UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

NATANAEL DAGOSTIN GHELLERE

MODELO DE GERENCIAMENTO DE COBERTURA FUNCIONAL DE TESTE DE SOFTWARE BASEADO EM RETORNO DO INVESTIMENTO (ROI)

CRICIÚMA
2012

NATANAEL DAGOSTIN GHELLERE

MODELO DE GERENCIAMENTO DE COBERTURA FUNCIONAL DE TESTE DE SOFTWARE BASEADO EM RETORNO DO INVESTIMENTO (ROI)

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: MSc. Gustavo Bisognin

CRICIÚMA

2012

NATANAEL DAGOSTIN GHELLERE

MODELO DE GERENCIAMENTO DE COBERTURA FUNCIONAL DE TESTE DE SOFTWARE BASEADO EM RETORNO DO INVESTIMENTO (ROI)

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Engenharia de Software.

Criciúma, 25 de Junho de 2012.

BANCA EXAMINADORA

Prof. Gustavo Bisognin - MSc - UNESC - Orientador

Prof. Paracelso de Oliveira Caldas -MSc - UNESC

Prof.ª Ana Claudia Garcia Barbosa - MSc - UNESC

Dedico este trabalho a minha família, aos meus amigos, principalmente a todos que sempre apoiaram e acreditaram em mim.

AGRADECIMENTOS

Os agradecimentos são para todos os meus familiares, amigos, professores que sempre me apoiaram e ajudaram nessa etapa da vida.

Ao meu pai Amilton Ghellere, a minha mãe Maria de Fatima Dagostin e a tia Neide, que sempre me incentivaram a fazer uma graduação.

Aos colegas de ônibus que durante dois anos e meio de segunda a sexta faziam o trajeto Jacinto Machado a Criciúma, e que sempre me faziam dar boas gargalhadas.

A colega de faculdade e prima Denise que sempre me ajudou quanto tive dificuldades nas matérias, e me ajudava a repor a matéria quando estava ausente das aulas devido a viagens de trabalho.

Ao meu orientador Gustavo Bisognin, que me apoiou e ajudou no desenvolvimento desse trabalho de conclusão de curso.

A minha namorada Fernanda que me aturou nos momentos em que eu precisava estudar para a elaboração desse trabalho.

“Uma Pessoa inteligente resolve um problema, um sábio o previne.”

Albert Einstein

RESUMO

Hoje em dia é fundamental a execução de testes em software, seja ele formal ou informal. Para a garantia da qualidade, o ideal é possuir uma equipe especialista em teste de software. Nesse trabalho de conclusão de curso, estuda a engenharia de software, com foco nos testes. Será abordado sobre teste e suas técnicas, para então chegar a um objetivo final que é o cálculo do ROI. O problema no qual foi proposto é testar um software garantindo o retorno de investimento. Para isso foi necessário criar um modelo de testes em que liste todos os casos de teste juntamente com sua priorização, para então fazer um planejamento se tal caso de teste será executado ou não. Conforme é realizado o planejamento dos testes a fórmula do ROI é automaticamente calculada e mostra o resultado, será possível planejar até que o retorno de investimento seja positivo. A partir do momento que o planejamento chegue a um resultado negativo, é necessário refazer o planejamento dos testes, assim eliminando alguns testes até que o resultado se torne positivo novamente. E assim, um dos princípios desse trabalho é a eliminação dos custos com os testes de software, a fim de criar um planejamento interno para a seleção de quais testes será executado, porém sempre pensando na qualidade do produto final.

Palavras-chave: Teste de software. ROI. Qualidade de software. Gerenciamento de testes.

ABSTRACT

Nowadays are fundamental running software tests, whether formal or informal. To ensure quality, the ideal is having an expert team on software tests. This final Project studies software engineering, focusing on testing. It`ll be discussed about tests and its techniques, and then achieve the final goal that is ROI calculation. The proposed problem is testing one software ensuring return of the investment. Thereunto was necessary create a tests standard that joins all the tests cases and its prioritization, and then planning if such test case will be or not executed. While the tests planning is being executed, the ROI formula is automatically calculated and shows the result, it will be possible planning until the investment return become positive. When the tests planning reach a negative result, is necessary a test replanning, therefore eliminating some tests until the result became positive again. And so, one of the goals in the project is eliminate some costs on software tests, in order to create an internal planning for choosing what tests will be executed, but always focused in the final product quality.

Keyword: Software tests, ROI, Software quality, tests management.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CDct	Custo de Defeitos com Testes
CDst	Custo de Defeitos em Testes
CT	Custo de Testes
ROI	Retorno do Investimento
TBM	Teste Baseado em Modelo
TC	Caso de teste
UC	Caso de uso
V & V	Verificação e Validação

SUMÁRIO

1 INTRODUÇÃO

Para garantir a qualidade de um produto de software, é preciso definir uma estrutura visando identificar o custo-benefício de sua qualificação, uma vez que, a qualidade do sistema esta diretamente relacionada ao seu custo de desenvolvimento. O processo tradicional propõe a criação de rotinas planejadas de teste, conforme conceituado por Pezzè e Young (2008). A execução destas rotinas é aplicada no esforço para encontrar e corrigir os defeitos que passaram despercebidos desde o levantamento de requisitos até a etapa de teste.

De acordo com Rios e Moreira (2006), o processo de testes deve estar sempre passando por uma rotina de melhorias e aperfeiçoamento, sendo que, para testar um software exige qualificação e dedicação do responsável.

O teste é definido como uma das etapas da fase de construção definida no ciclo de vida de desenvolvimento de um sistema, onde também são consideradas outras etapas da engenharia de software, como o levantamento de requisitos, modelagem, codificação e implantação. Para validarmos um teste é necessário ter um caso de testes em mãos, onde nele são definidas as situações que devem ser validadas.

Segundo Rios e Moreira (2006), a regra 10 de Myers define que o custo para correção dos softwares crescem exponencialmente de acordo com a evolução das fases do projeto de desenvolvimento. Sendo assim se o erro for encontrado em sua primeira etapa de vida, existirá custo para a correção, só que se o mesmo for encontrado na segunda etapa o custo será 10 vezes maior.

As organizações necessitam que sejam reduzidos os erros de software, para que também diminua os custos em relação à correção de futuros erros. Diante deste quadro, Rios e Moreira (2006) destacam que, as organizações estão em busca de redução de custos no desenvolvimento de software, porém também querem que o produto final esteja com qualidade, sendo que hoje em dia isso se tornou um desafio devido à complexidade dos negócios, ainda mais que os softwares estão crescendo de forma surpreendente.

Para alcançar o ponto ideal de qualidade do software, não necessariamente deve ser testado todo o caso de teste possível, pois o custo deste nível de cobertura, geralmente não é viável. Neste contexto Poderão ser testadas apenas as situações que tenham prioridade alta desconsiderando as demais, com

isso assume-se o risco de não realizar todos os testes possíveis. Contudo, a responsabilidade da pessoa que definiu a massa de teste aumentará, pois terá que ser garantido que o sistema seja liberado de uma forma que não contenha defeitos. O problema em questão é como garantir que o software será liberado sem defeitos, sendo que nem todos os casos de testes foram executados para garantir a sua cobertura funcional.

1.1 OBJETIVO GERAL

Elaborar um modelo para o gerenciamento de testes baseado em ROI para a validação dos requisitos funcionais dos produtos de software.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desse trabalho são assim definidos pelas alíneas abaixo:

a) estudar ROI em testes de software;
b) estudar sobre o ciclo de desenvolvimento e teste de software;
c) descrever sobre tipos de teste de software;
d) elaborar modelo de definição de testes baseado em ROI;
e) definir priorização dos testes;
f) aplicar ROI em uma massa de teste;
g) analisar resultados do ROI.

1.3 JUSTIFICATIVA

Para Rios e Moreira (2006, p. 01) “Informações de mercado dizem que mais de 90% dos sistemas são liberados com graves defeitos, software com problemas de performance e com defeitos na execução são custosos”. Como evitar essa situação? É de acordo com essa afirmação que as empresas estão adotando o aprimoramento em seus testes de software.

“Quando tratamos de testes de software, estamos também nos referindo à qualidade de software” (RIOS; MOREIRA, 2006, p. 02). Sempre que executado um teste completo, pode-se dizer que o software tem qualidade, assim será liberado ao

usuário com a confiabilidade necessária para ser usado, sem que o usuário fique desconfiado se tal ferramenta realmente funciona como deve ser.

Segundo Rios e Moreira (2006, p. 02) “devido às pressões crescentes para a liberação desses sistemas, tais atividades nem sempre são executadas com a eficiência que seria requerida”.

Para executar um teste, primeiramente deve ser feito o planejamento referente aos ciclos de execução e a cobertura funcional em relação ao que deve ser testado, visto que em muitas situações não se tem o tempo necessário para efetuar todos os processos. Levando em consideração o curto tempo, como realizar os testes garantindo que o software terá qualidade? Quais as situações que realmente é necessário testar?

De acordo com Pezzè e Young (2008, p. 26), “O custo da verificação de software frequentemente corresponde a mais da metade do custo total do desenvolvimento”. Com base nesta afirmação, é visível que de alguma forma deve-se tomar alguma providência pelas empresas de software para tornar seus softwares confiáveis, e também ter um maior retorno do investimento, evitando futuros gastos com correções.

Após o término da implementação do sistema, o mesmo deve passar pela equipe de testes, sendo que deve ser realizado um gerenciamento e estudo em relação ao caso de teste. Em meio a ele, será elaborada uma fórmula que irá definir as prioridades de cada situação e também definirá até quando testar o requisito em questão, também levando em consideração a qualidade do mesmo. A partir dessa fórmula as empresas de sistemas poderão estar diminuindo o tempo de testes e aumentando o ROI, pois se não testar todas as situações necessárias o custo para a correção desse software será grande, sendo que o mesmo poderia ter sido corrigido no momento dos testes.

O presente trabalho irá apresentar uma proposta de gerenciamento de testes baseada em ROI, fornecendo uma base sólida para a tomada de decisão sobre qual massa de testes deve ser executada, proporcionando ao engenheiro de teste responsável, a segurança necessária para assumir o risco referente à redução da cobertura funcional mantendo a qualidade do produto de software testado. O objetivo principal é encontrar através de uma fórmula em planilha eletrônica o ponto ideal de testes dos requisitos funcionais, assim diminuindo a quantidade de defeitos e aumentando o retorno do investimento.

1.4 ESTRUTURA DO TRABALHO

No primeiro capitulo foi definido a introdução do trabalho proposto, onde se detalhava qual o problema se encontra atualmente nas empresas que não possuem equipe de teste, ou até mesmo possuem, mas não possui um gerenciamento desses projetos. E então foi elaborada uma justificativa, e que será criado algo em que irá apoiar os engenheiros na rotina dos testes.

Para desenvolver tal ferramenta, é necessário o estudo dos processos de teste, esse item está sendo abordado no segundo capitulo desse trabalho. No capitulo em questão foi e estão ainda sendo estudados, alguns pontos que está diretamente ligada à rotina de testes, isso para ter uma base sólida e poder prosseguir com os demais estudos.

No terceiro capitulo é definido os ciclo de vida do teste, de quando se inicia os testes de integração e sistema até a fase de regressão. Essas fases se fazem necessário para entender a rotina de teste em uma empresa de software, pois essas etapas é que oferecem uma maior organização na rotina dos testes.

No quarto capitulo é definido então a questão de investimento, esse melhor conhecido como ROI, pois é com estudos nele que será trabalhado na terceira etapa desse trabalho. Será utilizado o retorno de investimento aplicado sobre os testes de software a fim de reduzir os custos que existem com relação a defeitos do software quando não identificados pela equipe de testes.

Nos capítulos 7 e 8, é apresentado como foi elaborado o retorno do investimento (ROI) sobre os testes de software, juntamente com os resultados alcançados com a pesquisa.

2 PROCESSOS DE TESTE DE SOFTWARE

Um software pode ser considerado como sendo um conjunto de regras e funções que possuem elementos de entrada e saída. Nesse contexto o teste de software é executado afim de que possa ser garantido que o software execute sem que exista uma saída com valores inesperados, ou até mesmo que durante o processamento não ocorra um defeito. Assim é aplicado o teste para que seja possível garantir que o software reaja exatamente como foi programado, evitando situações de exceções, garantindo assim a alta qualidade (NETO; SANTOS, 2001).

Pode-se dizer que o processo de teste de software tem como principal objetivo a obtenção de valores válidos de saída, ou seja, após executar o software é validado o seu funcionamento comparando os valores fornecidos na entrada de dados com os valores apresentados como saída, podendo-se assim observar se o comportamento apresentado pela aplicação é de fato o esperado. Neste contexto, pode-se evidenciar a presença de defeitos ou oportunidades de melhoria no sistema testado, (NETO; SANTOS, 2001).

A garantia da qualidade de um software é determinada pela quantidade de defeitos apresentada durante seu ciclo de evolução, tornando a atividade de validação indispensável no processo de garantia a qualidade dos produtos gerados pelos projetos de software.

A afirmação de que um produto possui qualidade, só é possível com a certificação que o sistema apresente um baixo índice de defeitos. No intuito de obter esta garantia, é necessário que seja feito um levantamento de casos de testes a ser realizada, preparação do ambiente de teste, formalização da equipe na qual efetuará os testes, formalização do cronograma etc.. .Todos esses itens, entre outros, fazem parte do processo de teste, cuja função é garantir que as etapas essências para a garantia da qualidade dos projetos de software sejam efetivamente cumpridas.

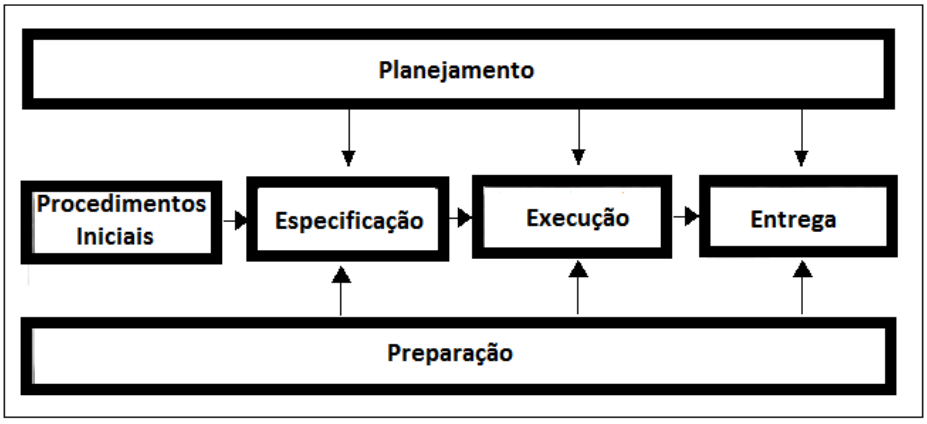
Segundo Rios (2007?), o processo de teste de software é diretamente responsável pelos riscos de defeito no produto, pois é no processo de teste que é elaborado o plano de teste, onde contém todo o levantamento dos possíveis testes, esse denominado de casos de testes. Durante o processo de teste, todas as funcionalidades do software têm que ser bem definidas, pois se esquecer de levantar algum caso de teste, ou esse ser mal definido ou estar no plano de testes e a equipe de teste não executar todos os procedimentos passados pelo engenheiro

de teste, pode-se dizer que o levantamento foi mal executado ou testado de forma incorreta. Rios (2007?) ainda ressalta que igual ao desenvolvimento de um sistema, o processo de teste também possui um ciclo de vida, com inicio e fim, sendo que seu inicio começa junto ao desenvolvimento do software, e termina quando o produto é liberado ao usuário final.

