Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

ANTÔNIO ARCANJO JÚNIOR

PROCESSO DE TESTE DE SOFTWARE: Uma descrição com a perspectiva da qualidade de software.

Belo Horizonte
2011

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Especialização em Informática: Ênfase: Análise de Sistemas

PROCESSO DE TESTE DE SOFTWARE: Uma descrição com a perspectiva da qualidade de software.

por

Antônio Arcanjo Júnior

Monografia de final de Curso

Prof. Ângelo de Moura Guimarães
Orientador

Belo Horizonte
2011

ANTÔNIO ARCANJO JUNIOR

PROCESSO DE TESTE DE SOFTWARE: Uma descrição com a perspectiva da qualidade de software.

Monografia apresentada ao Curso de Especialização em Informática do Departamento de Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Especialista em Informática.

Área de concentração: Analise de Sistemas.

Orientador: Prof. Ângelo de Moura Guimarães

Belo Horizonte
2011

ARCANJO JÚNIOR, Antônio
PROCESSO DE TESTE DE SOFTWARE: Uma descrição com a perspectiva da qualidade de software / Antônio Arcanjo Júnior. Belo Horizonte. 2011.
Xi, 36 f. : il.

Orientador: Prof. Ângelo de Moura Guimarães
Monografia – Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.

1. Qualidade de Software. 2. Teste de Software. 3. Metodologia de testes. I. ARCANJO JÚNIOR, Antônio. II. Universidade Federal de Minas Gerais. Departamento de Ciência da Computação. III. Título.

A Deus que sempre me abençoou e me orientou em todos os momentos de minha vida, a minha amada esposa que tanto me incentivou e compreendeu, aos meus pais que tanto me auxiliaram.

“Uma pessoa inteligente resolve um problema. Uma pessoa sábia evita-o”.

Einstein.

RESUMO

Este trabalho busca baseado nas sugestões da bibliografia da área, mostrar que encontrar erros e defeitos em um programa é a finalidade básica do PROCESSO DE TESTE DE *SOFTWARE*, tendo em vista a obtenção de um software de qualidade. É através dessas técnicas de pesquisa e expurgo de erros que se pode medir o nível de qualidade de um *software*. Testar não é mostrar que o *software* funciona, mas sim encontrar, de forma pré-definida e criteriosa, o maior número de erros possíveis, antes que o *software* chegue ao ambiente de produção. Para isso, devem-se definir objetivos de testes específicos para cada projeto de teste, tanto testes CAIXA BRANCA, em que se avalia o comportamento interno do componente, quanto de CAIXA PRETA, em que se avalia o funcionamento final no nível de interface. O trabalho destaca que todo esse processo deve ser realizado por uma equipe técnica qualificada, em um ambiente adequado e com ferramentas de automação. Tais processos devem, ainda, ser realizados paralelamente ao desenvolvimento do projeto. Dessa forma, o processo se torna mais seguro e o resultado com maior qualidade fica mais fácil de ser conseguido. A proposta desta monografia, portanto, é apresentar considerações gerais sobre teste no contexto da qualidade de software e mostrar os benefícios da implantação do processo de teste de software na empresa Cinco Sistemas LTDA para se alcançar um nível maior de qualidade de software e conseqüente aumento de sua confiabilidade.

Palavras-chaves: Teste, qualidade de sistema, *software.*

ABSTRACT

This work search based on the suggestions of the bibliography of the area show that finding errors and faults into a program is the basic finality of SOFTWARE TESTING PROCESS, with a view to obtaining a quality software.. It is through these research techniques and purge of errors that can measure the quality of software. Testing is not to show that software works, but finding, in a predefined and judicious way, the largest possible number of bugs, before software arrives at the production environment. This should be set objectives of specific tests for each test project, both WHITE BOX testing, which evaluates the internal behavior of the component, as the BLACK BOX, which evaluates the final run-level interface.. All of this process must be performed by a qualified team, in a suitable environment and with automation tools. Such processes must still be performed parallely to the project development. Thus, the process becomes safer and result in higher quality is more easily achieved. The purpose of this monograph, therefore, is to present general considerations on testing in the context of software quality and show the benefits of deploying the process of testing software in the enterprise Cinco Sistemas LTDA for achieving a higher level of software quality and the consequent increase in its reliability.

Key-words: Testing, System's quality, *software*.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IEEE *Institute of Electrical and Electronics Engineers*

RUP Rational Unified Process (Processo Unificado Racional)

XP *eXtreming Programming (Programação Extrema)*

FDD Feature Driven Development (Desenvolvimento Guiado por Funcionalidades)

SUMARIO

1. INTRODUÇÃO

A qualidade de um *software* está vinculada a todas as fases do desenvolvimento, porém para garantir que ele tenha sucesso durante seu ciclo de vida, é vital que seja construído de forma a garantir esta qualidade. A garantia da qualidade do software está relacionada, de acordo com BASTOS et al. (2007, p.11), com "***o aprimoramento da atividade de testes***". Quando os processos de testes e de revisão são devidamente executados, há uma redução notória dos custos de manutenção, o que para os clientes eleva a credibilidade e a satisfação na utilização do produto.

Conforme explica BASTOS et al. (2007, p.11), "***os testes eram efetuados pelos próprios desenvolvedores de software, cobrindo aquilo que hoje chamamos de testes unitários e testes de integração.***" Os próprios desenvolvedores, após a codificação, testavam suas aplicações. A busca feita para identificar "erros" era bastante restrita, visto que procedimentos de testes eram ainda pouco utilizados, não permitindo que todos, ou pelo menos a grande maioria dos defeitos, fossem diagnosticados. Os *softwares*, então, eram liberados em sua versão final e disponibilizados aos clientes.

Diante dessa dificuldade, mudanças organizacionais importantes surgiram no ambiente das empresas. De acordo com BASTOS et al. (2007, p.13) "***testar um software não é uma tarefa simples, mas requer profundo conhecimento de técnicas não disponíveis para os desenvolvedores e usuários.***" Dessa forma, os produtos podem adquirir confiança, funcionalidade adequada e aumento de desempenho, características que, em seu conjunto, constroem um *SOFTWARE* COM GARANTIA DE QUALIDADE.

1.1 Empresa

O projeto foi realizado na empresa Cinco Sistemas LTDA, cuja principal finalidade é desenvolver sistemas para o setor contábil.

A empresa desenvolve softwares para escritórios de contabilidade, possuindo sistemas para as seguintes áreas:

a) Pessoal;

b) Contábil;

c) Fiscal.

A Cinco Sistemas LTDA possui dois analistas e dois programadores, sendo um deles responsável pelo sistema da área Pessoal e o outro pelos sistemas das áreas Contábil e Fiscal. A empresa não possui uma metodologia de desenvolvimento de sistemas definida, nem um padrão de desenvolvimento. O analista define a proposta do que será realizado, que seria a definição, em um editor de textos e passa esta definição para o programador fazer. Após a realização da tarefa o programador apresenta o resultado para o Analista que revisa e implanta o código no sistema.

1.2 Objetivos dos trabalhos

O objetivo principal deste trabalho foi aplicar, de forma prospectiva, parte de uma metodologia para auxiliar o controle de testes funcionais de software na empresa Cinco Sistemas LTDA.

Os objetivos específicos do trabalho são:

a) Iniciar a implantação de um processo de testes e verificações de erros;

b) Analisar o resultado e avaliar os impactos da utilização de um processo padronizado na empresa.

c) Estudar as condições para implementar a melhoria na qualidade final do software produzido.

