso – Desenho e Métodos (2 ed.). Porto Alegre: Bookman, 1994.

9. GLOSSÁRIO

Array: estrutura de dados que armazena uma coleção de elementos de tal forma que cada um dos elementos possa ser identificado por, pelo menos, um índice ou uma chave.

Bug: defeito, falha ou erro no código de um programa que provoca seu mau funcionamento.

Cloud Computing: em português, computação em nuvem, é a disponibilização da informação como um serviço ao invés de um produto, onde recursos compartilhados, software e outros são fornecidos, permitindo o acesso através de qualquer computador, tablet ou celular conectado à Internet.

CMMI: o *Capability Maturity Model Integration* (CMMI) é uma abordagem de melhoria de processos que fornece às organizações elementos essenciais de processos eficazes. Pode ser usado para guiar a melhoria de processo em um projeto, divisão ou em uma organização inteira.

Cucumber: framework para BDD.

Fonte: na informática, o código-fonte é o conjunto de linhas de texto que dá instruções ao computador para executar um software, isto é, indica-lhe o que tem de fazer e de que maneira.

Fortune: a Fortune é uma revista sobre negócios americana, fundada por Henry Luce em 1930. A Fortune 100 é a revista que lista as 100 Melhores empresas para se trabalhar.

Framework: um framework (ou biblioteca), em desenvolvimento de software, é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica. Um framework pode atingir uma funcionalidade específica, por configuração, durante a programação de uma aplicação.

Go-live: momento em que um produto ou sistema é colocado em produção no cliente ou se torna disponível ao público.

Issue: problema, assunto ou caso.

Layout: leiaute em português. Em computação, seria o plano sobre o qual será construído um conteúdo na internet (sites, blogs, páginas, *softwares* etc).

Link: uma palavra, texto ou imagem que quando é clicada pelo usuário, o encaminha para outra página na internet, que pode conter outros textos ou imagens.

Log: em computação, log de dados é uma expressão utilizada para descrever o processo de registro de eventos relevantes num sistema computacional.

Login: nome escolhido pelo usuário quando tem que fazer a autenticação para usar um determinado sistema ou serviço. O login é feito com o nome de usuário e com a senha que foi escolhida.

Pop-up: janela que abre, em separado, quando se acessa uma página na web ou algum link de redirecionamento.

Print Screen: tecla de atalho. No Windows®, captura em forma de imagem tudo ou parte do que está presente na tela e copia para a Área de Transferência.

Quality Assurance: o QA define-se como um conjunto de atividades para garantir a qualidade nos processos de desenvolvimento.

Release: liberação pública de uma nova versão de um programa ao qual foram adicionadas correções e melhorias.

RSpec: é uma estrutura de teste de unidade para a linguagem de programação Ruby.

Scorecard: BCS - Balanced Scorecard é uma ferramenta de planejamento estratégico na qual a entidade tem claramente definidas as suas metas e estratégias, visando medir o desempenho empresarial através de indicadores quantificáveis e verificáveis.

Suite: uma suite de software ou conjunto de aplicativos é uma coleção de programas de computador, geralmente software aplicativo ou software de programação, de funcionalidade relacionada, muitas vezes compartilhando uma interface de usuário mais ou menos comum e alguma habilidade para trocar dados com facilidade uns com os outros.

Template: template (ou "modelo de documento") é um documento de conteúdo, com apenas a apresentação visual e instruções sobre onde e qual tipo de conteúdo deve entrar a cada parcela da apresentação, por exemplo, conteúdos que podem aparecer no início e conteúdos que só podem aparecer no final.

Update: um termo inglês e significa atualizar ou modernizar. Esta palavra costuma ser utilizada para se referir às versões mais recentes de determinada tecnologia, como softwares, jogos eletrônicos, etc.

WebDriver: também conhecida como Selenium, o WebDriver é um conjunto de bibliotecas (bindings) para as linguagens de programação suportadas.

Workflow: é um termo inglês que significa “fluxo de trabalho”, na tradução para a língua portuguesa. O conceito do workflow é de uma sequência de passos necessários para automatizar processos, de acordo com um conjunto de regras definidas, permitindo que estes possam ser transmitidos de uma pessoa para outra.

ANEXO

ANEXO A – QUADROS DE DEFEITOS

Quadro 5 – Defeitos de Usabilidade

Detalhamento Tipos de Defeito	Quando usar
Erro em tela	Tela fora do padrão (Formatação de cores, estilo, tipo e tamanho de fonte).
	Textos dos labels (botões, campos, informações), títulos, links, hints, mensagens etc. com erros de ortografia.
	Habilitação/desabilitação dos objetos da tela (botões, campos, links, ícones) incorretos.
	Telas não está intuitiva.
	Texto explicativo (HINT) dos botões e campos incorretos.