Para Rios e Moreira (2006), o processo de teste deve possuir etapas, assim tornando o ambiente mais organizado e garantia de uma maior definição com relação aos processos a executar durante os testes, esses aspectos influenciam diretamente na qualidade do software. É definido assim os processos por seis etapas conforme figura 1 e detalhadas logo abaixo:

a) **procedimentos Iniciais**: essa é a primeira etapa, onde é elaborado um documento com a definição do produto no qual serão testados, quais os objetivos do teste quando aplicado a esse produto, levantamento do pessoal que será envolvido para garantir a qualidade do produto, as responsabilidades de cada um, os riscos, entre outros;
b) **planejamento**: nessa etapa é feito o levantamento das estratégias e do plano de teste, depois de concluído é revisado;
c) **preparação**: aqui é preparado o local onde serão efetuados os testes;
d) **especificação**: levantamento da cobertura funcional do produto, elaboração e revisão dos casos de teste;
e) **execução**: inicio dos testes, onde é executado o caso de teste e preenchimento dos resultados obtidos em cada caso;
f) **entrega:** após o encerramento dos testes e correção, nessa etapa é realizado a entrega do produto.

Figura 1 - Processos de teste de software



Fonte: RIOS; MOREIRA (2006, p. 9)

O processo de teste é fundamental para garantia de qualidade do software, com o processo bem formalizado e executado, além de o ambiente de testes ficarem mais agradável pela organização, o teste se torna mais eficaz e robusto. O processo favorece também a equipe que realiza os testes, pois os casos de testes são mais estruturados, deixando-os bem definidos e detalhados para ser executado.

2.1 DEFINIÇÃO DE TESTE DE SOFTWARE

O teste é aplicado em um software com um único objetivo, encontrar defeitos. Muitas das vezes o software é liberado ao usuário final, sem a devida revisão para verificar se esse realmente realiza as atividades conforme planejado, para as empresas que desenvolvem software existe um custo muito alto nessa situação, isso pela questão de ter que corrigir algo que poderia ter sido encontrado enquanto o software ainda estive-se na empresa. É exatamente para que não ocorra esse tipo de situação que hoje em dias as empresas têm adotado o teste de software, é a garantia de qualidade para o produto e redução de custos com correção. Está cada vez mais visível para as empresas a necessidade de formalização de uma equipe de testes (SEVERO, 2007).

Pode-se dizer que o teste de software é a garantia de que o produto está sendo liberado ao cliente, com a qualidade na qual o mesmo espera, sendo que a confiança sobre um software está diretamente ligada a sua funcionalidade, quando o produto tem muitos defeitos ele acaba se tornando algo sem garantia para o cliente. Após o desenvolvimento do produto, esse tem que passar por um setor de testes que irá realizar várias simulações de possíveis entradas no sistema para se obter a saída, é então verificado se o software se comportou conforme estava especificado no caso de teste. Dessa forma são evitados possíveis defeitos após estar com o usuário final, pois se esse fosse entregue ao cliente, sem ter passado por todos os devidos testes, o produto poderia estar com defeitos, sendo que a empresa perderia total credibilidade com cliente e o custo para a correção é muito maior se levado em consideração que poderia ter sido corrigido durante processo de teste.

O principal objetivo do processo de teste é encontrar defeitos, sendo assim, somente pode-se considerar um teste com sucesso aquele que encontra defeitos durante o processo, e quanto maior o número de defeitos encontrado, maior

a qualidade dos testes, assim são minimizados os defeitos e praticamente eliminadas as chances que o produto final tenha algum defeito, garantindo a satisfação do cliente com o produto (SEVERO, 2007).

A equipe de testes é a grande responsável pela garantia do produto, além dos testes, são eles que dizem se o software está apto ou não para ser liberado ao cliente, e depois de passar pelos testes e estiver conforme estabelecido no plano, não irá passar por mais ninguém da empresa para revisão dos requisitos, apenas o usuário final que irá identificar se ficou ou não algum defeito para traz. É no teste que se faz a revisão de especificação, projeto e da codificação (PRESSMAN, 2003).

Para Rios e Moreira (2006), o teste de software se caracteriza na verificação do produto, onde é feito a comparação do que foi proposto, com o que o programa realmente está fazendo, e levantamento também do que era para fazer, mas não está fazendo, assim o teste é a validação do produto final.

É aplicado o teste de software com o propósito de fazer uma verificação e validação, onde a verificação nada mais é que a garantia que o produto realiza realmente uma determinada funcionalidade, e na validação é analisar se a o retorno do software está dentro das solicitações do cliente (CARTAXO, 2006).

2.1.1 Verificação e Validação

O teste de software está diretamente relacionado a dois pontos fundamentais, esses definidos de V & V, ou Verificação e Validação. Eles englobam várias atividades que se diz respeito à qualidade de um software (PRESSMAN, 2000):

a) **verificação**: são todas as atividades que tem em vista assegurar que no software está implementado corretamente uma função especifica.
b) **validação**: é o conjunto de atividades que visa assegurar que o produto está em conformidade com os requisitos do software, ou seja, que o produto está sendo desenvolvido de acordo com os requisitos estabelecidos inicialmente.

2.1.2 Teste baseado em modelo

Nos dias de hoje cada vez mais vem se popularizando a orientação de objetos e a utilização de modelos na engenharia de software, e no meio disso está tendo um grande desenvolvimento na técnica de teste funcional conhecida por TBM, ou teste baseado em modelo (ANDRADE, 2007).

A definição para teste baseado em modelo é diretamente ligada à geração de teste de software a partir de modelos de aplicação para futuros testes de caixa preta, nesse caso os casos de teste são executados para avaliar a correspondência entre o modelo criado e o sistema. A especificação do software que receberá o teste necessita estar formalmente ou semi formalmente descrita em um modelo de modo que seja caracterizado de forma precisa seu comportamento (BEIZER, 1995).

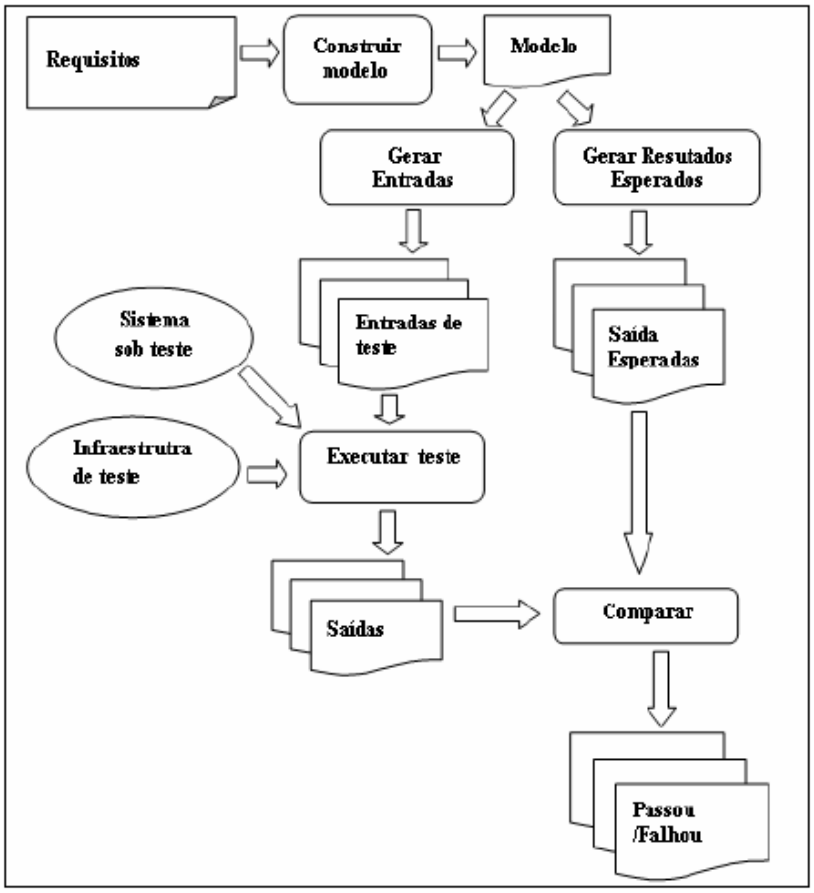
As atividades diretamente relacionadas ao teste baseado em modelo estão exibidas na figura 2 e também descritas a seguir (CARTAXO, 2006):

a) **construir o modelo**: nessa etapa é elaborado o modelo formal ou semi formal, ele é construído utilizando os requisitos da aplicação;
b) **gerar entradas**: as entradas do teste são geradas partindo do modelo, elas serão utilizadas para exercitar o software que está no processo de teste;
c) **gerar saídas esperadas**: as saídas esperadas durante o teste são geradas de acordo com o modelo formal. Essas são responsáveis por dizer se o software fez ou não de forma correta o processo proposto;
d) **executar os testes**: o software é exercitado com as entradas e então é obtido as saídas;
e) **comparar saídas com saídas esperadas**: Aqui é comparado se a saída do software é a mesma saída na qual foi gerado a partir do modelo.

O processo de teste baseado em modelo é iniciado juntamente com os requisitos, assim que todos os requisitos estiverem definidos pode-se iniciar o processo de teste. Estando os requisitos já definidos, a próxima etapa então é construir um modelo que retrate de forma integra o comportamento requisitado. Tendo o modelo construído, é possível ter uma melhor visão e entendimento quanto o software e inicia-se a elaboração dos casos de teste esses muitas vezes são

gerados através de software de forma automática, nos casos de teste contém as entradas e saídas esperadas. Após formalizar todos os processos são então iniciados os testes sobre o sistema assim podendo comparar as saídas do software com as saídas esperadas, essa comparação é feita para identificar se a funcionalidade possui ou não defeito.

Figura 2 - Abordagem de teste baseado em modelo



Fonte: CARTAXO (2006, p. 13)

2.2 TESTES DE CAIXA BRANCA

Teste de caixa branca é também conhecida por teste estrutural, é dada a caracterização de caixa branca, pelo propósito de que se pode ver o que tem no interior dela, se aplicado isso em um software, o conteúdo da caixa seria o código fonte (PRESSMAN, 2003).

“O teste de caixa branca é um método de projeto de casos de teste que usa a estrutura de controle do projeto procedimental para derivar casos de teste” (PRESSMAN, 1995, p. 793).

O teste se inicia desde o momento que está sendo desenvolvido o software, aonde após concluir um trecho de implementação o programador testa no próprio ambiente de desenvolvimento. Assim não é testado o software por inteiro, e sim apenas por partes do código fonte.

O teste de caixa branca, dado esse nome por ser contraditório ao teste de caixa preta, é diretamente focado em torno da estrutura interna da implementação. Nesse caso, o testador tem que ter total acesso ao código fonte do programa para realizar os testes, diferentemente do teste de caixa preta, no qual é executado sobre o produto final.

2.3 TESTES DE CAIXA PRETA

O teste de caixa preta é aplicado para a validação dos requisitos funcionais sem se preocupar com o código fonte, ele é aplicado sobre o software em sua forma final, ou seja, quando já possui um executável, por isso o nome de caixa preta, onde se imagina algo em que não é possível ver seu interior.

Pressaman (1995) destaca que o teste de caixa preta, ou teste funcional, é aplicado em cima dos requisitos funcionais do software. Esse método é utilizado para encontrar defeitos diferentes dos encontrados quando usado o processo de teste de caixa branca. O teste de caixa preta tem como objetivo ter várias condições de entrada em um software e a partir disso testar todos os requisitos funcionais de um programa, observado o processo de entrada até sua saída.

Esse teste recebe o nome de caixa preta, pelo fato de que o testador não possui nenhum acesso ao código fonte do programa, é uma caixa fechada, onde só é possível visualizar o lado externo do produto, ou seja, a entrada fornecida para execução, e as respostas que são considerados as saídas. Nesse teste apenas são verificados as funções sem estar verificando o código fonte para correções (MYERS, 1979). O testador irá apenas seguir o roteiro de testes onde tem as especificações funcionais do sistema, chamados de caso de testes (BEIZER, 1990). Dessa forma, a especificação tem que ser correta e com um nível de detalhes desejável e de entendimento ao testador, para garantir que todos os testem sejam executados e o software seja liberado sem defeitos (VINCENZI, 1998).

O teste funcional tem dois aspectos principais que envolvem toda sua lógica, a primeira é a identificação das funções que o software deve fazer, assim

criando o caso de testes, a outra é usar os casos de teste para checar se realmente o produto realiza as funções esperadas (PRESSMAN, 1992).

O objetivo do teste funcional é encontrar defeitos relacionados a cinco categorias, denominadas assim por Pressman (2002):

a) funções incorretas ou omitidas;

b) erros de interface;

c) erros de estrutura de dados ou de acesso aos dados externos;

d) erros de comportamento ou desempenho;

e) erros de iniciação e término.

O teste funcional é utilizado para revelar falhas que com a metodologia de teste baseada em código, teste estrutural, muitas das vezes não é encontrada, pois testar o software por partes é diferente do que testá-lo como produto final (BEIZER, 1995).

2.4 DEFINIÇÃO DE COBERTURA FUNCIONAL

A definição de cobertura funcional é de extrema importância para garantia da integridade do software a ser testado. Pois é ela que garante que todas as funcionalidades do software estejam descrita em um documento para que quando essa sofrer alterações possa assim ter um conjunto de testes já formalizados para realizar, garantindo que todas as possíveis entradas no sistema tenham sido exercitadas.

Silva (2007) define cobertura funcional, como uma técnica que é utilizada para medir o quanto da funcionalidade do software está sendo verificada. A cobertura funcional é um dos itens importantes no segmento de testes, pois também é responsável por garantir a qualidade de software, é através dela que é relacionado tudo o que for possível realizar em uma determinada tela do aplicativo a ser testado.

Buraco de cobertura é quando não é feito corretamente a cobertura funcional de um software, ou seja, não foi levantada alguma funcionalidade do software (SILVA, 2007).

Para Silva (2007), na execução dos testes funcionais, um dos maiores problemas é saber quando parar os testes garantindo que a verificação esteja completa, ou seja, ter a garantia que todas as funcionalidades tenham sido testadas.

Nesse caso ele sugere que exista por meio de algum mecanismo, uma forma de detectar se todas as funcionalidades realmente foram testadas.

A cobertura funcional nada mais é que uma técnica utilizada a fim de medir o progresso dos testes e reportar as funcionalidades do software na qual não foram testadas. A total cobertura ajuda na qualidade dos testes e conseqüentemente na qualidade do software, através da cobertura também é mais fácil de descobrir áreas que não foram adequadamente testadas, assim direcionando os testes para criar e cobrir essas áreas não testadas, elas são mais conhecidas por buraco na cobertura. Esse deve ser descoberto e solucionado o quanto antes, pois se existir buraco na cobertura automaticamente irá impedir que o software tenha sua verificação completa, dessa forma deixando que possíveis defeitos passem despercebidos durante a fase de testes (SILVA, 2007).

Silva (2007) descreve que uma das formas de definir uma total cobertura funcional é através do modelo cross-product, que também é conhecida por cross-coverage. Esse é um modelo criado onde é executado testes em todas as possibilidades na qual existe dentro de um conjunto de funcionalidades.