2. TESTES DE SOFTWARE

2.1 Conceitos

O TESTE DE *SOFTWARE*, bem executado, não deve ser realizado na intenção de provar que o *software* funciona. A finalidade básica do teste é analisar um programa visando descobrir erros e defeitos, tornando-se, assim, uma atividade investigativa, que envolve ações por todo o ciclo de desenvolvimento, a fim de descobrir seus problemas. Conforme explica NEVES, "***quanto mais tarde os erros forem descobertos, maior impacto causarão***". Erros não identificados durante o desenvolvimento levam a defeitos que podem ser produzidos pelos desenvolvedores e que só serão identificados após a liberação do produto final, normalmente encontrados pelos usuários.

Os testes devem assumir o papel de simular as operações com características mais próximas possíveis do seu uso, ainda em produção, com o objetivo de analisar os mesmos problemas que poderão ser enfrentados pelos usuários, sem o aparecimento de defeitos.

Testar, em seu conceito básico, não é mostrar que o *software* funciona. Técnicas são utilizadas para analisar todas as possibilidades que podem causar um mau funcionamento e os testes devem ser capazes, conforme explica BASTOS et al. (2007, p 31) "***de encontrar o maior número possível de defeitos no software***".

É necessário observar também o momento de interrupção dos testes. Avançar nos prazos pré-estabelecidos com o cliente não devem ser considerados para não interromper os testes. Todo prazo deve ser respeitado em seu limite acordado. Por outro lado, o exagero no tempo planejado para incluir os testes pode colocar em risco o prazo de entrega do *software*. É preciso avaliar o momento exato da paralisação dos testes. Ele deve ocorrer a partir de um grau pré-definido de confiança no software, alem de ser considerado realisticamente no planejamento do tempo necessário para conduzi-los.

2.2 Objetivos dos testes

É importante definir exatamente o objetivo a ser alcançado através dos testes. Ele pode estar diretamente ligado à localização do maior número possível de defeitos no *software*, diminuindo os riscos para o negócio e garantindo que as

necessidades dos clientes serão atendidas. O objetivo pode também estar ligado ao projeto de teste, que pode ser definido por meio dos requisitos de testes criados a partir dos requisitos do negócio (requisitos funcionais). Nesse caso, o objetivo pode ser também o de se alcançar e garantir um tempo de resposta adequado às necessidades do negócio (requisitos não funcionais).

É possível, então, definir objetivos específicos para cada projeto de teste.

Como os testes podem ser variados, é importante concentrar-se nos objetivos e no resultado final a ser analisado, ou como diz CHINA "***para medir a qualidade do sistema, deve-se averiguar se tudo está funcionando de acordo com as especificações passadas à empresa contratada.***"

2.3 Classificações dos Testes

Os principais tipos de testes com processos, segundo MACORATTI, podem ser definidos em: CAIXA BRANCA e CAIXA PRETA.

Quadro 1: Diferença entre os testes CAIXA BRANCA e CAIXA PRETA.

Tipo de teste	Teste caixa branca	Teste caixa preta
visibilidade do testador	Tem visibilidade do código e escreve casos de teste baseado no código	não têm visibilidade do código e escreve os casos de teste baseado em possíveis entradas e saídas e das funcionalidades documentado nas especificações e / ou requisitos
O que revela um caso de teste que falha?	Um problema (falha)	Um sintoma de um problema (falha)
Controlado?	Sim - O caso de teste ajuda a identificar as linhas específicas do código envolvido	Não - pode ser difícil encontrar a causa do fracasso no teste

Fonte: http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf (Tradução do autor)

A CAIXA BRANCA é uma técnica de teste que avalia o comportamento interno do componente de *software*. Trabalha diretamente sobre o código fonte.

Responsável por conferir a codificação elaborada pelo desenvolvedor, essa técnica é utilizada para avaliar aspectos relacionados aos testes de condição, teste de fluxo de dados, testes de ciclos, e basicamente o caminho lógico da estrutura.

Para se realizar este tipo de teste, podem utilizar diversas ferramentas, como no caso do Java, o **JTest**[1]. Alem de outros tipos de teste, ela permite a execução do teste caixa branca (static test), calculando métricas, tais como herança de profundidade, falta de coesão, complexidade ciclomática, blocos aninhados profundidade, número de filhos.

A CAIXA PRETA são testes conduzidos na interface do *software*, sem preocupação com a estrutura lógica interna do mesmo. Não se considera o comportamento do código-fonte para a obtenção do resultado final. Os aspectos a serem abordados são os dados de entrada e o processamento. O resultado obtido é comparado a um resultado esperado, previamente conhecido.

As categorias que o teste de Caixa Preta envolve são: as funções incorretas ou ausentes, erros de interface, estrutura de dados, acesso a base externa e desempenho do *software*.

Dentro de uma estrutura de programação, é possível utilizar uma mesma classe ou métodos para executar tarefas diferentes. Para uma mesma funcionalidade podem existir vários caminhos de execução e esses caminhos dependem da forma que o usuário opera o sistema e dos dados que serão informados.

De acordo com DIAS NETO, "***é recomendada para os níveis de TESTES DE UNIDADE e TESTE DE INTEGRAÇÃO, cuja responsabilidade principal fica a cargo dos desenvolvedores do software***", como será visto adiante.

2.4 Desenvolvimento de componentes de software

Um componente individual é um módulo que encapsula um conjunto de funções relacionadas (ou dados). Todos os processos do sistema são colocadas em componentes separados, para que todos os dados e funções dentro de cada componente sejam semanticamente relacionados (assim como com o conteúdo de

[1] Disponível em http://www.parasoft.com/jsp/resources/static_analysis

classes). Devido a este princípio, muitas vezes é dito que os componentes são modulares e coesos (Cox, 1991).

A utilização de componentes de software é essencial para a adoção de diversas estratégias de teste (seja caixa branca ou caixa preta). Isto porque ao focar em funcionalidades bem definidas e singulares, o teste pode se concentrar em poucas variáveis, facilitando a criação de casos de teste para os testes de unidade

Este enfoque exige, por outro lado, um esforço maior nos testes de integração.

3 INTRODUÇÃO AO PROCESSO DE TESTE

Muitas organizações têm boas intenções com relação aos testes que serão executados, porém, na maioria das vezes, os testes são considerados apenas na fase final, realizados por desenvolvedores ou analistas de sistemas após a conclusão de seu desenvolvimento e são vistos como se fizessem parte apenas da etapa final. Assim, não fornecem garantia de que o *software* foi testado adequadamente.

Em alguns casos, há um conflito com os prazos já definidos e com pressões vindas dos clientes, gerando uma possível eliminação da fase de testes. A verificação final acontece apenas, como afirma BASTOS et al. (2007 p.17) "***para garantir que as especificações ou os requisitos do negócio foram de fato implementados***". Esse processo causa um resultado insatisfatório e afeta a qualidade do produto.

Para a realização dos testes, conforme BASTOS et al. (2007, p.18), "***é necessário que os testes sejam executados por profissionais capacitados, usando metodologia apropriada.***"

Os testes devem seguir seu próprio processo, independente do processo de desenvolvimento, porém devem estar completamente integrados. Dessa forma, os testes não serão mais considerados como uma etapa do desenvolvimento e podem ser iniciados paralelamente ao projeto.

As atividades de teste dependem da conclusão dos produtos gerados pela atividade do processo de desenvolvimento. Enquanto a equipe de desenvolvimento começa a planejar os requisitos e preparar a implementação, a equipe de testes inicia o planejamento e a estratégia de testes para assegurar a qualidade dos produtos (*software* x documentação) que serão entregues.