Fonte: documento da empresa pesquisada

Quadro 6 - Defeitos de Documentação

Detalhamento Tipos de Defeito	Quando usar
Help Online e assistente	*Help on-line* e contextual incompleta ou incorreta.
	Funcionalidade não possui assistentes.
Material de treinamento	Material de treinamento incompleto ou incorreto.
Artefatos de inovação / manutenção	Documentos do projeto inexistente ou incompletos (especificação, plano projeto, plano testes, caso testes).
Release Notes	*Release Notes* inexistente/incompleto/não entendível/inadequados.
Etapa de documentação	Não passou pela etapa de documentação.
Template incorreto	Não utilização do *template* vigente.
Texto incompreensível	Falta de clareza e sequência lógica no texto.
Erros gramaticais e ortográficos	Acentos, pontuação, concordância.
Alteração do contexto original	
Nome do Boletim Técnico incorreto	Não conformidade entre o nome do Boletim com relação à funcionalidade descrita.
Nome do pdf incorreto	Nome do pdf fora do padrão da Eng. Corporativa
Ausência de pdf no pacote	

Detalhamento Tipos de Defeito	Quando usar
Falta de informação técnica	Boletim que não tenha as informações técnicas necessárias, tais como Sistemas Operacionais, Fontes e Tabelas utilizadas. 1. Da tabela do cabeçalho (versão, país, bancos de dados). 2. Da tabela das informações técnicas (tabelas utilizadas {sigla+descrição}, rotinas envolvidas, fontes, sistemas operacionais). 3. Tabela de Aplicabilidade, específica para o parceiro NG. 4. Falta de informação técnica para os “Procedimentos para implementação”, “Ajustes efetuados pelo compatibilizador”, “Procedimentos para configuração”. 5. Falta de informação técnica para os “Procedimentos para utilização” (indicação do módulo e/ou mudança deste, caminho da rotina no menu).
Falta de informação processual	Falta de informação do processo que envolve a funcionalidade descrita no Boletim.
Informações divergentes do dicionário de dados	Divergências de informações entre Boletim x Dicionário.
Informação confusa	Informação confusa, inconsistente (para quê, como, onde, por quê) ou falta de tópico necessário, disponibilizado no boletim padrão.
Documento incorreto anexado	Quando o documento anexado à tarefa não corresponde ao tipo da tarefa. Exemplo: trata-se de *Release Notes* e não de Boletim Técnico.
Boletim Técnico fora do TFS	Não utilização do Boletim Técnico do TFS.
Cabeçalho e rodapé do Boletim Técnico incorretos	Não conformidade no padrão de cabeçalho e rodapé.
Campos em branco no Boletim Técnico	Quando os campos das tabelas de atualização do Dicionário de Dados estiverem em branco e não forem removidas.
Links incorretos	*Links* incorretos no tópico que foi alterado e/ou criado.
Topic Id incorreto	*Topic Id* sem o nome da rotina na frente ou com caracteres especiais (hífen, ponto, ponto e vírgula, acentuação, cedilha, letras maiúsculas, espaço).
Dead link	Novos tópicos que não foram incluídos no TFS (*dead link*).
Arquivos não liberados	Arquivos ainda reservados (*check-out* ou *lock*) no TFS.
Geração de versões futuras	
Fora do padrão	Não realização do padrão auditoria no tópico do Help que foi alterado.
Ausência de Visio	Fluxos/caixas/gráficos novos não adicionados ao Visio correspondente à rotina/módulo em questão.
Ausência da Evidência de *Help / Release Notes*	Quando não houver alterações no *Help*, no *Release Notes* e/ou no Manual de Integração, deve haver uma Evidência informando isso.
Ausência de Evidência de Manual de Integração	

Fonte: documento da empresa pesquisada

Quadro 7 - Defeitos de Performance / Desempenho

Detalhamento Tipos de Defeito	Quando usar
Tempo de Resposta	Tempo de resposta da execução das transações acima do estipulado.
Sobrecarga de *Hardware*/Recursos	Utilização dos recursos (rede, CPU, memória, disco, etc.) acima do esperado;
Excesso Usuários/Conexões	Sistema não suporta ao inserir uma carga superior ao espaço reservado no banco pela aplicação.
Stress (volume de dados)	Sistema não suporta ao efetuar transações com grande volume de dados.

Fonte: documento da empresa pesquisada

Quadro 8 - Defeitos de Segurança

Detalhamento Tipos de Defeito	Quando usar
Permissão de acesso	Usuários não autorizados acessando o sistema ou funcionalidade.
	Testar alguma ação que não seja permitida devido ao estado atual de uma entidade.
Confidencialidade	É a propriedade da informação pela que não estará disponível ou divulgada a indivíduos, entidades ou processos sem autorização. Em outras palavras, confidencialidade é a garantia do resguardo das informações dadas pessoalmente em confiança e proteção contra a sua revelação não autorizada.
Integridade	Ter a disponibilidade de informações confiáveis, corretas e dispostas em formato compatível com o de utilização, ou seja, informações íntegras, integridade é um dos itens que a caracteriza, e significa que a informação não foi alterada de forma não autorizada ou indevida. Se a informação é alterada de forma errada ou mesmo falsificada ela perde sua eficácia e confiabilidade, tornando vulneráveis decisões que a partir dela são tomadas, e tirando a credibilidade do ambiente (site ou empresa) que a forneceu.
Disponibilidade	Sistema informático resistente a falhas de hardware, software e energia, cujo objetivo é manter os serviços disponibilizados o máximo de tempo possível.
Autenticidade ou Facilidade de Manutenção	Propriedade que garante que a informação é proveniente da fonte anunciada e que não foi alvo de mutações ao longo de um processo. Entende-se por autenticidade a certeza de que um objeto (em análise) provém das fontes anunciadas e que não foi alvo de mutações ao longo de um processo.