No modelo de cobertura funcional cross-coverage, um exemplo pode ser definido da seguinte forma, na funcionalidade de login de qualquer sistema existe diversos testes a ser realizados, considerando apenas os campos de login e senha, como podemos definir alguns testes abaixo:

a) **teste 1**: usuário informa apenas usuário;

b) **teste 2**: usuário informa apenas senha;

c) **teste 3**: usuário informa usuário correto e senha incorreta;

d) **teste 4**: usuário informa usuário incorreto e senha correta.

Nesse caso foram especificados apenas quatro possíveis testes a ser executada na funcionalidade do sistema, essa foi à cobertura funcional levantada a principio da tela de login, os demais testes que não foram relacionados ficam assim definidos como buraco na cobertura.

Sendo assim o espaço de cross-coverage é formar a partir de todas as funcionalidades de um sistema, as possíveis entradas do usuário na funcionalidade e essas devem ser todas testadas, garantindo assim total cobertura funcional do software e garantia de qualidade ao usuário final (SILVA, 2007).

2.5 DERIVAÇÃO DE CASO DE TESTE

Caso de testes funcionais é utilizado como roteiro de testes, é nele que contém o procedimento correto de como e quais valores serão utilizados para alimentar o sistema, e determina qual a saída o software terá ao finalizar a operação proposta no caso de teste.

Um caso de teste é considerado um conjunto de vários testes onde esses são executados em sequencia em busca de um objetivo, obter um maior controle sobre os testes e reduzir o tempo gasto para execução dessa rotina. Um caso de teste vai gerar vários testes, onde nele terá valores de entrada, saídas esperadas e qualquer outra informação adicional que envolve a funcionalidade a ser testada, tudo isso para facilitar a execução dos testes (FEWSTER; GRAHAM, 1999).

O caso de teste é um dos principais itens que contribui para a qualidade dos testes, de acordo com Myers (2004) o caso de teste tem por tarefa encontrar defeitos durante o teste, sendo que quanto maior a quantidade de defeitos detectada pelo caso de teste, maior será o sucesso do teste. Mesmo assim ainda existem pessoas que acreditam que o caso de teste de sucesso é aquele no qual não encontrou defeitos durante sua execução.

Nos casos de teste são definidas todas as funcionalidades possíveis da aplicação, garantindo uma alta definição da cobertura, e redução das funcionalidades não testadas, dessa forma o testador pode executar uma a uma comprovando que o produto realiza as atividades conforme determinado, os casos de testes podem ser definidos também durante o desenvolvimento da aplicação (CARTAXO, 2006).

O software quando chega a sua fase final de desenvolvimento, ele é encaminhado para a equipe de testes, é necessário um roteiro de testes, esse por sua vez é definido como os casos de testes. Nesse documento possui todos os requisitos funcionais da aplicação e como testá-los, considerando várias situações para entrada. Dentre esses casos o testador precisa executá-la de acordo com o especificado, e verificar qual sua saída, e comparar com o especificado no caso de teste, se o resultado obtido for igual ao definido no caso de teste, essa funcionalidade está correta, agora caso a saída do software não seja conforme determinado, esse deve ser descrito no roteiro de teste e encaminhado para correção.

Para Jorgensen (1995), o ponto principal para o teste de software é no momento da derivação dos casos de teste, onde se define que o caso de teste é estruturado pelas entradas e saídas. A entrada possui a condição inicial, e os passos que deve ser realizado para se obter as saídas. Na saída é definido qual resposta o sistema deve emitir, levando em consideração a entrada inicial, e por fim a pós-condição que representa o estado final do sistema. É dessa forma que deve ser formalizado um caso de teste, pronto para o testador utilizar, com todas as condições possíveis para testar e o máximo possível de derivação de suas funcionalidades.

O mais indicado é ter um caso de teste com todas as derivações possíveis, ou seja, descrever todas as funcionalidades do software. Pois com um caso de teste bem definido, o objetivo é que possa ser encontrada a maioria dos defeitos, com o mínimo de tempo e esforço, fazendo com que a produtividade seja maior.

Uma funcionalidade do software só irá funcionar corretamente se tal função for feita manualmente antes do desenvolvimento do software, sendo assim, o programador só irá implementar após entender como deve funcionar o sistema, e para isso se faz necessário a aplicação da funcionalidade em um caderno para entender o que será feito do inicio ao fim. Com base nisso, definimos que para criar os casos de teste também existe a necessidade de realizar o processo manualmente, isso para se obter as saídas esperadas para comparação com as saídas do software.

Existem algumas técnicas que fazer a derivação dos casos de teste de forma automática, sem a necessidade de ter que fazer o processo manualmente cada vez que o software tem modificações, dessa forma a cada alteração ou desenvolvimento de funcionalidade nova no sistema, são gerados automáticos os casos de testes para efetuar os testes.

Após deixar caso de teste pronto, a função do testador é executar um a um os casos de acordo com o roteiro montado, e registrar os resultados obtidos, sempre tentando alcançar um único propósito, nesse caso encontrar defeitos. Após compara-se a saída com o resultado que seria esperado para então definir se o teste passou ou não. Quando no teste é encontrado defeitos, o software deve passar por correções para então voltar à equipe de testes, esse processo é realizado sempre

que encontrados defeitos, quando não mais encontrado qualquer divergência, então é liberado ao usuário final.

2.6 COMPOSIÇÃO DA MASSA DE TESTE

Para ser possível efetuar os testes em um software é necessário ter em mãos um documento, onde nele terá um roteiro para ser seguido, e exatamente o que tiver descrito nesse roteiro a equipe de testes tem que seguir. Os casos de testes, definido anteriormente, têm por objetivo conter toda a massa de teste, ou seja, todos os casos de testes acabam se tornando a massa de teste. Dentro dos casos de testes é possível ter diversos testes, entre eles está relacionado cada cenário e como o software terá que se comportar de acordo com as entradas que também está especificada nele. É de muita importância ter uma boa massa de testes, assim é possível ter todos os testes que irão garantir que o software não tenha nenhum defeito.

Porém, mesmo tendo toda a massa de teste nem sempre é necessário efetuar todos os testes dessa massa de teste, pois muitas vezes não tem tempo hábil para efetuar tal procedimento sendo que existem processos mais críticos para ser testado. Assim testam-se as funções do sistema, mas não necessariamente são testadas todas as possíveis entradas, pois esses dados não vão deixar o software com baixa qualidade e nem será liberado o software ao cliente com defeito considerado de nível importante. Essa situação geralmente acontecerá apenas com os processos de menor vulnerabilidade.

3 CICLO DE VIDA DE TESTE DE SOFTWARE

O teste de software como qualquer outro processo eficaz necessita de organização para obter uma maior qualidade no resultado final, esse é definido pelo ciclo de vida do teste. Ele é dividido em etapas com o intuito de criar uma organização e dividir em etapas os testes, assim evitando que tudo seja testado de uma única vez, logo garantindo a qualidade do software e satisfação do usuário final.

Para Pressman (2002), ele destaca que as fases de testes são consideradas uma estratégia para a condução dos testes, nada mais é que a separação dos testes para esses serem executados por etapas, eles iniciam-se através dos testes aplicados a um nível de componentes, e conclui-se quando os testes se aplicam no sistema integrado.

Os testes são organizados assim por três fases, essas definidas por Pressman (2000) da seguinte forma:

a) **teste de unidade**: Aplica-se o teste por módulos sem qualquer integração;
b) **teste de integração**: É o teste aplicado quando já se possui o software em forma de executável, tem por objetivo testar a interface;
c) **teste de sistema**: Momento em que se testa o software como um todo, verifica se realmente a integração entre os módulos está de acordo.

Também existe a fase de regressão, que essa é definida pelo teste aplicado ao software que já passou pela equipe de teste e retorna novamente com as devidas correções nos defeitos encontrados primeiramente.

Mesmo aplicando essas fases de teste a um software, não se pode dizer que essas irão exterminar quaisquer defeitos que ainda exista. Porém se for seguido um caminho com maior organização e nível de detalhamento, pode-se aumentar as chances de que o software execute de forma correta as suas funcionalidades especificadas. Contudo ainda existem limitações próprias da atividade de teste, se aplicados todos os processos de maneira bem planejada pode-se garantir ao software características que influenciam diretamente na qualidade do produto como também para o seu processo de evolução (DELAMARO, 1993).

3.1 FASE DE INTEGRAÇÃO

O teste de integração é considerado uma das fases do ciclo de um teste de software, essa tem como objetivo encontrar defeitos com relação à interface da aplicação, já que no teste de unidade são testadas apenas as funções do software sem se preocupar com interface, logo fica reservado para essa fase o trabalho de encontrar defeitos na interface do produto.

Pressman (2002) afirma que no teste de integração é objetivado a encontrar defeitos associados à interface. Nessa etapa é construída a estrutura do sistema na qual teve determinação no projeto. Ele ainda diz que esse teste é dividido assim em dois tipos: incremental e não incremental, assim definidas por Pressman (2002):

a) **integração não incremental**: Os módulos são integrados e o teste é aplicado sobre o sistema completo;
b) **integração incremental**: Os testes são aplicados em blocos, nesse caso a interface têm maior probabilidade de ser totalmente testado, aumentando as chances de encontrar defeitos.

Mesmo após o software ter passado pelo teste de unidade, não quer dizer que esse esteja trabalhando corretamente, o teste de unidade apresenta limitações e não se pode garantir que cada unidade trabalhe adequadamente em todas as situações, pois até então os testes foram com relação apenas sobre determinadas funções, mas quando combinadas estas a outras funções podem ocorrer divergências (DELAMARO, 1997).

O teste de integração é aplicado para eliminar qualquer defeito que o software tenha com relação à interface, sendo que se encontrado defeitos desse tipo em um estágio mais avançado, principalmente se esse defeito tiver relação direta com a unidade já testada e essa ter que passar novamente por alteração e testes o seu custo para correção é muito elevado, se considerado que poderia ter sido corrigido durante os processos de teste.

3.2 FASE DE SISTEMA

O teste de sistema é executado pela equipe de teste, e visa à execução dos testes no software como um todo, ou seja, como se tive-se sendo utilizado pelo

cliente. Nesse teste devem ser executadas todas as funções do software de forma que chegue o mais próximo possível da situação real desse quando executado em ambiente de produção.

Mas segundo Orozco (2009) o teste de sistema não é responsável por testar as funções do sistema completo, até porque nesse caso seria o mesmo processo de teste funcional, assim se tornando redundante. Esse tipo de teste tem uma particularidade que é de comparar o sistema com os objetivos definidos no inicio do levantamento de requisitos.

O teste de software segue um único objetivo sempre, que é encontrar defeitos no software em questão (PRESSMAN, 2009), ou seja, garantir que esse esteja de acordo com os requisitos inicial propostos. Quando se fala em processo de teste de software, também nos referimos em etapas de teste, segundo Sommerville (2003) essas etapas de testes, vão adquirindo complexidade e formalidade, assim podendo se tornar o teste do momento em um processo simples ou até mesmo em um teste complexo.

Ao procurar por diferenças entre o programa e seus objetivos, esse teste foca-se nos defeitos gerados durante o processo de definição das especificações externas. Assim tornando esse teste um processo de extrema importância que identifica defeitos durante a fase do ciclo de desenvolvimento. Outro ponto que diferencia esse teste do teste funcional é que para derivar os casos de testes, não pode ser utilizada as especificações externas (OROZCO, 2009).

O teste de sistema é realizado após a integração do sistema, esse teste visa encontrar e corrigir defeitos de funções e características de desempenho que não estejam de acordo com a especificação (PRESSMAN, 2000).

Esse teste é responsável por validar o software considerando que o mesmo tenha sido incorporado a um sistema maior, onde houve a integração de vários sistemas que se comunicam entre si, nesses casos os testes são aplicados sobre o software afim de simular a real situação do ambiente de trabalho. Essa técnica tem por objetivo assegurar que o software e os demais elementos que compõem o sistema, tais como, hardware e banco de dados, vão ter o desempenho adequado para rodar as aplicações integradas, nessa fase utiliza-se a técnica de teste funcional (PRESSMAN, 2000).

3.3 FASE DE REGRESSÃO

O teste de regressão visa à garantia que o software permaneça sem defeitos depois de novos testes serem realizados, ou seja, esse é o tipo de teste que ocorre quando o software foi testado e foi para correção, após retornar para equipe de teste, o sistema não pode ter defeitos em funcionalidades já testadas e aprovadas pelos testes.

A definição de teste de regressão para Sommerville (2003) é representada por uma etapa mais específica, que por sua vez garante que todo o esforço dos testes já aplicados em outro momento em relação ao software continue oferecendo a garantia de qualidade após esse sofrer qualquer alteração.

O teste de regressão pode ser definido como uma estratégia importante para redução de defeitos. Consiste em refazer novamente os testes que já foram realizados em versões anteriores a fim de garantir a integridade das funcionalidades não alteradas na nova versão.

O desenvolvimento de um software complexo é longo e continuo, muitas vezes sofrendo impactos de mudanças drásticas durante ou após o desenvolvimento. Seja por correções de defeitos ou implementação de novos processos, até mesmo mudança na forma de realizar determinado processo, ou seja, o software sofre constantes modificações, sempre com objetivo de melhorar seu desempenho e funcionalidades. Um dos problemas é que todas essas alterações sempre são realizadas por mais de um desenvolvedor (SIMONS, 2010). Nesse caso o software pode ter maior probabilidade a ter defeitos, pois nem todos os desenvolvedores que fizeram as alterações nele conhecem todos os processos, onde pode ocorrer de mexer em determinada função e alterar outra na qual já havia sido testada, isso pelo fato de não ter conhecimento no software.

Os casos de teste de regressão devem ser reutilizados quando o software sofre alterações, assim garantindo que qualquer alteração no sistema não afete a já alcançada estabilidade de funcionamento do mesmo (RAKITIN, 2001). Dessa forma, qualquer alteração na qual o software venha a ter, tem que ser testado seguindo o roteiro de testes, mas o engenheiro também deve se preocupar com as funcionalidades do software que não sofreram alterações, pois existe a probabilidade de se encontrar defeitos nelas, esses testes são assim definidos por fase de regressão.

O software tem que se manter sempre atualizado, em busca de melhorias para manter a satisfação do cliente. Só que de nada adianta promover inovações e aplicar alterações exigidas pelos clientes para novas funcionalidades no sistema se essas irão alterar de forma negativamente as funcionalidades na qual já vinham sendo executadas de maneira desejável, por isso é que foi criada a fase de regressão, para manter a integridade dessas funcionalidades na qual não sofreram alterações (ELFRIEDE, 2003).

Após o software sofrer uma manutenção, onde sofreu modificações no sistema, novos testes devem ser realizados, esses são denominados de teste de regressão, onde o principal foco é encontrar defeitos nas funcionalidades do software que foi modificado (PRESSMAN, 2000).

A etapa de testes de regressão é mais uma peça fundamental para garantia da qualidade do software, essa etapa contribui no sentido de mesmo havendo alterações no software, as funcionalidades que já foram verificadas através da rotina de testes não sejam influenciadas pelas correções de outra funcionalidade, assim praticamente eliminando os possíveis defeitos que esses venham a ter.