3.1 Objetivo do Processo de Teste

Conforme nos orienta TOZELLI, o processo de teste visa definir o **que será testado**, no que tange a FUNCIONALIDADE, USABILIDADE, PERFORMANCE, ACEITAÇÃO, CONFIABILIDADE, RECUPERAÇÃO e SEGURANÇA; **quando** os testes serão realizados, envolvendo os TESTES DE UNIDADE, INTEGRAÇÃO, SISTEMA, REGRESSÃO e ACEITAÇÃO e **como** os testes serão executados: quais

os métodos ou técnicas serão utilizados, CAIXA BRANCA (estrutural) ou CAIXA PRETA (funcional), por exemplo.

3.2 Verificações x Validação

Existem duas áreas relacionadas ao Processo de Teste:

A VERIFICAÇÃO, em que são realizadas inspeções/revisões nos produtos gerados pelas diversas etapas do processo de teste (documentos e códigos do sistema), sendo possível responder, conforme explica BASTOS et al. (2007, p. 30) "***se o sistema foi construído corretamente***". Está relacionada aos testes estáticos e garantem que o *software* programe corretamente uma função específica.

Possibilita encontrar defeitos nas fases iniciais do projeto, antes da versão disponível do *software*. Revisar documento de requisitos com o propósito de checar inconsistências.

A VALIDAÇÃO, na qual o sistema é avaliado de forma a atender aos requisitos do projeto, sendo possível responder, conforme explica BASTOS et al. (2007, p. 30) *"**se construímos o software correto**"*. Está relacionada aos testes dinâmicos e garante que o *software* construído corresponde aos requisitos do cliente. Os testes mais utilizados nessa área são testes relacionados à funcionalidade, como TESTE DE REGRESSÃO, TESTE DE INTEGRAÇÃO e TESTE DE SISTEMAS.

Os testes de verificação e validação são complementares e devem ser executados durante todo o ciclo de vida do processo de testes. Vistos como independentes são responsáveis por fortalecer o processo de detecção de erros, o que aumenta a garantia do bom funcionamento do produto, e consequentemente, a eficiência do *software*.

3.3 Ciclos de vida do processo de teste

De acordo com RIOS e MOREIRA (2003), o ciclo de vida do Processo de Teste envolve várias etapas:

- Procedimentos Iniciais: estuda os requisitos de negócio de origem.
- Planejamento: elabora a estratégia de teste e do plano de teste, prepara a análise de risco do projeto.

- Especificação: elabora/revisa os casos e roteiros de testes.
- Execução: executa os testes planejados.
- Entrega: arquiva documentos de teste.
- Preparação: determina como deve ser preparado o ambiente de teste.

São elaborados, para cada uma dessas etapas, as atividades, os produtos e documentos. Também deve ser recolhida toda a documentação de teste preparada e confeccionado um relatório gerencial com as conformidades e não-conformidades encontradas.

A Norma IEEE[2] 829-1998, contempla grande parte da documentação e conceitos utilizados no processo de teste. Descreve um conjunto de documentos para as atividades de testes de um produto, sendo divida em três partes: PREPARAÇÃO DO TESTE, EXECUÇÃO e REGISTRO.

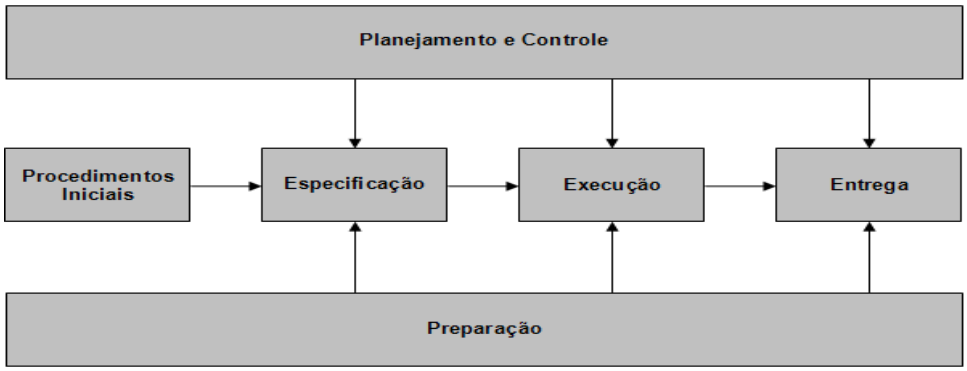

FIGURA 1: Modelo do ciclo de vida do Processo de Teste
FONTE: RIOS, E. & MOREIRA, T. Teste de Software. Rio de Janeiro, Alta Books, 2003.

[2] *Instituto de Engenheiros Elétricos e Eletrônicos - tradução nossa*

4. ESTRATÉGIAS DE TESTE

Uma estratégia de testes de *software* descreve a abordagem geral e os objetivos das atividades a serem executadas. Conforme explica BASTOS et al. (2007, p. 148) "***a Estratégia de Teste é um documento usado por algumas empresas para deixar o Plano de Teste mais leve***".

A estratégia de testes de *softwares* também deve descrever com clareza os critérios para a conclusão dos testes e os critérios de sucesso a serem usados, além de definir as ferramentas e técnicas a serem adotadas. A utilização desses critérios pode permitir que o *software* evolua para o teste de aceitação quando 95% dos casos de teste tiverem sido executados com êxito.

Trata-se da definição da fase do desenvolvimento em que o teste será aplicado.

É primordial que as dimensões dos testes sejam bem delimitadas durante a definição da estratégia de teste.

- Tipos de Testes => O que testar?
- Técnicas de Testes (Estrutural e Funcional) => Como Testar?
- Estágios ou Níveis de Teste => Quando Testar?

Como proposta para desenvolver uma boa estratégia de teste alguns passos das atividades podem ser executados:

- Os requisitos do sistema devem ser avaliados de forma a garantir que todos sejam cumpridos de acordo com as expectativas do cliente.
- Analisar os riscos para o negócio, onde os defeitos devem ser mensurados de acordo com os módulos mais utilizados e críticos do sistema, evitando prejuízos para o cliente.
- Avaliar e estabelecer prioridades para os testes.
- Definir os tipos e as técnicas a ser utilizado antes de os testes serem iniciados.

Não seria adequado se os testes fossem executados sem a existência de técnicas. Muito provavelmente, grande parte deles seria deficiente, desorganizada e sem propósitos bem definidos.

Segue abaixo, as Técnicas de Teste (Estrutural e Funcional) que podem ser adotadas segundo BASTOS et al. (2007 pgs. 142 – 144):

➔ **Técnica Estrutural** (Produto estruturalmente sólido):

- Testes de Estresse: submetem o sistema a condições extremas de execução, com baixo recurso de *hardware,* altos volumes de dados e transações.
- Testes de Execução: utilizado para verificar os tempos de resposta, os tempos de processamento e os tempos de desempenho.
- Testes de Recuperação (Contingência): força o sistema a falhar de diversas maneiras e verifica se sua recuperação é executada adequadamente.
- Testes de Operação: o sistema deve ser integrado com o ambiente operacional
- Testes de Conformidade: verifica se a aplicação foi desenvolvida seguindo os padrões, procedimentos e guias da área de processos.
- Testes de Segurança: confere se os acessos ao sistema estão em conformidade com o que foi estabelecido, garante a segurança do sistema e de seus módulos.

➔ **Técnica Funcional** (Todos os requisitos e especificações foram atendidos):

- Testes de Requisitos: verifica se o sistema está executando corretamente as funcionalidades, sem a presença de falhas, por um período pré-determinado.
- Testes de Tratamento de Erros: verifica se o sistema é capaz de responder adequadamente a transações incorretas.
- Testes de Interconexão: determina se a comunicação de dados entre os *softwares* está apropriada e de acordo com o esperado.
- Testes Paralelos: são utilizados para conferir se as funcionalidades existentes no sistema antigo estão consistentes e definidas corretamente no novo sistema.