Fonte: documento da empresa pesquisada

Quadro 9 - Defeitos de Ambiente

Detalhamento Tipos de Defeito	Quando usar
Inconsistência na estrutura de Dados	
Inconsistência de Repositório de Objetos	

Fonte: documento da empresa pesquisada

Quadro 10 - Defeitos de funcionalidade

Detalhamento Tipos de Defeito	Quando usar
Defeito não corrigido/requisito não implementado	O sistema não está contemplando os requisitos especificados. (Não está contemplando os requisitos especificados para inovação\manutenção, ou seja, o problema reportado pelo cliente ou requisito não foi resolvido. Quando o programa considerado da mesma rotina for alterado somente um deles. Exemplo: On-line e API.).
Defeito pré-existente	Novo problema encontrado. (Problemas já existente no produto (já expedido) e que impacta a conclusão da rotina).
Defeito regra de negócio	Irá ocorrer quando o analista define a regra de negócio incorretamente ou incompleto. Quando a orientação do analista não está de acordo com todos os problemas/requisitos reportados. Será de acordo com o que o analista colocou no Texto de Orientação do chamado / DIS. Este parecer deve contar negativo para o analista. (Análise de impacto incompleta).
Defeito Colateral	Ocorre quando programador na tentativa de corrigir o problema do cliente ou requisito, afeta outro ponto do sistema, inserindo um novo problema que não ocorria antes.
Procedimento Incorreto	Não foi anexado nas evidências o *log* de compilação ou *code-check*. Falta de *patch*, xml, conversor. (Quantidade errada de argumentos na função, falta/erro de xml ou problema em conversores impedem o funcionamento).
Menu	Não conformidades relacionadas a menu (procedimento, texto orientação, etc.).

Fonte: documento da empresa pesquisada

Quadro 11 - Defeitos de Codificação

Detalhamento Tipos de Defeito	Quando usar
Qualidade de código fonte	Programação não conforme (Lógica de programação, excesso de código, excesso de comentários nos fontes).
Cod3 - Falta réplica	Repasse para demais versões.
Erro de compilação	Erro de compilação nos objetos liberados.
Objeto sem *lock* no TFS.	Verificado paralelismo sem prévio acordo e necessidade de retorno para analise de impacto.

Fonte: documento da empresa pesquisada

Quadro 12 – Defeitos de Planejamento

Detalhamento Tipos de Defeito	Quando usar
Label	Falta de label ou se o label planejado for incorreto, gerando problemas para o cliente.
Réplica	

Fonte: documento da empresa pesquisada

APÊNDICE

APÊNDICE A – TERMO DE CONFIDENCIALIDADE

TERMO DE CONFIDENCIALIDADE

Pesquisa de dissertação do curso de Especialização em Gestão de TI

Dados do pesquisador
Nome: Robson Cérgoli
Instituição: Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Registro Acadêmico: 1580965
Orientador: Professor Ivan Francolin Martinez

Este termo tem como objetivo apresentar à “empresa parceira” um descritivo das ações que serão realizadas pelo pesquisador durante a sua pesquisa.

A observação se dará através de pesquisas em documentações, online ou não, entrevistas informais e análises de dados. Após a execução da pesquisa, o Pesquisador irá transcrever para o seu trabalho os resultados obtidos porém, antes da publicação e defesa final do trabalho ao Instituto Federal de Ciência, Educação e Tecnologia de São Paulo será entregue quantas cópias à empresa parceira solicitar para validar as informações que estão expostas no trabalho não ferem este termo de confidencialidade.

Além disso, o Pesquisador deixa claro que o objetivo desta pesquisa não é o de julgar a capacidade nem a qualidade do produto estudado, mas apenas descobrir se os responsáveis pelo processo de testes percebem o produto aderente às práticas de mercado. Esta pesquisa não gera nenhum vínculo empregatício e a empresa parceira não terá responsabilidade para pagamento de qualquer valor em espécie ao pesquisador.

Assim sendo, o pesquisador se compromete a:

- ser totalmente imparcial;
- não divulgar os nomes dos participantes envolvidos;
- não divulgar o nome da empresa envolvida neste trabalho;

O pesquisador ainda deixa claro que esta pesquisa não tem nenhum caráter comercial, sendo uma pesquisa de caráter meramente acadêmico.

Uma cópia deste termo, assinada em 15 de maio de 2017, encontra-se em poder do pesquisador.

São Paulo, _____ de _____ de 2017.

_____ _____

Robson Cérgoli Empresa parceira