4 RETORNO DO INVESTIMENTO (ROI)

O investimento é fundamental dentro de uma empresa, eles estão ligados a expansão, manutenção e a modernização dos negócios. Mas nem sempre investir em algum projeto significa ter retorno. É preciso estar bem ciente dos riscos e analisar todas as variáveis possíveis. Só assim é possível escolher o melhor investimento (PEREIRA; REIMER; ROCHA, 2006).

O conceito de ROI para Palmeira (2004, p. 11) é a “relação entre o lucro gerado e montante do investimento realizado”.

Peters (1979) define ROI como o índice que mostra quanto foi investido em determinado empreendimento com relação à recompensa gerada pelo investimento ao longo do tempo.

Quando se comenta sobre retorno do investimento logo se vem à questão do risco, e de acordo com Gitman (2001), risco é a chance de perda financeira, ou seja, utiliza-se o termo risco quando se tem a incerteza ao se referir à variabilidade de retornos associados a um determinado ativo.

O retorno é definido como o total de ganhos ou perdas ocorridos através de um dado período de tempo, sendo assim é um investimento no qual não se sabe se terá retorno positivo ou negativo (GITMAN, 2001). É dessa forma que é necessário se ter um planejamento sobre qualquer situação que poderá ter um impacto sobre o lucro da empresa.

As empresas de hoje em dia estão sempre em busca de lucratividade, qualquer investimento tem a necessidade de ser tratado em forma de projeto, esse tem que ser bem definido para evitar que todo o investimento se torne em gasto sem retorno (WOILER; MATHIAS, 1996). Dessa forma o ROI deve estar presente em qualquer tipo de projeto, afinal de contas os projetos precisam dar a empresa retorno do investimento aplicado, ou não faz qualquer sentido levá-lo adiante.

As empresas, seja ela de pequeno, médio ou grande porte, necessitam ter a noção básica do ROI, pois ele faz parte do planejamento dos projetos. É a partir dele que é possível medir até que ponto determinado projeto trará retorno para a empresa, nesse caso é possível abortar determinado projeto que em breve não trará mais retorno do investimento e sim apenas custos em manter o mesmo.

4.1 ROI EM TESTE DE SOFTWARE

Um defeito no software é definido por Myers (2004) como uma má interpretação dos requisitos que faz com que a especificação do sistema seja feito de forma incorreta.

O custo para correção do software aumenta de acordo com o tempo que esse defeito leva para ser encontrado. Os custos para encontrar e corrigir os defeitos do software aumenta exponencialmente na proporção que o trabalho evolui através das fases do projeto (MYERS, 2004).

Rios e Moreira (2006) chegaram a algumas conclusões quanto ao teste de software e seus defeitos, definidos por eles da seguinte forma:

a) Enquanto está sendo desenvolvido o software já é aplicado o teste de caixa branca, assim pode-se definir que quanto melhor a qualidade destes testes, menores serão os custos de manutenção;
b) Outro ponto impactante é quando solicitada manutenção por parte do usuário no software, essas são fontes de novos defeitos, até mesmo podendo ocasionar defeitos em funcionalidade nem mesmo alteradas. Para identificar essas situações aplica-se o teste de regressão completo, dessa forma evita-se de testar apenas as modificações realizadas;
c) Quando mais especializada for à equipe de testes, maior a qualidade do software e menor os custos.

Como foi dito anteriormente, o processo de teste bem definido e estruturado na empresa reduz uma grande quantia dos defeitos nos software, isso enquanto o software ainda não foi liberado ao cliente, reduzindo também o custo com possíveis correções. Só que ainda existem muitos que acreditam que equipe de teste só traz atraso aos projetos e aumento nos custos, deste modo, vir a criar resistência entre os responsáveis pelo projeto de desenvolvimento do software com a equipe de testes. Sendo assim os processos de teste deveria ter seu tempo reduzido para que não crie esse ambiente entre as equipes (RIOS; MOREIRA, 2006).

A seguir no quadro 1, Rios e Moreira (2006) definiram como seria o investimento e a qualidade do software se ela conta-se com uma equipe de testes bem definida.

Quadro 1 - Custo de correção a defeitos de software

	PROCESSO SEM ESTRUTURA FORMAL PARA TESTES	PROCESSO COM ESTRUTURA FOMAL PARA TESTES (MANUAL)
HORAS DO PROJETO	10000	10000
ERROS ENCONTRADOS	1000	1000
INVESTIMENTO EM TESTES		
Pessoal	0,00	90.000,00
Infra-estrutura	0,00	16.000,00
TOTAL INVESTIMENTO	0,00	106.000,00
DESENVOLVIMENTO/TESTE		
Defeitos encontrados	250	250
Custo de correção	2.500,00	2.500,00
TESTES (*)		
Defeitos encontrados	0	350
Custo de correção	0,00	35.000,00
MANUTENÇÃO (**)		
Defeitos encontrados	750	400
Custo de correção	750.000,00	400.000,00
CUSTO DA QUALIDADE	752.500,00	543.500,00
RETORNO		
(*) Testes realizados pela equipe de testes **(**) Defeitos encontrados após a implantação afetando os usuários**		

Fonte: RIOS; MOREIRA (2006, p.19)

Conforme quadro 1, Essa está dividida em três colunas, na primeira foi definido os itens no qual se tem despesas com relação a elaboração do software, na segunda coluna está as despesas da empresa, levando em consideração que a mesma não possuía equipe de testes formal, já a terceira coluna mostra a redução dos custos caso esse mesmo projeto agora passado pela equipe de testes formal. Apenas com essa demonstração da tabela 1 já é possível entender o quanto se faz necessário um software ser submetido a uma verificação completa.

Barros (2008) definiu o cálculo do ROI em testes de software da seguinte forma:

$$ROI = ((CDst - CDct) – CT) / CT$$

Onde:

a) CDst = Custo de defeitos sem testes;

b) CDct = Custo de defeitos com testes;

c) CT = Custo de testes.

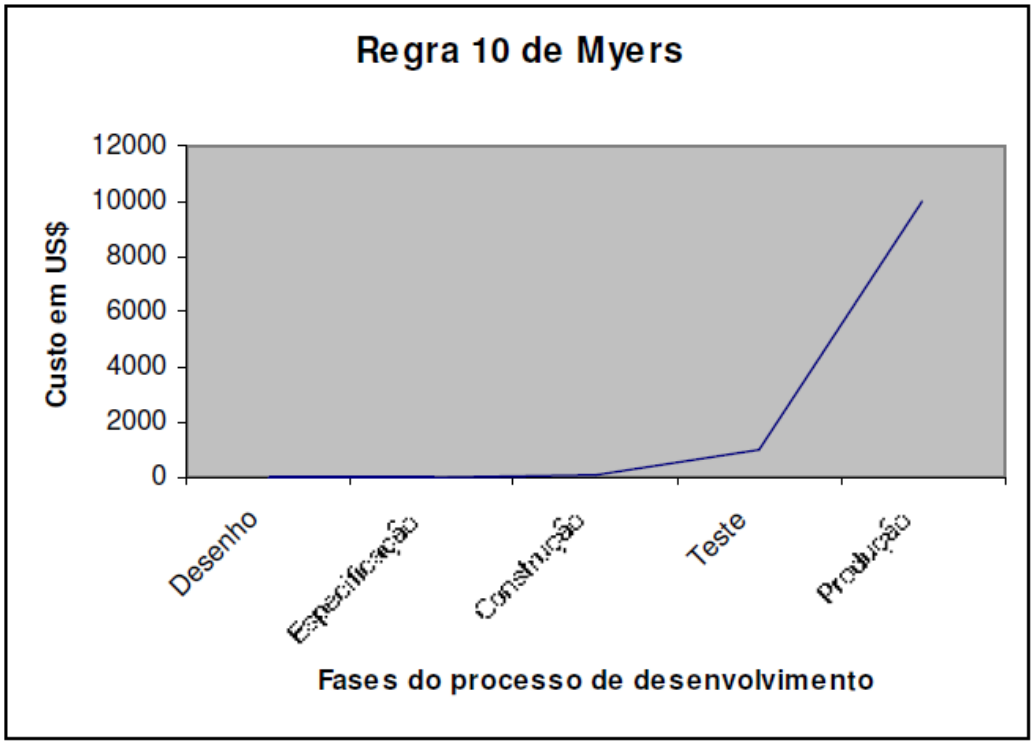
Após a aplicação dos valores nessa fórmula, pode-se dizer que o ROI será positivo se:

$$CDst > CT + CDct$$

Segundo Myers (2004), quanto mais cedo se inicia os testes menores se tornam os custos com correções que terá no decorrer do projeto. Enquanto o software está em sua fase inicial e é encontrado defeito, o custo total para correção desses pode se tornar 1.000 ou mais vezes mais barato do que se o mesmo fosse encontrado apenas quando o sistema já estive-se em ambiente de produção.

É possível analisar na figura 3 o quanto se torna caro a correção de defeitos, ela é conhecida pela regra 10 de Myers, onde os custos aumentam de forma exponencialmente à medida que o projeto de desenvolvimento avança dentro do ciclo de vida do sistema, assim a cada etapa esse custo aumenta de forma impactante.

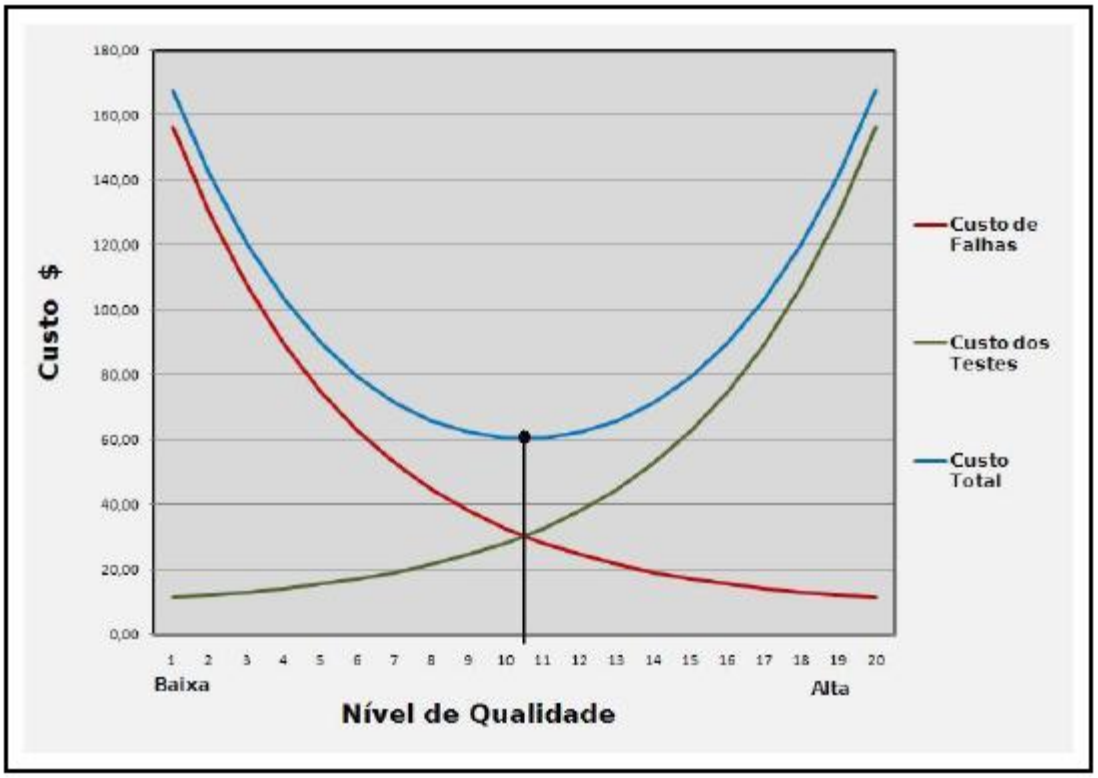
Figura 3 - Regra 10 de Myers



RIOS; MOREIRA (2006, p.29)

A figura 4 mostra o ponto ideal de parada dos testes de software. Se o custo dos defeitos que possam vir a ocorrer num ambiente de produção não é alto, então não compensa gastar mais dinheiro tentado evitá-los. Se levar em consideração o nível critico do software, logo terá que ser gasto mais dinheiro para garantir a qualidade do produto, ou seja, o que define a quantidade de testes é o nível do negócio (BASTOS; CRISTALLI; MOREIRA; RIOS, 2006).

Figura 4 - Representação gráfica do ROI



Fonte: BARROS (2008)

Barros (2008) sugere que, para o cálculo do ROI as variáveis a serem consideradas são:

a) custos diretos
- suporte,
- manutenção,
- visita ao cliente;

b) custos Indiretos:
- desgaste com o cliente,
- desgaste com a equipe,
- imagem da organização;

c) custos com testes

- despesas com ferramentas, ou seja, automação de testes, controle de versão, entre outros,
- despesas com treinamento,
- despesas com mão de obra,
- despesas com equipamentos;

É com todos esses números que tem necessidade de ser estabelecido de alguma forma um método em que possa corrigir todos os defeitos do software, não necessariamente todos, mas sim aqueles que de uma forma e outra são essenciais para o bom funcionamento do sistema. Dessa forma tem que ser estabelecido a partir de todos os testes quais tem maior prioridade a ser executado, então terá que ser estabelecido prioridades para todos os testes e após aplicar a formula de ROI para identificar quais desses serão testados.

4.2 COMO CALCULAR ROI

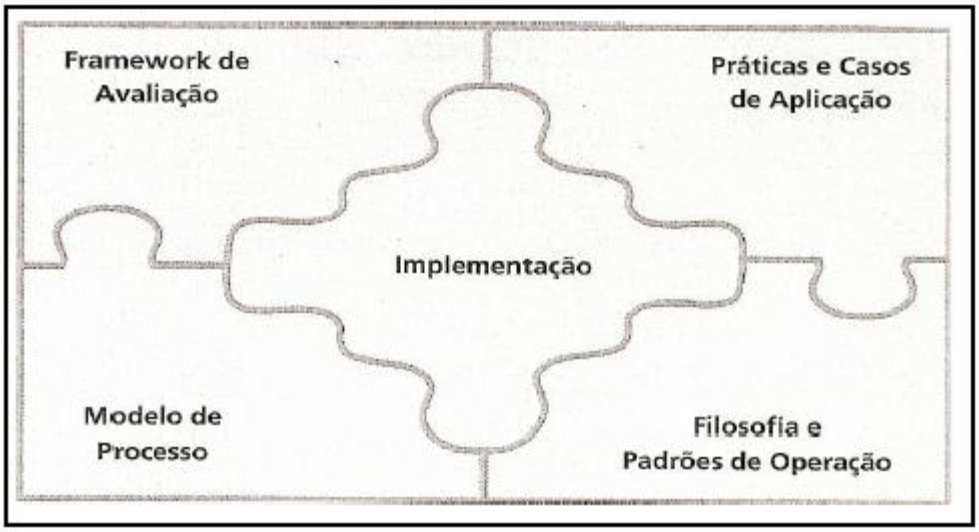
A atividade de medir é de necessidade para os gerentes de projeto, pois assim ele conhece o ambiente em que está trabalhando. Somente usando uma medição é possível identificar os processos que não estão trazendo lucratividade para a empresa e fazer algo para melhoria desse projeto. É uma maneira de eliminar barreiras, corrigir os defeitos, antes mesmo de o produto ser entregue ao cliente (PRESSMAN, 2002).