5. AMBIENTE DE TESTE

O ambiente e a estratégia de teste têm uma relação direta. Eles determinam toda a estrutura onde o teste será executado e abrange a configuração do *hardware*, massa de dados, modelos de dados – incluindo o conjunto de cenários –, configuração do *software*, tipo e técnicas de testes que serão empregadas.

O ambiente de teste não está direcionado apenas na configuração dos computadores onde os *softwares* serão testados e sim em toda a estrutura envolvida como equipes, *hardware*s, *softwares* de gerenciamento, automação, treinamentos e outros.

Deve-se planejar o ambiente de forma a ficar o mais similar possível da realidade do usuário, para que possa garantir, conforme explica BASTOS et al. (2007, p. 83) "**a *descoberta de erros reais – ou seja, aqueles que realmente ocorreriam na produção e que porventura não foram descobertos em tempo de desenvolvimento***".

6. ANÁLISE DE RISCO

A análise de risco é tratada no escopo do projeto e pode apontar, de forma negativa, danos ao mesmo. Um risco não pode ser considerado nem como certeza de uma ocorrência, nem como um evento que não irá acontecer. O risco, propriamente dito, ocorre dentro de uma determinada faixa de probabilidade de concretização. Conforme nos explica BASTOS et al. (2007, pgs. 92 – 98), "***os riscos podem ser decorrentes da tecnologia da informação, relativos ao teste de software ou baseados nas características de qualidade***". Podem ser previstos e gerenciados antes de se tornarem problemas para a construção do sistema.

O impacto para o negócio do cliente, se o sistema apresentar defeitos, deve ser avaliado na análise de risco. É esse o maior objetivo dessa atividade.

Qualquer alteração em um módulo considerado crítico deve ser submetida a exaustivos testes. Um bom projeto de teste deve ser baseado nos riscos que um defeito pode oferecer ao negócio do cliente.

A seguir alguns riscos associados ao projeto de teste, conforme BASTOS et al. (2007, pgs. 95 – 96):

- Orçamento.
- Qualificação da equipe técnica de teste.
- Ambiente de teste.
- Ferramentas
- Metodologias
- Cronograma de recebimento de programas para teste e cronograma de devolução.
- Testware.
- Novas Metodologias.

É importante ressaltar que os riscos sofrem alterações à medida que as mudanças ocorrem no *software*.

7. PLANEJAMENTOS DOS TESTES DE SOFTWARE

O plano de testes é um documento que descreve o planejamento para a execução do teste. É o documento principal para que os gerentes de projeto de testes possam gerenciar e controlar os projetos de testes de *software*. Nesse documento são descritos equipes de execução, atividades, recursos, cronogramas, estratégias de testes, abordagem, e outros.

Para construir um plano de teste é necessário seguir quatro etapas:

- Selecionar a equipe de teste:
  não permitir que os próprios desenvolvedores executem essa atividade.
- Entender os riscos do projeto:
  realizar um entendimento real dos objetivos do projeto.
- Definir a estratégia:
  identificar todas as possibilidades de testes antes de iniciá-los.
- Elaborar o Plano de Teste:

Desenvolver roteiros de testes para apresentação dos casos de testes e como eles se relacionam.

Alguns desafios podem ser encontrados durante a elaboração do plano de teste: falta de envolvimento dos usuários, falta de entendimento dos clientes, falta de treinamento, insistência da gerência em manter a atividade de testes sob responsabilidade de desenvolvedores, falta de ferramentas, falta de apoio da gerência, prazo indevido para a execução dos testes, alterações emergentes sendo liberadas sem a realização dos testes, testadores sem condições adequadas para a realização de suas tarefas, e outros obstáculos. Porém, mesmo encontrando tais dificuldades, é fundamental a existência desse documento para um bom gerenciamento dos testes.

8. ELABORAÇÃO DOS TESTES

Conforme BASTOS et al. (2007 pg. 150) os documentos relacionados à fase de Elaboração dos testes são: especificação de Projeto de teste, especificação de caso de testes e especificação do procedimento de teste.

- **Especificação do projeto de teste** é um documento detalhado do Plano de Teste e identifica as funcionalidades e características a serem testadas pelo projeto.
- **Especificação de Caso de Teste** define os casos de teste, incluindo os dados de entrada, resultados e condições para a realização do mesmo.
- **Especificação do Procedimento de Teste** identifica a sequência de funções (roteiro) para usar o sistema e praticar os Casos de Teste especificados, de maneira a cobrir o Projeto de Teste planejado.

Os Casos de Teste servem de base para que os testadores possam executar os testes. É uma filosofia utilizada com a finalidade de testar os elementos do *software*. Podem ser elaborados para identificar defeitos nas estruturas internas e ainda garantir que os requisitos sejam plenamente atendidos. Devem ser escritos sempre com base na estratégia e no planejamento dos testes.

O analista de teste, responsável por descrever os casos de teste, deve pensar em todas as situações que podem provocar falhas, porque um teste bem sucedido é aquele que revela um erro ainda não descoberto.

Os Casos de Teste são originados dos requisitos do sistema, que desenvolvem uma especificação formal denominada **casos de uso**, em que os cenários são identificados e, em seguida, são elaborados os Casos de Teste.

Os Casos de Teste devem ser utilizados como centro motivador do teste.

9. TESTES REGRESSIVOS

Teste regressivo é a atividade de teste que consiste em reanalisar um programa, ou parte dele, depois que o mesmo sofre alterações, este procedimento visa garantir que as mudanças realizadas não trarão erros antigos ou novos. Os testes regressivos são realizados fazendo-se novas reexecuções de um conjunto de testes já existentes. Diante disto podemos concordar com CAETANO quando nos diz que "**Teste regressivo ou teste de regressão é o termo utilizado para o ciclo de re-teste de uma ou mais funcionalidades, a fim de identificar defeitos introduzidos por novas funcionalidades ou correção de defeitos.**"

Uma das ferramentas que nos auxiliam a realizar o Teste Regressivo é o JUnit[3], ele permite verificar se cada unidade de código esta funcionando de acordo com a forma esperada, facilita a criação, execução automática de testes e a apresentação dos resultados analisados. Através do Teste de Unidade com o JUnit, conseguimos obter os seguintes benefícios no desenvolvimento do software:

- Prevenção de aparecimento de erros em decorrência de códigos mal escritos;
- Maior confiança no código testado;
- Testar situações de sucesso ou falha;

A figura abaixo apresenta a arquitetura das classes do JUnit.