De forma simples, pode-se dizer que o ROI é apenas um cálculo financeiro, Porém desenvolver o ROI é um processo muito mais detalhado e abrangente do que se possa imaginar.

Não existe projeto financeiro que compense investir sem que haja um retorno, sendo que para poder dar andamento a um projeto é necessário a aplicação de investimento. É com essa teoria que o ROI aparece, justamente para medir se tal negócio é atrativo ou não, ou seja, qual o percentual de retorno. Esse conceito é simples e de fácil entendimento, só que não era utilizado pela grande maioria das empresas. Atualmente o ROI é exigido por qualquer investidor antes de dar andamento em qualquer projeto, isso é o mínimo que se espera de um empreendimento bem planejado (PEREIRA; REIMER; ROCHA, 2006).

O processo de ROI deve ser encarado como um quebra cabeça com diversas partes que devem ser organizadas e unidas, essas peças estão definidas na figura 5.

Figura 5 - Elementos do processo ROI



Fonte: PHILLIPS (2007)

4.2.1 Framework de Avaliação

Existem seis tipos de dados usados no processo de ROI, estes são independentes, mas seus significados possuem ligações. Phillips (2007) os define da seguinte maneira:

a) **reação/satisfação em relação ao projeto**: Essa informação é obtida através de questionários e pesquisas genéricas. Através desses dados é possível identificar o nível de satisfação dos participantes do projeto, mas isso não quer dizer que estes adquiriram o conhecimento para a execução do projeto;

b) **aprendizagem das habilidades e do conhecimento necessário ao sucesso do projeto**: mede o que os participantes do projeto aprenderam e devem aprender durante a implementação do mesmo. A verificação da aprendizagem ajuda a garantir que os participantes sabem como fazer para garantir que o projeto em questão seja um sucesso;

c) **aplicação e implementação do projeto**: é a forma de medir para determinar se o projeto foi ou não implementado com sucesso. Para

isso, é medido todas as etapas, ações, tarefas e processos que envolvem o projeto;

d) **impacto no negocio e consequências do projeto**: esta é à medida que se foca totalmente nos resultados reais do negócio, abrangendo os quesitos de saídas, qualidade, custos, desempenho, tempo e satisfação do cliente;
e) **ROI**: é a avaliação dos benefícios com a comparação de custos do projeto. Pode ser expresso de várias maneiras, só que a forma que mais é apresentada é através da porcentagem ou uma relação custo/beneficio;
f) **benefícios intangíveis**: Além de benefícios tangíveis, os projetos de TI na sua maioria produzem benefícios intangíveis, não monetários.

4.2.2 Modelo de Processo

A primeira etapa do processo do ROI é descrita por Phillips (2007) é, planejamento da avaliação, essa foca nas questões criticas de planejamentos como o desenvolvimento de objetivos apropriados para as iniciativas.

Tendo já definido os objetivos, o próximo passo é o plano de coleta de dados, serão indicados os tipos de dados que deverão ser coletados, o método que será utilizado para a coleta, as origens dos dados, o cronograma e as responsabilidade de coleta (PHILLIPS, 2007).

Logo em seguida, deve-se definir o plano de analise de ROI, onde será determinado como os dados serão convertidos em valores, as categorias apropriadas de custo e as medidas intangíveis (PHILLIPS, 2007).

Phillips (2007) destaca que a coleta dos dados é uma atividade importante para garantir que sejam feitas as adaptações necessárias para manter o projeto no caminho certo. Os dados no qual for coletados após o projeto serão comparados com os dados coletados antes da execução do projeto.

Logo abaixo estão definidos alguns métodos que podem ser utilizados para a coleta de dados:

a) pesquisas de acompanhamento, questionários de satisfação;
b) observação no trabalho para levantar a aplicação e o uso;
c) testes, avaliações e ou auto-avaliações para medir o aprendizado;

d) entrevistas para medir a reação;
e) determinar o grau de aplicação do projeto no ambiente de trabalho;
f) criação de planos de ação que mostrem o processo da implementação no ambiente de trabalho e o impacto obtido;
g) monitoração do desempenho do negócio focando em melhorias.

O isolamento de defeitos do projeto é uma atividade importante, pois os dados de desempenho são influenciados por muitos fatores após a implementação de um projeto. Segue algumas estratégias que são utilizadas por Phillips (2007) para o isolamento dos defeitos:

a) comparação de um grupo de participantes da implementação com um grupo que não participa da mesma, isolando o impacto do projeto;
b) utilizações de linhas de tendência para prever os valores do impacto e após um projeto estas projeções são comparadas com valores reais;
c) estimativas de gerentes e supervisores sobre o impacto de um projeto nas medidas dos resultados;
d) pesquisas e estudos para fornecer informações sobre o impacto do projeto;
e) estimativas de especialistas sobre o impacto nas variáveis de desempenho;
f) informações de clientes sobre a influência de suas decisões para utilizar um produto ou serviço.

Para que possa ser calculado o ROI, é necessário transformar os dados em valores monetários, para isso cada unidade de dado recebe um valor. Abaixo está relacionado uma lista com estratégias de conversão de dados, a estratégia escolhida irá depender do tipo de dados a ser convertida (PHILLIPS, 2007):

a) serviços prestados são convertidos em contribuição ao lucro e relatado como um valor padrão;
b) o custo de uma medida de qualidade é calculado e relatado como valor padrão;
c) tempo economizado por funcionários é convertido em compensação integral;
d) custos comuns à organização ou valores de medidas, por exemplo, prevenções de tempo perdido em acidentes, são usadas quando estão disponíveis;

e) estimativas de medidas são feitas por profissionais experientes;
f) tempo de resposta da rede e bases de dados externas possuem valores próximos;
g) custos e valores de categorias de dados estimados pelos participantes quando há conflitos entre o grupo de trabalho;
h) gerentes ou superiores ao estarem aptos estimam os valores e custos;
i) valor do dado estimado pela equipe de funcionários.

Na formula do ROI é necessário ter o retorno liquido, e como denominador está relacionado à soma de todos os custos, listados assim por Hoffmeister (2009):

a) analises, avaliações iniciais e durante o ciclo de vida do programa;
b) custo de compra/aquisição;
c) custo de elaboração do projeto;
d) tempo dos participantes do projeto;
e) materiais utilizados no projeto;
f) custos da aplicação e implementação do projeto;
g) custos de manutenção e monitoração;
h) custos de administração;
i) custos de documentação.

Para que se chegue ao valor do ROI, é necessário ter os benefícios líquidos divididos pelos custos. A fórmula para calcular o retorno do investimento é definida logo abaixo (PHILLIPS, 2007):

Figura 6 - Cálculo do ROI

$$ROI = \frac{Benefícios\ Liquidos \times 100}{Custos}$$

Fonte: PHILLIPS (2007)

4.2.3 Casos de aplicação e Relatório

Uma etapa final do processo ROI é a elaboração do estudo de impacto para documentar os resultados obtidos pelo projeto e comunicá-los ao grupo de

analise do ROI. Esse estudo é a apresentação do processo básico usado na geração das seis categorias de dados (PHILLIPS, 2007).

Essa geração de relatórios é um importante método de distribuição das informações relacionadas ao projeto para compreensão dos clientes. O ponto desta etapa é ser possível analisar em audiências detalhadas durante o planejamento da avaliação, assim podendo criar relatórios apropriados para atender a necessidade do projeto (BRAUN, 2008).

4.2.4 Guias de Operação

Phillips (2007) criou alguns guias para ser usado como padrão na implementação do ROI, isso para garantir que o ROI seja desenvolvido da mesma maneira por todos:

a) ao por em prática avaliações de níveis elevados, os dados de níveis inferiores devem ser obtidos;
b) ao planejar uma avaliação em um nível superior, as avaliações dos níveis anteriores não precisam ser tão abrangentes;
c) ao obter e analisar os dados, as fontes devem ser seguras, ou seja, possuir total credibilidade;
d) ao analisar os dados, a alternativa mais conservadora para o calculo dos resultados é a melhor opção para analise dos dados;
e) utilizar ao menos um método de isolamento de defeitos;
f) pouca ou alguma melhoria terá ocorrido se dados de melhorias não estiverem disponíveis;
g) estimativas de melhoria devem ser ajustadas para possíveis erros;
h) não utilizar itens de dados externos para cálculo do ROI;
i) custos devem ser todos considerados para analisar o ROI;
j) medidas intangíveis devem ser propositalmente não transformadas em valores não monetários.

4.2.5 Implementação

Toda mudança ou novidade muitas vezes apresentam casos de resistências na equipe. Parte dessa resistência estará baseada em barreiras reais, já

outra parte será baseada em má compreensão ou até mesmo em problemas percebidos que podem nem existir. Para que isso seja superado a implementação do processo do ROI deve ser criada de forma cuidadosa e metódica.

A implementação possui pontos como atribuições de responsabilidades, desenvolvimento de planos e metas, organização de ambientes e preparação dos participantes do projeto.

4.3 DIFICULDADE DE MEDIR ROI

Para Braun (2008) existe uma grande dificuldade de medir ROI na área da TI, principalmente se aplicado nos testes de software, ele destaca que se avaliarem os aspectos financeiros nem sempre é possível identificar o real valor de um processo. Na verdade a dificuldade está na identificação de métricas que não estejam somente vinculadas a resultados financeiros, um exemplo simples é a relação entre o investimento em processos de teste de software, onde nesse caso não se deve analisar apenas o retorno financeiro, tem que ser levado em consideração também a questão do nível de qualidade do sistema em questão.

Para chegar a um bom resultado do ROI, primeiramente se devem equilibrar várias questões, dentre elas se destacam a simplicidade, credibilidade e confiabilidade. Definitivamente o ROI não é um processo fácil de ser usado, pois vários fatores influenciam em seu cálculo, nesse caso podemos destacar a constante mudança tecnológica que a TI sofre. Saber o objetivo do projeto é um passo importante para medir o ROI, pois assim poderão ser priorizadas metas, além de já poder saber quais problemas será encontrado no decorrer (PHILLIPS, 2007).

Desse modo, a grande dificuldade de encontrar o ponto de parada dos testes será determinar a partir de todos os testes qual será realmente testado e qual será executado em outro momento, sem que o software tenha perda de qualidade.

5 TRABALHOS RELACIONADOS

Nesse capitulo será listado alguns trabalhos que estão diretamente ou indiretamente ligados a esse trabalho que está sendo desenvolvido.

5.1 MODELO PARA MEDIÇÃO DO ROI EM PROCESSOS DE TESTE DE SOFTWARE

Este trabalho de conclusão de curso foi proposto pela Acadêmica: Lusiane Braun do curso de Ciência da Computação, pelo Centro Universitário Feevale, no ano de 2008, afim de obtenção do titulo de Bacharel, sob orientação do prof. Msc. Eduardo Pretz.

A proposta do trabalho foi definir um modelo para medir o retorno de investimento em processos de teste de software. Esse modelo define como selecionar métricas adequadas seguindo o método Goal-Question-Metrics (GQM). O modelo desenvolvido nesse trabalho define também como calcular o ROI e apresenta um método para analise dos resultados desse cálculo. Então será desenvolvido um framework que apoiará as organizações de desenvolvimento na difícil tarefa de avaliar o ROI sobre os processos de teste.

5.2 UM MÉTODO DE TESTE FUNCIONAL PARA VERIFICAÇÃO DE COMPONENTES

Essa dissertação de conclusão de Pós-Graduação foi proposta pela Acadêmica: Carina Machado de Farias do curso de Ciência da Computação, pela Universidade Federal de Campina Grande - UFCG, no ano de 2003, afim de obtenção do grau de mestre em Informática, sob orientação do prof. Msc. Patricia Duarte de Lima Machado.

O trabalho proposto teve como objetivo propor um método de teste funcional aplicável a componentes do software. O método de teste proposto é apresentado dentro de um processo de desenvolvimento de especificações UML (Unified Modelling Language). Nesse trabalho também foi desenvolvido um estudo de caso.

5.3 LINHA DE PRODUTOS DE TESTES BASEADOS EM MODELOS

Essa dissertação de conclusão de curso foi proposta pelo Acadêmico: Alex Mulattieri Suarez Orozco do curso de Ciência da Computação, pela Pontificia Universidade Católica do Rio Grande do Sul - PUCRS, no ano de 2009, afim de obtenção do grau de mestre em Ciência da Computação, sob orientação do prof. Dr. Avelino F. Zorzo.

Hoje em dia, existem diversas formas de realizar o teste de software baseado em modelos. Uma forma de integrar estas possibilidades é através da aplicação dos conceitos de linha de produtos de software. O trabalho em questão propõe a realização de uma arquitetura de linha de produtos de software para integrar as diferentes técnicas de testes baseados em modelos.

6 TRABALHO DESENVOLVIDO

Nesse trabalho foi focado nos processos de teste de software e em retorno de investimento, assim podendo entender o funcionamento dele e de alguma forma buscar algo para melhorar e reduzir os tempos de teste sem perda de qualidade e garantia do retorno de investimento.

É com esse intuito que foi elaborado nesse trabalho um modelo de gerenciamento dos casos de teste, dessa forma será possível fazer um planejamento dos testes antes de executar, sendo que durante esse planejamento será possível saber até que ponto efetuar os testes, isso devido à elaboração da formula do ROI, que mede o ponto de parada dos testes.

7 METODOLOGIA

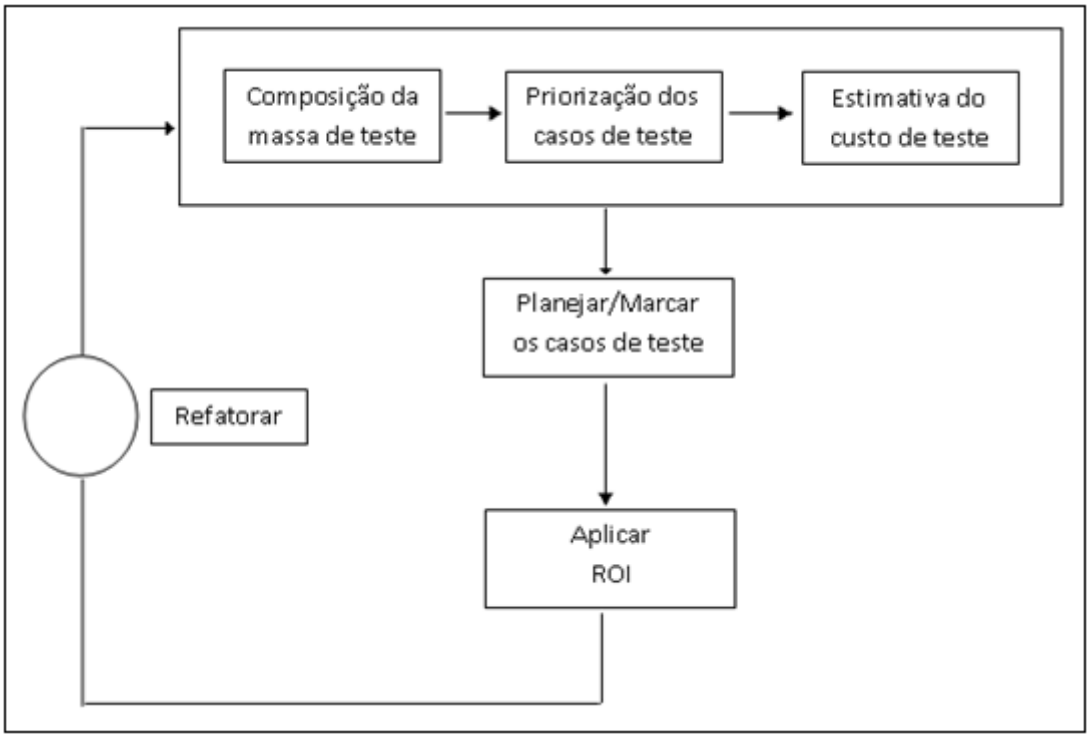
Para a elaboração do presente trabalho, foi necessário criar um modelo de gerenciamento dos casos de teste, assim podendo aplicar a fórmula do ROI a fim de identificar a quantidade de testes necessária para garantir a qualidade do software, com total foco nos custos dos testes, mas não menosprezando a qualidade do produto.

Sendo assim foi necessário realizar o estudo sobre algum software a fim de ter uma massa de dados consistente, definindo essa como a primeira etapa. Para isso foi utilizado o software *Ieducar, por se tratar de um software livre onde* está disponível para todos no site software livre do governo. Com o software funcionando começou os estudos para poder entender o seu objetivo e as regras de negócio.

Terminando essa primeira etapa, o fluxo dos trabalhos seguiu conforme a figura 7, esse é o modelo proposto a seguir para determinar o ROI sobre teste de software.

Nos próximos subcapítulos será detalhado o que deve ser feito em cada item da figura 7, iniciando pelo processo da composição dos testes.

Figura 7 - Processos para aplicação do ROI



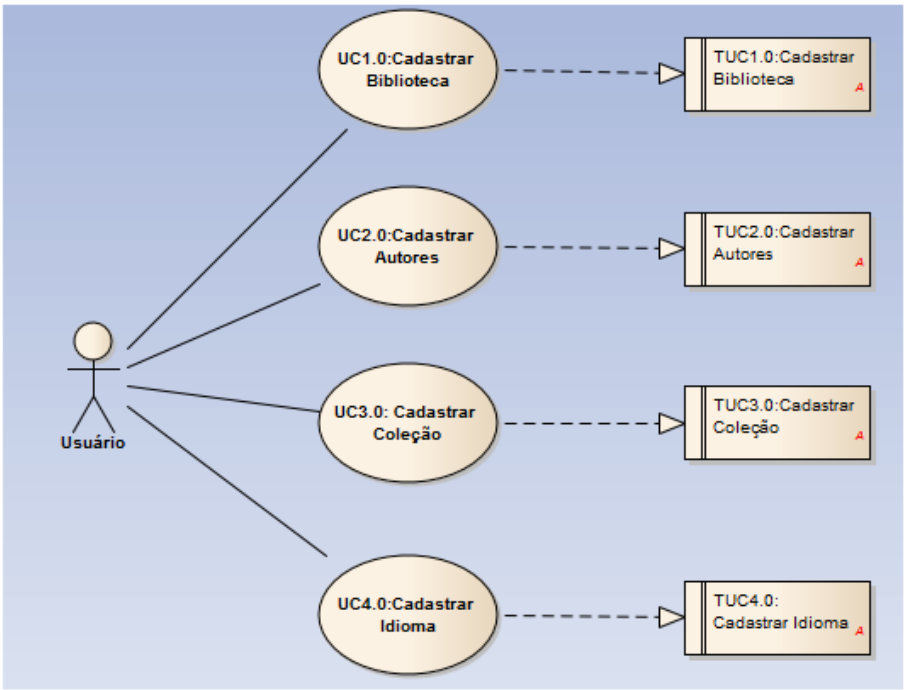
7.1 COMPOSIÇÃO DA MASSA DE TESTE

O primeiro ponto necessário para iniciar o trabalho proposto, foi definir um software que seria utilizado para realizar uma bateria de testes, pois além de definir a fórmula do ROI, existe a necessidade de executar a fórmula para obter os resultados e avaliar os mesmos, assim verificando se os dados utilizados para o ROI tem coerência.

O software escolhido foi o Ieducar, é um sistema para gestão escolar, utilizado pelas escolas municipais, assim reduzindo a necessidade de uso de papel, ele possui cadastro de escolas, alunos, professores, e também existe um módulo para gerenciamento de biblioteca. Foi necessário fazer a instalação desse software, assim que ele estava pronto para o uso, iniciou-se um estudo sobre suas ferramentas, a fim de entender as regras de negocio e a sua utilização. Foi escolhido o módulo biblioteca, e de todos os seus requisitos funcionais, foi gerado casos de uso para quatro funções, conforme figura 8, que são elas:

a) UC1.0 Cadastrar Biblioteca
b) UC2.0 Cadastrar Autor
c) UC3.0 Cadastrar coleção
d) UC4.0 Cadastrar Idioma

Figura 8 - Caso de uso software IEducar



Tendo os casos de uso definidos, iniciou o processo de elaboração da massa de testes, para cada caso de uso, foram criados diversos casos de teste, tendo como identificação para cada caso a nomenclatura TC1.0.

A geração dos casos de testes foi feita de forma manual, a partir do conhecimento de testes e do software utilizado. Assim para cada caso de uso foi gerados desde os testes menos críticos, até os mais críticos. Sendo assim, definimos como massa de teste, todos os casos de testes gerados para os requisitos funcionais levantados.

Os casos de teste foram formados com as seguintes características:

a) **titulo:** Identificação e nome atribuído ao teste, forma rápida e clara de se entender o objetivo do mesmo;
b) **descrição:** Descrição mais detalhada do que se deve fazer e o que vai acontecer;
c) **entrada:** Nessa etapa são fornecidos ao testador os dados de entrada no sistema;
d) **saída:** Descreve o que deve acontecer após o software receber a entrada e processar a mesma.

Com a massa de testes bem detalhada e pronta, antes de executa-las, iniciou-se o processo de priorização de cada caso de teste, assim teria uma ordem para testar, do teste com maior prioridade até o com menor.

7.2 PRIORIZAÇÃO DOS CASOS DE TESTE

A priorização se faz necessário para saber por onde iniciar o teste, ele serve como uma ordenação, onde o teste com maior prioridade é o primeiro a ser executado até chegar ao menos prioritário.

Existem diversas técnicas de priorização dos casos de teste, nesse trabalho foi aplicada a técnica GUT, onde são atribuídas três notas para cada caso de teste, as notas vão de 1 a 5, sendo que quanto maior o valor atribuído maior será a prioridade no final. Primeiro se faz a avaliação da Gravidade, depois da urgência e por fim a tendência, para esses três pontos é atribuído um valor,

a) **gravidade:** Impacto do problema na empresa, efeitos que vão surgir com o tempo se não for solucionado;
b) **urgência:** Tempo que se tem para resolver o problema;

c) **tendência:** crescimento do problema.

Tendo as notas, essas são somadas, e divido por três, no final cada teste terá apenas uma nota, então se ordena pela nota final, sendo essa a sua priorização e ordenação de execução dos testes.

Para fazer a priorização dos testes, foi solicitado a ajudar de dois profissionais da área de teste, cada um fez sua analise e realizou a atribuição de notas, ao final foi feito a média das priorizações.

Aplicando essa técnica podemos afirmar que os testes já estão priorizados. Essa priorização será importante para o momento do cálculo do ROI, onde serão marcados os testes com maior prioridade para ser executado, assim podendo ficar de fora da bateria de teste os com menor priorização, onde nos dá ao entender que são testes que não a necessidade de se executar.

7.3 ESTIMATIVA DO CUSTO DE TESTE

Para a execução da fórmula do ROI, é necessário fazer uma estimativa de custo do teste, esse é um elemento que será utilizado na fórmula do ROI, é fundamental para o resultado final.

Caso não se tenha um histórico de testes executados, para a primeira vez pode ser estimado um valor. Para quem já possui uma massa de teste já executada e com os tempos de teste, a estimativa pode ser feita sobre esses valores, assim tornando o resultado final mais próximo da realidade. O ideal é que a massa de teste já tenha sido executada pelo menos uma vez, para ter dados consistentes, caso a massa de teste tenha sido executado mais de uma vez, a recomendação é que seja realizada a média dos tempos de execução de cada caso de teste, assim as probabilidades do valor do ROI ser mais próximo da realidade.

Para se estimar o custo do teste, é necessário ter em reais o valor da hora trabalhada pelo testador, pelo analista e pelo desenvolvimento. O tempo de teste deve ser transformado totalmente em horas, no caso de um teste levou trinta minutos para executá-lo, deve-se dividir o trinta por sessenta, assim tendo no final, 0,5 horas de teste. Por fim é feita a multiplicação da quantidade de horas trabalhadas pelo custo da hora, resultando um valor total de R$5,00 reais. Esses cinco reis é a estimativa do preço se for executar um caso de teste, levando em consideração que no primeiro teste foi levado 30 minutos, foi baseado em um teste

já executado outra vez, isso não significa que na próxima vez que for necessária executar o mesmo teste, o tempo gasto será o mesmo, por isso que se usa o termo estimativo.

Os seguintes valores abaixo também foram estimados para a demonstração dos resultados obtidos no capitulo 8, valores da hora de trabalho de cada envolvido:

a) **custo hora analista (R$)**: 10,00;

b) **custo hora desenvolvedor (R$)**: 14,00;

c) **custo hora teste (R$)**: 8,00.

Tempo médio da correção dos defeitos encontrados no teste:

a) **tempo médio analise (Min.)**: 7;

b) **tempo médio desenvolvimento (Min.)**: 25;

c) **tempo médio reteste (Min.)**: 8.

Para a correção dos defeitos externos, o tempo médio foram considerados os valores acima multiplicados por 2.

7.4 PLANEJAR OS CASOS DE TESTE

No planejamento dos casos de teste são definidos quais deles serão executados. O planejamento inicia com a definição das variáveis definidas abaixo:

a) **quantidade de bugs:** é a quantidade de bugs encontrados durante a execução dos testes. Esses bugs serão considerados a média de bugs que o teste encontrou nos testes anteriores;

b) **quantidade de bugs externos**: é a quantidade de bugs que o teste não encontrou, e esses foram encontrados no ambiente de produção. Deve ser realizada a média dos testes anteriores.

Com os valores acima preenchidos na planilha com o nome ROI, o cálculo já foi iniciado, o próximo passo é informar quais testes serão executados.

Na planilha que possui todos os casos de teste, foi elaborada uma coluna chamada de planejamento previsto, essa serve para definir quais testes será executada, a marcação é com um ‘X’, conforme vai marcando os testes, automaticamente o ROI é calculado, então o responsável pela definição dos testes deve ficar acompanhando os resultados, pois o valor final do ROI deve ser positivo,

a partir do momento que esse se tornar negativo os custos serão maiores que o retorno.

7.5 APLICAR O ROI

Para o cálculo do ROI serão consideradas várias variáveis, o cálculo inicia-se no momento em que são planejados os casos de teste. Assim conforme são feitas as marcações a fórmula é executada e mostra o valor na planilha. O ROI será calculado através da fórmula definida abaixo:

$$ROI = ((CDst - CDct) – CT) / CT$$

Onde:

a) **CDst** = Custo de defeitos sem testes;

b) **CDct** = Custo de defeitos com testes;

c) **CT** = Custo de testes.

Porém para chegar aos valores de cada variável foi necessário um estudo para encontrar um valor próximo da realidade. Abaixo está à definição de como chegar aos valores das variáveis acima:

a) **CDct** = CT + CR + CCORDEV + CA;

b) **CDst** = CA + CCORDEV;

c) **CT** = CT.

Onde, as variáveis são definidas assim:

a) **CT**: Média das horas de teste * Custo da hora do teste em reais;

b) **CR**: Média das horas de reteste * Custo da hora do teste em reais;

c) **CCORDEV**: Média das horas de correção do desenvolvimento * Custo da hora do desenvolvimento em reais;

d) **CA**: Média das horas de analise * Custo da hora do analista em reais.

Após a aplicação dos valores na fórmula do ROI, os testes serão planejados até que:

$$CDst > CT + CDct$$

A partir do momento em que o custo do defeito sem teste for menor que o custo do teste mais o custo de defeito com teste, o planejamento deve ser parado e revisto os casos de teste, havendo a necessidade de tirar testes que estavam planejados para a execução, nesse caso é realizado o processo definido no próximo subcapitulo.

O cálculo do ROI foi elaborado de duas formas, uma sobre os custos reais, e outra sobre a regra 10 de Myers, onde Myers define que os custos aumentam exponencialmente de acordo com o ciclo do software, nesse caso foi definido que, o custo de correção para os bugs encontrados pela equipe de teste foi multiplicado por 10, já os bugs externos serão multiplicados por 100.

Como será definido quais testes executar, nesse caso não será executada toda a massa de teste, estamos assumindo o risco de poder ocorrer bugs que o teste não vai encontrar, porém esses bugs não terão grande impacto visto que antes de planejar os testes foi realizada uma priorização, definindo que os testes com maiores riscos serão executados.

7.6 REFATORAR

A refatoração será utilizada quando ocorrer de após a realização do processo de planejamento dos testes, a fórmula do ROI chegue a um valor negativo, pois quando chegar a esse resultado significa que os custos serão maiores que os retornos. Então deve ser realizado um replanejamento dos casos de teste, fazendo novamente a marcação, onde o principal objetivo é não chegar a um valor negativo.

Realizado a refatoração e chegado ao resultado positivo do retorno de investimento, pode-se iniciar os testes.

8 APRESENTAÇÃO E ANÁLISE DOS DADOS

Com a massa de testes do software IEducar elaborada, iniciou-se a execução de cada caso de teste, e após a finalização do teste a planilha e foi alimentada com as informações se teve ou não bug, e o tempo de execução do teste. A partir desses dados foram gerados os dados para a planilha do calculo do ROI, então se iniciou o estudo sobre os resultados do ROI. Na execução de todos os casos de teste não foi encontrado nenhum defeito, isso se da ao fato de ser um software que está a um bom tempo sendo utilizado e também já comercializado, é um software estável.

Conforme figura 9, foi calculado o ROI sobre a realidade, foram executados 55 casos de teste, onde desses nenhum defeito foi reportado pela equipe de testes nem pelo usuário final. O resultado do retorno do investimento foi de -100% sobre os gastos reais e também sobre a regra 10 de Myers, pois todo o custo com o teste foi considerado gasto, e do total desse montante nada será recuperado.

Figura 9 - ROI sobre teste sem defeito

DADOS DOS TESTES	
Quantidade de TC:	55
Tempo estimado de teste p/ cada TC (Hora)	0,13
Quantidade de Bugs	0
Quantidade de Bugs Externos	0

CALCULO ROI	
Sobre Gastos Reais	-100%
Sobre regra 10 Myers	-100%

CDst		CT + CDct		Total
R$	-	R$ 58,67	-R$	58,67
R$	-	R$ 58,67	-R$	58,67

Na segunda situação, foi necessário simular os valores dos defeitos, então foram analisados os dados do retorno de investimento considerando que o teste reportou 20 defeitos, e os mesmos 20 defeitos foram encontrados em ambiente de produção. Conforme figura 10, o lucro foi de R$ 7.061,33. Se considerar a regra

10 de Myers os resultados nos proporcionaram uma margem de lucro bem mais elevada, tendo um retorno de R$ 3.263.141,33.