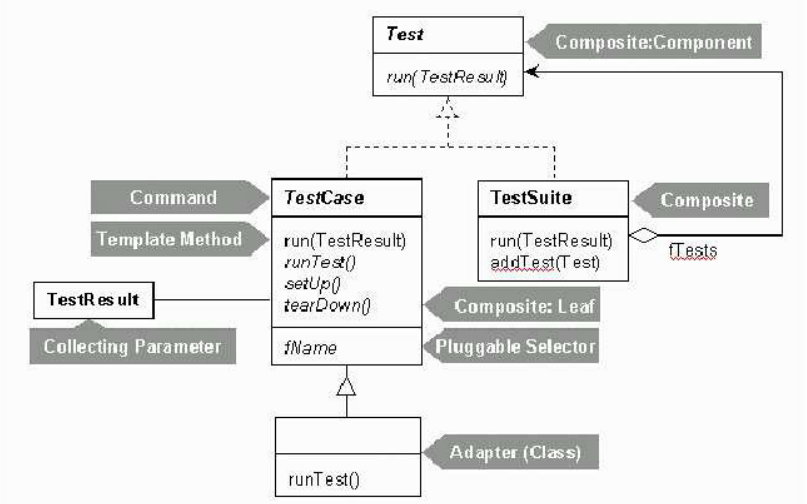

Figura 02: Arquitetura das Classes do JUnit

Fonte: Manual do JUnit (Cooks Tour)

[3] Ferramenta opensource de teste unitário em JAVA™ – disponível em < http://www.junit.org>

A implementação de testes de Unidade com o JUnit foi definida da seguinte forma:

a) Criação de uma classe que estenda a classe junit.framework.TestCase para cada classe a ser testada;

b) Para cada método testado foi definido um método public void test???() no TestCase.

As imagens abaixo apresentam uma classe para ser testada. Isso será feito através de chamadas aos seus métodos, utilizando uma outra classe Java no ambiente Eclipse.

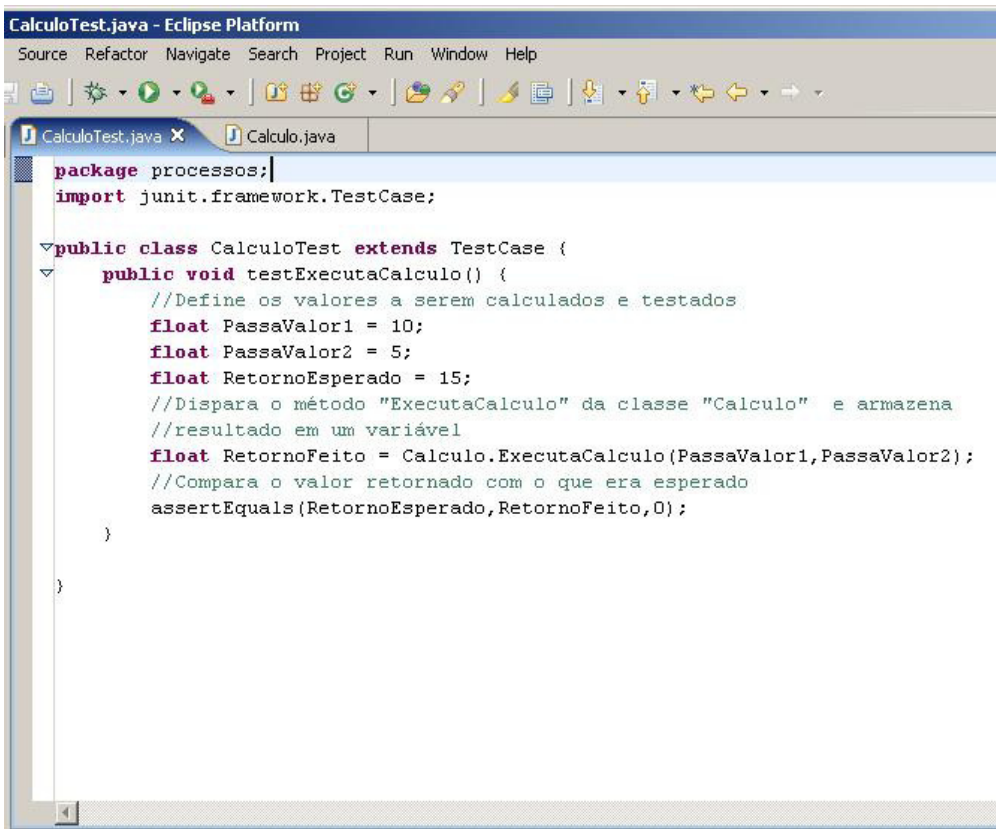

Figura 03: Classe de teste no Ambiente Eclipse

Criação da classe de teste:

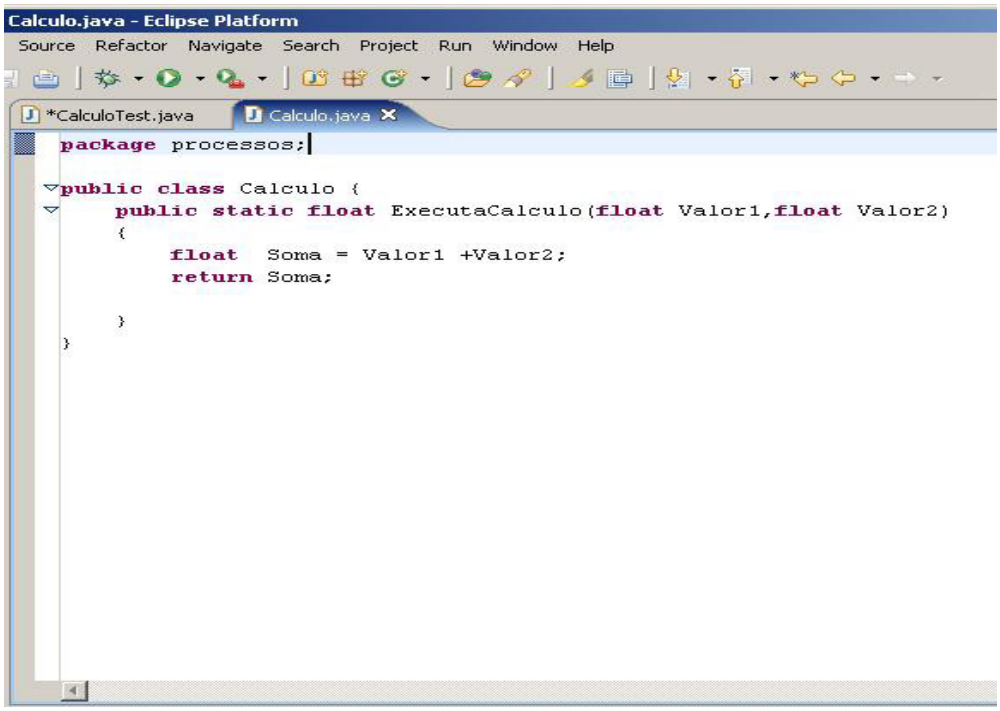

Figura 04: Classe de teste

Rodando o teste em modo gráfico

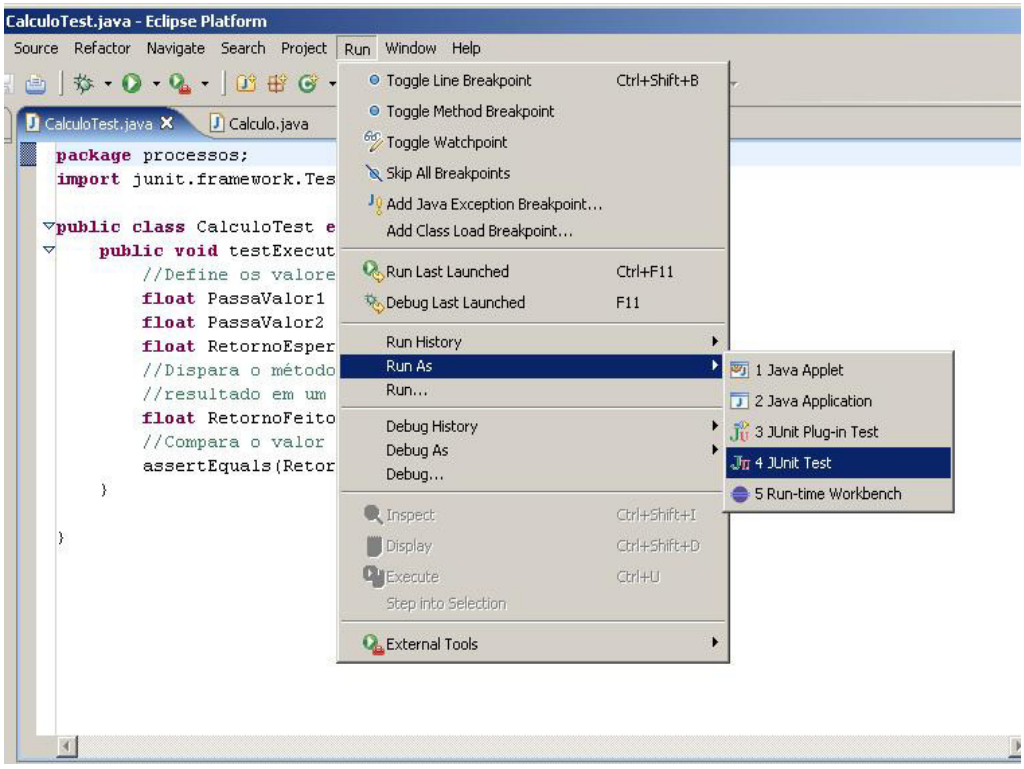

Figura 05: Classe de teste sendo executada através do Eclipse

Resultado em caso de Sucesso

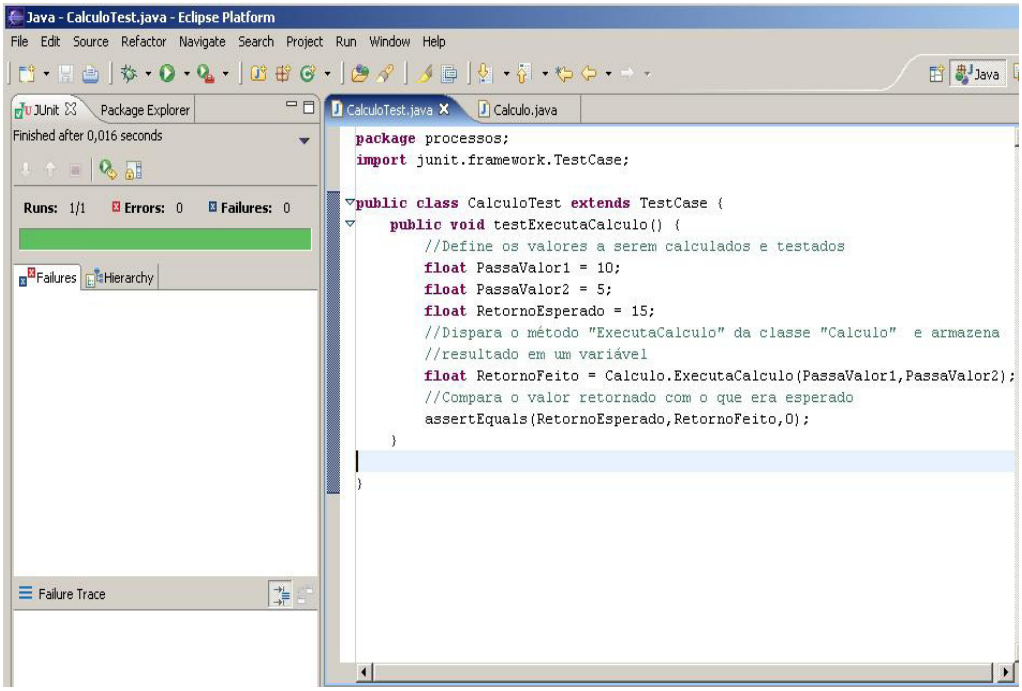

Figura 06: Eclipse informando que o teste foi um sucesso.

Resultado em caso de Falha

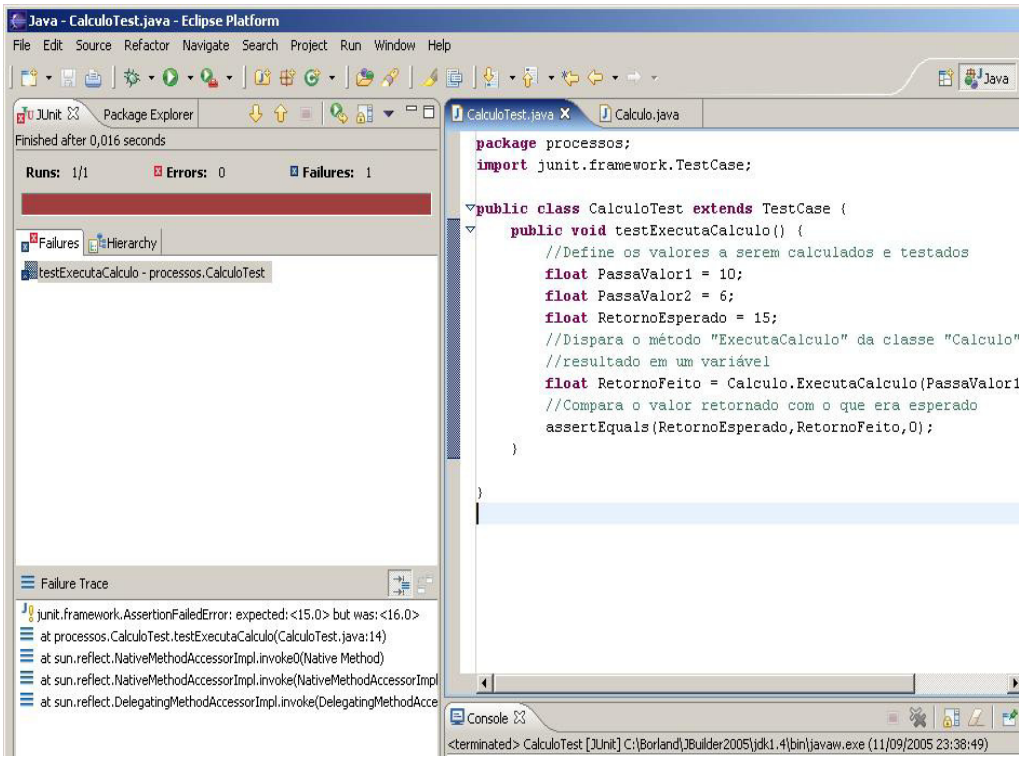

Figura 07: Eclipse informando que o teste não obteve sucesso.

Após a execução dos testes unitários, sendo obtido sucesso o Eclipse apresentará a view relacionada ao plugin JUnit com uma barra na cor verde, caso tenha obtido uma falha a barra será apresentada na cor vermelha. Com isto o analista conseguirá obter um controle maior do nível de testes de suas aplicações, assim como obter um histórico de testes a serem realizados a cada modificação sofrida pelos sistemas.

10. ESTUDO DE CASO: DESENVOLVIMENTO DE UMA PROPOSTA PROSPECTIVA DE MUDANÇA DO PROCESSO DE TESTE

10.1 Contexto

Os testes na empresa Cinco Sistemas LTDA são feitos pelos próprios programadores, que são verificações básicas de funcionamento do sistema, tais como resultados finais de cálculos previamente estabelecidos. Após terminar o desenvolvimento, o processo é liberado para a revisão do analista, que examina o código e libera a versão para o suporte técnico testar o funcionamento final do sistema e enviar para os clientes. Com o aumento do numero de clientes, da demanda de novas funcionalidades nos sistemas e da mudança freqüente da legislação vigente, o suporte não pode realizar um teste com mais critério, liberando versões que apresentaram erros já nos clientes, causando um grande problema para a empresa, que deveria liberar uma nova versão corretiva com mais urgência e desgastando sua imagem com o cliente.