Figura 10 - ROI sobre defeito em teste

DADOS DOS TESTES	
Quantidade de TC:	55
Tempo estimado de teste p/ cada TC (Hora)	0,13
Quantidade de Bugs	20
Quantidade de Bugs Externos	20

CALCULO ROI	
Sobre Gastos Reais	12036%
Sobre regra 10 Myers	5562173%

CDst	CT + CDct	Total
R$ 16.800,00	R$ 9.738,67	R$ 7.061,33
R$ 3.360.000,00	R$ 96.858,67	R$ 3.263.141,33

Na terceira situação simulada, foram estimados 20 defeitos corrigidos no teste, e 25 encontrados em ambiente de produção. Nesse caso apenas 5 defeitos não foram reportados pela equipe de teste, e mesmo assim os lucros são maiores que as despesas, conforme detalha a figura 11. Os lucros foram de R$ 6.421,33, sobre os gastos reais, e de R$ 4.054.741,33 considerando a regra 10 de Myers. Sendo assim, poderia ter sido marcado mais casos de teste para executar.

Figura 11 - ROI em defeitos externos

DADOS DOS TESTES	
Quantidade de TC:	55
Tempo estimado de teste p/ cada TC (Hora)	0,13
Quantidade de Bugs	20
Quantidade de Bugs Externos	25

CALCULO ROI	
Sobre Gastos Reais	10945%
Sobre regra 10 Myers	6911491%

CDst	CT + CDct	Total
R$ 21.000,00	R$ 14.578,67	R$ 6.421,33
R$ 4.200.000,00	R$ 145.258,67	R$ 4.054.741,33

Na figura 12 estão explanados os custos que foram levados em consideração para obter os valores do ROI da figura 11. Na coluna CT, está listado o custo do teste, na coluna CDST está detalhado o custo para correção de um defeito se não existi-se o teste na empresa, já na coluna CDCT são os valores de correção dos defeitos considerando que a empresa possui uma equipe de testes.

Figura 12 - Valores para calculo do ROI

	CT	CDST	CDCT
Quant TC:	55	0	0
Tempo Estimado:	7,33	0	0,00
Custo do Teste:	**R$ 58,67**	R$ -	R$ -
Quant Analise	-	-	20,00
Tempo Analise	-	-	140,00
Custo Analise:	**R$ -**	**R$ -**	**R$ 1.400,00**
Quant BUG	-	-	20,00
Tempo BUG FIX	-	-	500,00
Custo BUG FIX:	**R$ -**	**R$ -**	**R$ 7.000,00**
Quant Reteste:	-	-	20,00
Tempo Estimado:	-	-	160,00
Custo do ReTeste:	**R$ -**	**R$ -**	**R$ 1.280,00**
QuantidadeDefeitos externos:	-	25,00	5,00
Tempo Analise:	-	350,00	70,00
Custo Analise:	**R$ -**	**R$ 3.500,00**	**R$ 700,00**
Tempo Desenvolvimento:	-	1.250,00	250,00
Custo Desenvolvimento:	**R$ -**	**R$ 17.500,00**	**R$ 3.500,00**
Tempo Testes:	-	-	80,00
Custo Testes	**R$ -**	**R$ -**	**R$ 640,00**
Total:	**R$ 58,67**	**R$ 21.000,00**	**R$ 14.520,00**
TotalR10:	**R$ 58,67**	**R$ 4.200.000,00**	**R$ 145.200,00**

Com os valores mencionados acima, podemos perceber que a correção de um defeito mesmo sem medir o tamanho do impacto desse, quase sempre será vantajoso executar testes no software. Agora isso fica mais claro ainda quando é calculado o ROI sobre a regra 10 de Myers, onde ele mede não só o custo para a correção, mas também o impacto ocasionado por esse defeito, desde o cliente que pode ter ficado parado até a mobilização dos funcionários da empresa para encontrar o defeito e resolver a situação o quanto antes.

CONCLUSÃO

Nos dias de hoje as empresas buscam o melhor para seus clientes, no caso de uma empresa de software, o melhor é um sistema pronto par atender a necessidade do cliente e com qualidade. E um dos pontos que definem a qualidade do produto, é o teste, pois é a partir dele que será exercitado o sistema para saber se realmente ele faz o que deve fazer. Hoje em dia as empresas vêm buscando criar o setor de testes, e as que já têm, vem buscando o aperfeiçoamento desses. Porém quando falamos em teste, estamos falando em custo, e muitas vezes é um custo muito elevado, gerando no fim prejuízo ao invés de lucro.

É com esse principio que foi desenvolvido um modelo de testes para realizar o controle dos custos com relação ao investimento realizado. Assim antes de realizar os testes, é possível identificar até quando efetuar os testes, e qual o resultado do retorno do investimento (ROI).

Para atingir o proposto nesse trabalho, foram estudadas e documentadas diversas áreas da engenharia de software, mais especificamente o teste de software, bem como a elaboração dos casos de teste e a execução desses, após o foco se tornou sobre lucros e custos, para definição da fórmula do ROI em teste de software e como aplicar os valores nela.

Como resultado final, foi criado um modelo de teste, essa desenvolvida em planilha eletrônica, a fim de documentar os casos de teste nela para gerar o cálculo do ROI, assim o analista de teste vai saber informar quais testes serão executados, evitando que o investimento traga custos, assim podendo até gerar lucro.

Pôde-se concluir que todos os objetivos propostos no inicio da pesquisa foram atingidos, deixando como resultado final um modelo ágil e fácil de trabalhar, esse ainda pode servir de modelo para que nos trabalhos futuros seja desenvolvida uma aplicação para o estudo realizado nesse trabalho.

REFERÊNCIAS

ANDRADE, Wilkerson de Lucena. **Geração de Casos de Teste de Interação para Aplicações de Celulares.** 2007. 109 f. Dissertação (Mestre) - Ufcg, Compina Grande, 2007. Disponível em: <http://docs.computacao.ufcg.edu.br/posgraduacao/dissertacoes/2007/Dissertacao_WilkersonLucenaAndrade.pdf>. Acesso em: 01 nov. 2011.

BARROS, Hugo. **Retorno de Investimento em Testes de Software. In: Seminário Regional de Teste de Software**. ALATS, 1., 2008, Porto Alegre.

BASTOS, Anderson; CRISTALLI, Ricardo; MOREIRA, Trayahú; RIOS Emerson. **Base de conhecimento em teste de software**. Niterói: Traço & Photo, 2006. 300 p.

BEIZER, Boris. **Black-Box Testing: Techniques for Functional Testing of Software and Systems**. John Wiley & Sons, 1995.

BRAUN, Lusiane. **Modelos para medição do ROI em processo de teste de software.** 2008. 73 f. T (T) - Feevale, Novo Hamburgo, 2008. Disponível em: <http://tconline.feevale.br/tc/files/0001_1490.pdf>. Acesso em: 14 nov. 2011

CARTAXO, Emanuela Gadelha. **Geração de casos de teste funcional para aplicações de celulares**. 2006. 146 f. Dissertação (Mestrado) - Ufcg, Campina Grande, 2006. Disponível em: <http://docs.computacao.ufcg.edu.br/posgraduacao/dissertacoes/2006/Dissertacao_EmanuelaGCartaxo.pdf>. Acesso em: 18 set. 2011.

DELAMARO, Márcio; MALDONADO, José Carlos; JINO, Mario. . **Introdução ao teste de software**. Rio de Janeiro: Elsevier, 2007. 394p. ISBN 9788535226348 (broch.)

DELAMARO, Marcio Eduardo. **Proteum - Um Ambiente de Teste Baseado na Análise de Mutantes**. Dissertação de Mestrado. ICMC / USP, São Carlos, Sp, Outrubro 1993.

DELAMARO, Márcio Eduardo. **Mutação de interface: Um critério de adequação inter-procedimental para o teste de integração**. Tese de Doutorado, Instituto de Física de São Carlos - Universidade de São Paulo, São Carlos, SP, 1997.

FEWSTER, Mark, GRAHAM, Dorothy. **Software Test Automation. Addison-Wesley Professional**; 1st edition, 1999.

GITMAN, Lawrence J. **Princípios de administração financeira**. 2.ed. Porto Alegre: Bookman, 2001. 610 p. ISBN 857307776x

INTHURN, Cândida. **Qualidade & teste de software: engenharia de software, qualidade de software, qualidade de produtos de software, teste de software, formalização do processo de teste, aplicação prática dos testes**. Florianópolis: Visual Books, 2001. 108 p. ISBN 8575020269

JORGENSEN, Paul C.. **Software Testing – a Craftsman Approach**. CRC Press,1995. 272 p.

MYERS, G.; **The Art of Software Testing**, Wiley, New York, 1979.

MYERS, Glenford J. **The Art of Software Testing**. New York: John Wiley & Sons, Second Edition, 2004.

OROZCO, Alex Mulattieri Suarez. **Linha de Produtos de Testes baseados em Modelos.** 2009. 101 f. Dissertação - Pucrs, Porto Alegre, 2009. Disponível em: <http://tede.pucrs.br/tde_arquivos/4/TDE-2009-05-22T055252Z-1937/Publico/412371.pdf>. Acesso em: 15 out. 2011.

PALMEIRA, Cristiana Gomes. **ROI de treinamento: retorno do investimento : sistemas de mensuração**. Rio de Janeiro: Qualitymark, 2004. 95 p. ISBN 8573035072

PETERS, Robert A. **Retorno do investimento: teoria aplicada e novos conceitos.** São Paulo, SP: McGraw-Hill, 1979. 167 p.

PEZZÈ, Mauro; YOUNG, Michal. **Teste e análise de software: processo, princípios e técnicas**. Porto Alegre: Bookman, 2008. 512 p. ISBN 9788577802623 (broch.)

PRESSMAN, R.S.; **Software Engineering: Practitioner's Approach**, McGRAW-HILL, 1992.

PRESSMAN, R. S. **Engenharia de Software**. 5 ed. Rio de Janeiro: McGraw-Hill, 843 p., 2002.

PRESSMAN, R. S. **Software Engineering - A Practitioner's Approach**. 5ª Edição, McGraw Hill, 2000.

PRESSMAN, R. **Software Engineering: Practitioner's Approach**. McGraw-Hill, 7a. Edição, 2009.

PEREIRA, Leonardo Dos Santos;REIMER, Leonardo Ribeiro; ROCHA, Luciano Guimarães. **Plano de Negócio em Tecnologia.** 2006. 56 f. - Ucg, Goiania, 2006. Disponível em: <http://aldeia3.computacao.net/greenstone/collect/trabalho/index/assoc/HASHc840.dir/doc.doc>. Acesso em: 12 nov. 2011.

PHILLIPS, Jack. **Como medir o retorno sobre o investimento: uma missão crítica para o gerente de projetos**. Revista Mundo PM, Curitiba, PR, n. 15, p. 9-17, jun./jul. 2007.

RIOS, Emerson; MOREIRA FILHO, Trayahú R. . **Teste de software.** 2. ed. rev. e ampl Rio de Janeiro: Alta Books, 2006. 222p. ISBN 8576081288 (broch.)

RIOS, Emerson. **Analise de risco em teste de software**. [2007?]. 12 f. Diponivel em: <http://www.riosoft.softex.br/media/artigo_emerson_teste.pdf> Acesso em: 15 set. 2011

SIMONS, Cristian. **Priorização de Casos de Testes de Regressão Usando Amostragem por Perseguição de Defeitos.** 2010. 103 f. Dissertação (Mestrado) - Pucpr, Curitiba, 2010. Disponível em: <http://www.ppgia.pucpr.br/lib/exe/fetch.php?media=dissertacoes:cristian_simons.pdf>. Acesso em: 20 out. 2011.

SILVA, Karina Rocha Gomes da. **Uma Metodologia de Verificação Funcional para Circuitos Digitais.** 2007. 121 f. Tese (Doutorado) - Ufcg, Campina Grande, 2007. Disponível em: <http://lad.dsc.ufcg.edu.br/lad_files/Lad/tese_karina.pdf>. Acesso em: 25 out. 2011.

WOILER, Sansão, MATHIAS, Washington Franco. **Projetos: planejamento, Elaboração e análise**. 2. ed. São Paulo, SP: Atlas, 2008, 292 p.

APÊNDICE(S)

APÊNDICE A – Artigo cientifico.

Modelo de Gerenciamento de Cobertura Funcional de Teste de Software Baseado em Retorno do Investimento (ROI)

Natanael Dagostin Ghellere[1], Gustavo Bisognin[1]

[1]Curso de Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC) – 88806-000– Criciúma – SC – Brasil

nghellere@hotmail.com, gbisog@gmail.com

Abstract. *Nowadays are fundamental running software tests, whether formal or informal. To ensure quality, the ideal is having an expert team on software tests. This final Project studies software engineering, focusing on testing. It`ll be discussed about tests and its techniques, and then achieve the final goal that is ROI calculation. The proposed problem is testing one software ensuring return of the investment. Thereunto was necessary create a tests standard that joins all the tests cases and its prioritization, and then planning if such test case will be or not executed. While the tests planning is being executed, the ROI formula is automatically calculated and shows the result, it will be possible planning until the investment return become positive. When the tests planning reach a negative result, is necessary a test replanning, therefore eliminating some tests until the result became positive again. And so, one of the goals in the project is eliminate some costs on software tests, in order to create an internal planning for choosing what tests will be executed, but always focused in the final product quality.*

Keyword: Software tests, ROI, Software quality, tests management.

Resumo. *Hoje em dia é fundamental a execução de testes em software, seja ele formal ou informal. Para a garantia da qualidade, o ideal é possuir uma equipe especialista em teste de software. Nesse trabalho de conclusão de curso, estuda a engenharia de software, com foco nos testes. Será abordado sobre teste e suas técnicas, para então chegar a um objetivo final que é o cálculo do ROI. O problema no qual foi proposto é testar um software garantindo o retorno de investimento. Para isso foi necessário criar um modelo de testes em que liste todos os casos de teste juntamente com sua priorização, para então fazer um planejamento se tal caso de teste será executado ou não. Conforme é realizado o planejamento dos testes a fórmula do ROI é automaticamente calculada e mostra o resultado, será possível planejar até que o retorno de investimento seja positivo. A partir do momento que o planejamento chegue a um resultado negativo, é necessário refazer o planejamento dos testes, assim eliminando alguns testes até que o resultado se torne positivo novamente. E assim, um dos princípios desse trabalho é a eliminação dos custos com os testes de software, a fim de criar um planejamento interno para a seleção de quais testes será executado, porém sempre pensando na qualidade do produto final.*

Palavras-chave: *Teste de software. ROI. Qualidade de software. Gerenciamento de testes.*

1. Introdução

Para garantir a qualidade de um produto de software, é preciso definir uma estrutura visando identificar o custo-benefício de sua qualificação, uma vez que, a qualidade do sistema esta diretamente relacionada ao seu custo de desenvolvimento. O processo tradicional propõe a criação de rotinas planejadas de teste, conforme conceituado por Pezzè e Young (2008). A execução destas rotinas é aplicada no esforço para encontrar e corrigir os defeitos que passaram despercebidos desde o levantamento de requisitos até a etapa de teste.

As organizações necessitam que sejam reduzidos os erros de software, para que também diminua os custos em relação à correção de futuros erros. Diante deste quadro, Rios e Moreira (2006) destacam que, as organizações estão em busca de redução de custos no desenvolvimento de software, porém também querem que o produto final esteja com qualidade, sendo que hoje em dia isso se tornou um desafio devido à complexidade dos negócios, ainda mais que os softwares estão crescendo de forma surpreendente.