Com a verificação deste problema e a necessidade de garantir uma melhor imagem da empresa com seus clientes, verificou-se que a empresa deveria possuir uma área ou uma pessoa especifica e especializada para se dedicar integralmente ao processo de teste dos sistemas, utilizando ferramentas que garantissem uma melhor qualidade dos testes e qualidade final dos sistemas.

10.2 Passos iniciais

A seguir são descritos os esforços no sentido de tentar uma modificação nos processos de teste da empresa, tendo em vista a melhoria de qualidade dos testes executados.

Foi elaborado, com este autor em conjunto com os analistas, o Sr. Mateus Barreto e a Sr. Graciela de Andrade Boschetti, uma pesquisa sobre algumas ferramentas de software livre que pudessem auxiliar no processo de teste. A ferramenta escolhida foi o JUNIT que facilita o desenvolvimento e a execução de testes de unidade, como visto no tópico 9.

A proposta se concentrou no uso da ferramenta JUnit como auxiliadora no desenvolvimento de técnicas de teste, ainda não utilizada pela empresa Cinco Sistemas LTDA.

Para o teste prospectivo foram seguidos os passos preconizados pelas metodologias, como visto nos tópicos. Em primeiro lugar foram projetadas e escritas as classes de teste e depois as classes com as regras de negócio de cada sistema.

Para se realizar o planejamento dos testes, foram definidos os seguintes processos que foram seguidos e realizados:

- Criação de lista de tarefas a serem desenvolvidas;
- Construção da classe de teste (test case) e criação do método de teste para uma tarefa da lista;
- Teste de falha da classe no JUnit;
- Construção do código da forma mais simples para realizar o teste;
- Remoção das duplicações de dados através de refatoração;
- Refinamento do teste existente, caso necessário, ou escrever novo teste;
- Executar todos os passos para cada uma das tarefas listadas.

Foram rodadas as classes de teste várias vezes ao dia para se corrigir os problemas o mais cedo possível, antes que o projeto termine e a propagação do erro se torne um problema maior e mais complexo.

A ferramenta foi testada em um sistema de Controle Interno da empresa. A finalidade deste sistema é controlar os dados de cada cliente, assim como todos os pedidos e contatos realizados com os mesmos, de forma que a empresa tenha um controle de todas as solicitações dos clientes e do andamento de cada pedido ou alteração solicitada. Por se tratar de um sistema interno, de uso restrito da empresa e de menor tamanho e complexidade, os analistas propuseram que este fosse o primeiro sistema a ser trabalhado com a nova proposta de desenvolvimento, assim o resultado obtido poderia ser percebido com maior rapidez, visto que os demais sistemas já estavam em processo avançado de desenvolvimento e uso contínuo por seus clientes.

O sistema foi criado então com base na nova proposta definida, utilizando os processos comentados anteriormente, onde se pode observar uma maior rapidez na detecção de erros na construção das classes, que foram corrigidos mais rapidamente.

Por conter informação vital da empresa, por questões de segurança de informação, não serão apresentados os códigos dos sistemas testados nessa monografia.

A ferramenta se mostrou bastante útil para as necessidades iniciais da empresa, mas como a empresa não dispunha de tempo hábil e recursos suficientes para implantar uma metodologia completa de processos de teste a curto prazo, decidiu-se que tal metodologia seria implantada com o tempo em cada sistema, de acordo com o diretor administrativo da empresa, Sr. José Álvaro Pimenta.

10.3 Avaliação do teste prospectivo

Para a implantação prospectiva da metodologia de testes focada no teste regressivo, usando o JUnit na empresa encontramos os seguintes dificuldades:

- Resistências do paradigma atual da empresa;
- Custo de implantação da nova metodologia.

A seguir cada uma destas dificuldades serão tratadas em maior detalhe.

a) Resistência do paradigma atual da empresa – sempre que um novo paradigma surge é habitual do ser humano criar barreiras para que a mudança seja realizada. De fato, este problema foi observado e podemos concordar com BASTOS et al. (2007 p. 21), que afirma "**é necessário remover certos 'preconceitos' ligados a esta atividade**", tais como:

- Considerar o testador um inimigo do desenvolvedor: como já foi explicado, o objetivo do testador não é mostrar que o programador errou e sim buscar eliminar a maior quantidade possível de erros no sistema antes que o mesmo entre em processo de produção, sendo assim ambos devem buscar pela qualidade do produto final.
- Montar equipes de teste com desenvolvedores menos qualificados: a atividade de teste requer como explica BASTOS et al. (2007 p. 21), "**uma equipe treinada e qualificada**". Devem ter em mente que quanto maior a qualidade do teste realizado, melhor será a qualidade final do software.
- Testar o Software após a conclusão do mesmo: a atividade de teste deve ser realizada durante todo o processo de criação do software, desde sua analise ate a produção final, como explica BASTOS et al. (2007 p. 21), "**o teste**

passou a ser executado desde o inicio do projeto de desenvolvimento do software, e não apenas ao final do projeto".

b) Custo de implantação da nova metodologia – Sempre que algo novo é apresentado para uma empresa, sua primeira preocupação é com o "custo" de sua produção. Os administradores estão sempre preocupados com os lucros de sua empresa e buscam diversas formas de reduzir custos, contudo em certas situações não se pode simplesmente cortar custos, a atividade de teste de software, embora acrescente um custo inicial maior ao projeto, reduz muito os custos de manutenção do mesmo, conforme afirma BASTOS (2007 p 20), "**quanto menos defeitos deixarmos no software, mais barata será sua manutenção no futuro**". BASTOS (2007 p.20) nos afirma ainda que "**o custo do software (desenvolvimento + manutenção) tende a ser relativamente menor quando o software é bem testado**". Tal afirmação pode ser visualizada na figura abaixo, que descreve os benefícios do investimento em teste de software.

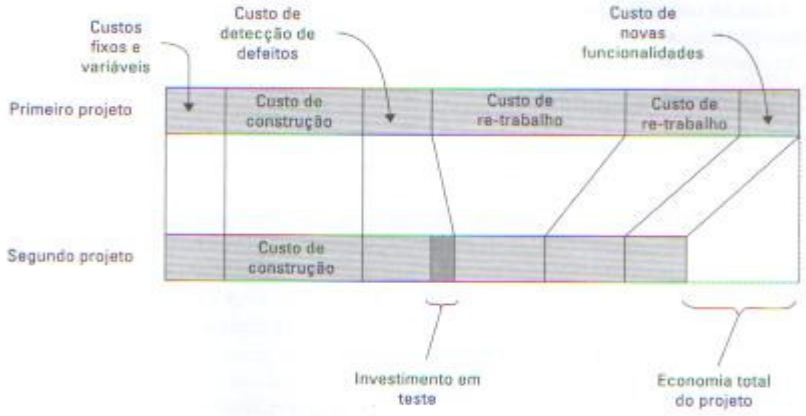

Figura 2: Benefícios do Investimento em teste de software

FONTE: BASTOS, A. et al. Base de Conhecimento em Teste de Software. São Paulo, Martins, 2007.

10.4 Implementação Futura

A partir da prospecção executada com o teste de arte da metodologia e, depois de avaliados os problemas encontrados, a gerencia da empresa decidiu por inserir a proposta de teste nos demais sistemas, contudo este processo será realizado na seguinte ordem:

- Sistema Contábil: por se tratar do sistema central da empresa, receber informações dos clientes e ser alimentado também pelos demais sistemas, o sistema contábil será o primeiro sistema a ser modificado adotando a proposta de teste.
- Sistema Fiscal: por ser o segundo sistema mais complexo da empresa, responsável pelo controle e calculo dos Impostos da área Fiscal, será o segundo sistema a ser modificado.
- Sistema Pessoal: por ser o mais complexo da empresa, este sistema será o ultimo a receber as modificações, visto ser este sistema o responsável pelo maior numero de pedidos solicitados pelos clientes.

Após a alteração de todos os sistemas, a implantação da proposta servirá para mostrar como é importante fazer uma boa carga de testes, assim como trabalhar com uma metodologia adequada para isto, evitando que erros conhecidos voltem a ocorrer e colocando foco naquilo que realmente importa para a empresa, o seu cliente.

11. CONSIDERAÇÕES FINAIS

O Processo de **Teste de *Software,*** como ferramenta para a obtenção de qualidade, se tornou imprescindível. Neste trabalho foram apresentados os conceitos e objetivos para se desenvolver um *software* com qualidade, assim como as técnicas mais usuais e conhecidas em seu desenvolvimento.

Foram apresentados os riscos advindos da falta das fases de teste ou de aplicá-las apenas na fase final do processo, definindo também que testes devem ser realizados em todas as fases de desenvolvimento de um sistema, desde sua concepção até sua implantação, tudo isto para reduzir os custos de manutenção, aumentar a qualidade do *software* e satisfação do cliente.

A importância de se elaborar uma base de testes abrangente, a fim de auxiliar a equipe de testadores, reflete-se na qualidade do *software.* O analista de testes deve procurar sempre atender a todas as situações possíveis, tornando o *software* cada vez mais confiável.

Assim, para a obtenção de um *software* de alta qualidade e plena satisfação para o cliente, deve-se procurar aplicar as técnicas e métodos do **Processo de Teste de *Software*,** sempre realizado por uma equipe especializada que utilize as ferramentas próprias para esse fundamento.

Foi apresentada uma ferramenta usual e de fácil manuseio para o controle e desenvolvimento de testes, o objetivo da proposta foi alcançado visto que atende as necessidades imediatas da empresa em melhorar seu controle de qualidade de teste de software e não impacta, por usar ferramentas open source, em um aumento significativo nos custos de produção do software.

O estudo de caso na empresa mostrou os principais passos para a implantação de parte de uma metodologia focada em testes regressivos. A partir da prospecção foram levantados os problemas para uma implantação definitiva que se mostrou viável para a necessidade da empresa, embora novas metodologias ainda precisem ser implantadas nos demais processos do desenvolvimento de sistemas, a aplicação do processo proposto já aumentou significativamente a qualidade do software produzido, reduzindo os erros e aumentando a confiabilidade dos sistemas desenvolvidos. No entanto, ficou claro que, sem a adoção de outras metodologias que levem à estruturação de sistemas baseados em componentes, o teste regressivo não poderá ser aplicado com eficácia. Na raiz do problema está a

necessidade de se trabalhar com unidades pequenas, parametrizadas e com poucos parâmetros para tirar proveito da força dos testes regressivos.

11.1 Trabalhos futuros

Implantar a metodologia de teste com o uso da ferramenta JUnit com base em uma metodologia de desenvolvimento de sistemas, onde podemos destacar o RUP[4], FDD[5], XP[6] dentre outros. Tais metodologias visam nortear o desenvolvimento de sistemas de forma a construir um software com maior controle e qualidade, dividindo o mesmo em componentes menores que são mais fáceis de serem corrigidos e testados, garantindo uma melhor qualidade no resultado final do sistema. Utilizar o JUnit sem uma metodologia de desenvolvimento de sistemas adequada, além de não trazer os benefícios prospectados, pode tornar o software difícil de ser atualizado, pois não se teve um controle desde o inicio de sua analise.

Para trabalhos futuros visando o aumento na qualidade do software produzido, sugerem-se a implantação de outros tipos de teste software, abrangendo não somente os testes unitários como também os testes de integração, testes de sistema e testes de aceitação, assim como a aquisição e uso de outras ferramentas já existentes no mercado.

[4] Rational Unified Process (ou Processo Unificado Racional) – Desenvolvido pela Rational Software Corporation, hoje adquirida pela IBM. Disponível em http://www.ibm.com/br/pt/

[5] Feature Driven Development (Desenvolvimento Guiado por Funcionalidades) – Disponível em http://www.heptagon.com.br/fdd

[6] Programação extrema (do inglês *eXtreme Programming*) - Disponível em http://improveit.com.br/xp

REFERÊNCIAS

BASTOS, Aderson; RIOS Emerson; CRISTALLI Ricardo; MOREIRA Trayahú. **Base de conhecimento em teste de Software.** São Paulo: Martins, 2007.

COX, BRAD J., NOVOBILSKI, A. J. (1991). *Object-Oriented Programming: An Evolutionary Approach*. 2nd ed. Addison-Wesley,

RIOS, Emerson & MOREIRA, Trayahú. **Teste de Software.** Rio de Janeiro, Alta Books, 2003.

NEVES, Luciane. ***A atividade de teste durante o ciclo de vida do software***. [on-line] [acessado em 27 de novembro de 2010] Disponível na internet em <http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1097>

DIAS NETO, Arilo Claudio. ***Artigo Engenharia de Software – Introdução a Teste de Software***. [on-line]. [acessado em 28 de novembro de 2010]. Disponível na internet em < http://www.devmedia.com.br/articles/viewcomp.asp?comp=8035>

CHINA, Luiz. ***Introdução a Qualidade de Software***. [on line]. [Acessado em 28 de novembro de 2010]. Disponível na internet em <http://bluesoft.wordpress.com/2010/07/19/intro-qualidade-de-software>

MACORATTI, José Carlos; ***Testes em desenvolvimento de Software – você precisa disto?*** [on line]. [Acessado em 28 de novembro de 2010]. Disponível na internet em < http://www.macoratti.net/tst_sw1.htm >

TOZELLI, Paulo. ***Processo de Teste de Software***. [on line]. [Acessado em 28 de novembro de 2010]. Disponível na internet em

<http://www.webartigos.com/articles/8299/1/Processo-De-Teste-De-Software/pagina1.html >

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. IEEE STD 829 Standard for software testing documentation. Nova York, IEEE Computer Society, 1998.

CAETANO, Cristiano. ***Artigo Engenharia de Software 4 - Verificação, Validação e Testes.*** [on line] [Acessado em 18 de julho de 2011]. Disponível na internet em <http://www.devmedia.com.br/articles/viewcomp_forprint.asp?comp=9879>

GLOSSÁRIO

R

Requisitos Funcionais: descrevem o comportamento do sistema, suas ações para cada entrada, ou seja, aquilo que deve ser realizado pelo sistema.

Requisitos não Funcionais: expressam como o software deve ser feito, definindo restrições ou atributos de qualidade para um software ou processo de desenvolvimento de sistemas.

T

Teste de Aceitação: Fase de teste onde se simulam operações de rotina no sistema, com objetivo de validar o comportamento do mesmo de acordo com as instruções solicitadas.

Teste de Integração: Fase de teste onde se objetiva encontrar falhas da integração dos componentes de um sistema, sendo geralmente encontradas falhas na transmissão de dados de um componente para outro.

Teste de Sistema: Fase de teste que visa executar o sistema sob a ótica do usuário final, executando funcionalidades na busca de falhas em relação aos objetivos pré-estabelecidos.

Teste de Unidade: Fase em que são testadas as menores unidades de software. O objetivo é de encontrar falhas no funcionamento individual de cada unidade de software.