De acordo com Pezzè e Young (2008, p. 26), "O custo da verificação de software frequentemente corresponde a mais da metade do custo total do desenvolvimento". Com base nesta afirmação, é visível que de alguma forma deve-se tomar alguma providência pelas empresas de software para tornar seus softwares confiáveis, e também ter um maior retorno do investimento, evitando futuros gastos com correções.

O presente trabalho irá apresentar uma proposta de gerenciamento de testes baseada em ROI, fornecendo uma base sólida para a tomada de decisão sobre qual massa de testes deve ser executada, proporcionando ao engenheiro de teste responsável, a segurança necessária para assumir o risco referente à redução da cobertura funcional mantendo a qualidade do produto de software testado. O objetivo principal é encontrar através de uma fórmula em planilha eletrônica o ponto ideal de testes dos requisitos funcionais, assim diminuindo a quantidade de defeitos e aumentando o retorno do investimento.

2. Teste de Software

O teste é aplicado em um software com um único objetivo, encontrar defeitos. Muitas das vezes o software é liberado ao usuário final, sem a devida revisão para verificar se esse realmente realiza as atividades conforme planejado, para as empresas que desenvolvem software existe um custo muito alto nessa situação, isso pela questão de ter que corrigir algo que poderia ter sido encontrado enquanto o software ainda estive-se na empresa. É exatamente para que não ocorra esse tipo de situação que hoje em dias as empresas têm adotado o teste de software, é a garantia de qualidade para o produto e redução de custos com correção. Está cada vez mais visível para as empresas a necessidade de formalização de uma equipe de testes (SEVERO, 2007).

O principal objetivo do processo de teste é encontrar defeitos, sendo assim, somente pode-se considerar um teste com sucesso aquele que encontra defeitos durante o processo, e quanto maior o número de defeitos encontrado, maior a qualidade dos testes, assim são minimizados os defeitos e praticamente eliminadas as chances que o produto final tenha algum defeito, garantindo a satisfação do cliente com o produto (SEVERO, 2007).

É aplicado o teste de software com o propósito de fazer uma verificação e validação, onde a verificação nada mais é que a garantia que o produto realiza realmente uma determinada funcionalidade, e na validação é analisar se a o retorno do software está dentro das solicitações do cliente (CARTAXO, 2006).

Existe diversos tipos de testes, todos de muita importância, entre esses um dos mais utilizados é o Teste de caixa branca é também conhecida por teste estrutural, é dada a caracterização de

caixa branca, pelo propósito de que se pode ver o que tem no interior dela, se aplicado isso em um software, o conteúdo da caixa seria o código fonte (PRESSMAN, 2003). O teste se inicia desde o momento que está sendo desenvolvido o software, aonde após concluir um trecho de implementação o programador testa no próprio ambiente de desenvolvimento. Assim não é testado o software por inteiro, e sim apenas por partes do código fonte.

O teste de caixa preta, é outro tipo de teste bastante utilizada, onde Pressaman (1995) destaca que o teste de caixa preta, ou teste funcional, é aplicado em cima dos requisitos funcionais do software. Esse método é utilizado para encontrar defeitos diferentes dos encontrados quando usado o processo de teste de caixa branca. O teste de caixa preta tem como objetivo ter várias condições de entrada em um software e a partir disso testar todos os requisitos funcionais de um programa, observado o processo de entrada até sua saída.

3. Retorno do Investimento (ROI)

O investimento é fundamental dentro de uma empresa, eles estão ligados a expansão, manutenção e a modernização dos negócios. Mas nem sempre investir em algum projeto significa ter retorno. É preciso estar bem ciente dos riscos e analisar todas as variáveis possíveis. Só assim é possível escolher o melhor investimento (PEREIRA; REIMER; ROCHA, 2006).

Quando se comenta sobre retorno do investimento logo se vem à questão do risco, e de acordo com Gitman (2001), risco é a chance de perda financeira, ou seja, utiliza-se o termo risco quando se tem a incerteza ao se referir à variabilidade de retornos associados a um determinado ativo.

No teste de software a definição da fórmula do ROI é definida de acordo com a figura 1.

$$ROI = ((CDst - CDct) - CT) / CT$$

Figura 1. Fórmula do ROI para aplicar em testes de software.

As definições das variáveis são definidas assim como, o CDst é o total em reais gasto para corrigir um defeito localizado em ambiente de produção. A variável CDct é o custo em reais para a correção de um defeito encontrado durante a rotina de testes, e o CT é custo total para realizar a rotina de um teste no software.

Após a aplicação dos valores nessa fórmula, pode-se dizer que o ROI será positivo de acordo com a figura 2.

$$CDst > CT + CDct$$

Figura 2. Resultado do ROI, valor em reais do retorno do investimento.

Segundo Myers (2004), quanto mais cedo se inicia os testes menores se tornam os custos com correções que terá no decorrer do projeto. Enquanto o software está em sua fase inicial e é encontrado defeito, o custo total para correção desses pode se tornar 1.000 ou mais vezes mais barato do que se o mesmo fosse encontrado apenas quando o sistema já estive-se em ambiente de produção.

De forma simples, pode-se dizer que o ROI é apenas um cálculo financeiro, Porém desenvolver o ROI é um processo muito mais detalhado e abrangente do que se possa imaginar.

4. Metodologia

Foi necessário criar um modelo de gerenciamento dos casos de teste, assim podendo aplicar a fórmula do ROI a fim de identificar a quantidade de testes necessária para garantir a qualidade do software, com total foco nos custos dos testes, mas não menosprezando a qualidade do produto.

Para realizar o processo de forma correta, deve ser seguido o fluxo da figura 3.

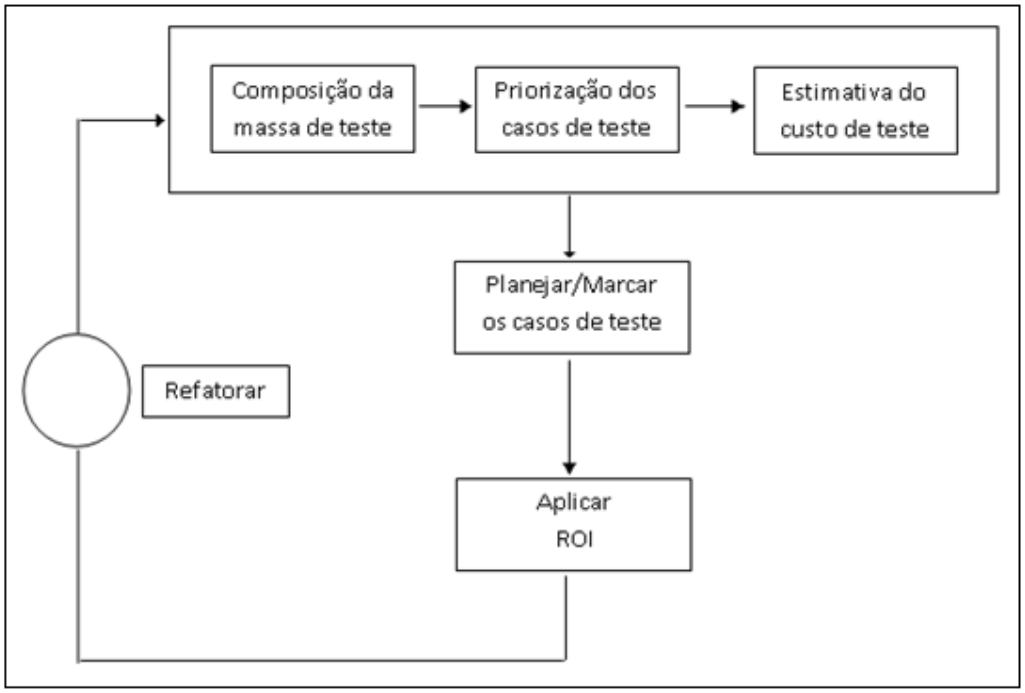


Figura 3. Fluxo do processo de teste de software.

4.1. Composição da Massa de Teste

O primeiro ponto necessário para iniciar o trabalho proposto, foi definir um software que seria utilizado para realizar uma bateria de testes, o software escolhido foi o Ieducar, é um sistema para gestão escolar, utilizado pelas escolas municipais, assim reduzindo a necessidade de uso de papel, ele possui cadastro de escolas, alunos, professores, e também existe um módulo para gerenciamento de biblioteca.

Assim iniciou-se a composição da massa de teste, a geração dos casos de testes foi feita de forma manual, a partir do conhecimento de testes e do software utilizado. Assim para cada caso de uso foi gerados desde os testes menos críticos, até os mais críticos. Sendo assim, definimos como massa de teste, todos os casos de testes gerados para os requisitos funcionais levantados.

4.2. Priorização dos casos de Teste

A priorização se faz necessário para saber por onde iniciar o teste, ele serve como uma ordenação, onde o teste com maior prioridade é o primeiro a ser executado até chegar ao menos prioritário.

Existem diversas técnicas de priorização dos casos de teste, nesse trabalho foi aplicada a técnica GUT, onde são atribuídas três notas para cada caso de teste, as notas vão de 1 a 5, sendo que quanto maior o valor atribuído maior será a prioridade no final. Primeiro se faz a

avaliação da Gravidade, depois da urgência e por fim a tendência, para esses três pontos é atribuído um valor.

A gravidade é medida pelo impacto do problema na empresa, efeitos que vão surgir com o tempo se não for solucionado, a nota para urgência é o tempo que se tem para resolver o problema e a tendência é o crescimento do problema. Tendo as notas, essas são somadas, e divido por três, no final cada teste terá apenas uma nota, então se ordena pela nota final, sendo essa a sua priorização e ordenação de execução dos testes.

4.3. Estimativa do custo de teste

Para a execução da fórmula do ROI, é necessário fazer uma estimativa de custo do teste, esse é um elemento que será utilizado na fórmula do ROI, é fundamental para o resultado final.

Caso não se tenha um histórico de testes executados, para a primeira vez pode ser estimado um valor. Para quem já possui uma massa de teste já executada e com os tempos de teste, a estimativa pode ser feita sobre esses valores, assim tornando o resultado final mais próximo da realidade. O ideal é que a massa de teste já tenha sido executada pelo menos uma vez, para ter dados consistentes, caso a massa de teste tenha sido executado mais de uma vez, a recomendação é que seja realizada a média dos tempos de execução de cada caso de teste, assim as probabilidades do valor do ROI ser mais próximo da realidade.

Para se estimar o custo do teste, é necessário ter em reais o valor da hora trabalhada pelo testador, pelo analista e pelo desenvolvimento. O tempo de teste deve ser transformado totalmente em horas.

4.4. Planejar/Marcar os Casos de Teste

No planejamento dos casos de teste são definidos quais deles serão executados. O planejamento inicia com a definição da quantidade de bugs que é a quantidade de defeitos encontrada durante a execução dos testes. Esses defeitos serão considerados a média de defeitos que o teste encontrou nos testes anteriores e a quantidade de bugs externos que é a quantidade de defeitos que o teste não encontrou, e esses foram encontrados no ambiente de produção. Deve ser realizada a média dos testes anteriores.

Na planilha que possui todos os casos de teste, foi elaborada uma coluna chamada de planejamento previsto, essa serve para definir quais testes serão executados, a marcação é com um ‘X’, conforme vai marcando os testes, automaticamente o ROI é calculado, então o responsável pela definição dos testes deve ficar acompanhando os resultados, pois o valor final do ROI deve ser positivo, a partir do momento que esse se tornar negativo os custos serão maiores que o retorno.

4.5. Aplicar ROI

Para o cálculo do ROI serão consideradas várias variáveis, o cálculo inicia-se no momento em que são planejados os casos de teste. Assim conforme são feitas as marcações a fórmula é executada e mostra o valor na planilha. O ROI será calculado através da fórmula definida na figura 1. Porém para chegar aos valores de cada variável foi necessário um estudo para encontrar um valor próximo da realidade. Abaixo está à definição de como chegar aos valores das variáveis acima:

CDct = CT + CR + CCORDEV + CA;
CDst = CA + CCORDEV;

CT = CT.
Onde, as variáveis são definidas assim:
CT: Média das horas de teste * Custo da hora do teste em reais;
CR: Média das horas de reteste * Custo da hora do teste em reais;
CCORDEV: Média das horas de correção do desenvolvimento * Custo da hora do desenvolvimento em reais;
CA: Média das horas de analise * Custo da hora do analista em reais.

O cálculo do ROI foi elaborado de duas formas, uma sobre os custos reais, e outra sobre a regra 10 de Myers, onde Myers define que os custos aumentam exponencialmente de acordo com o ciclo do software, nesse caso foi definido que, o custo de correção para os bugs encontrados pela equipe de teste foi multiplicado por 10, já os bugs externos serão multiplicados por 100.

5. Resultados

Conforme figura 4, foi calculado o ROI sobre a realidade, foram executados 55 casos de teste, onde desses nenhum defeito foi reportado pela equipe de testes nem pelo usuário final. O resultado do retorno do investimento foi de -100% sobre os gastos reais e também sobre a regra 10 de Myers, pois todo o custo com o teste foi considerado gasto, e do total desse montante nada será recuperado.

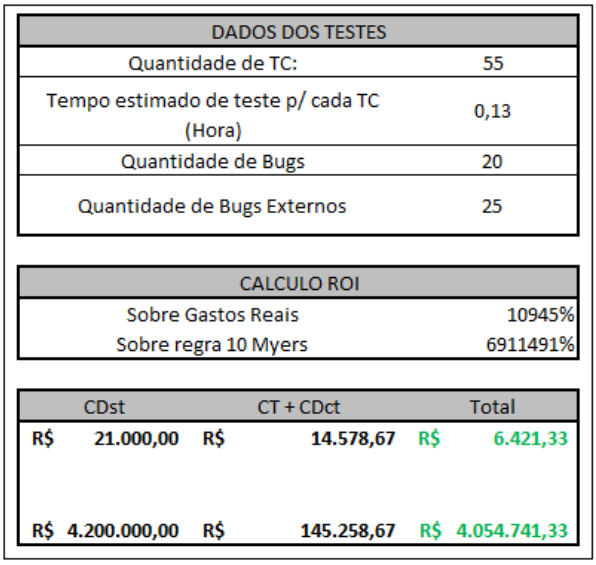
DADOS DOS TESTES	
Quantidade de TC:	55
Tempo estimado de teste p/ cada TC (Hora)	0,13
Quantidade de Bugs	0
Quantidade de Bugs Externos	0

CALCULO ROI	
Sobre Gastos Reais	-100%
Sobre regra 10 Myers	-100%

CDst		CT + CDct		Total
R$	-	R$ 58,67	-R$	58,67
R$	-	R$ 58,67	-R$	58,67

Figura 4. ROI aplicado em teste de software sem defeito.

Na terceira situação simulada, foram estimados 20 defeitos corrigidos no teste, e 25 encontrados em ambiente de produção. Nesse caso apenas 5 defeitos não foram reportados pela equipe de teste, e mesmo assim os lucros são maiores que as despesas, conforme detalha a figura 11. Os lucros foram de R$ 6.421,33, sobre os gastos reais, e de R$ 4.054.741,33 considerando a regra 10 de Myers. Sendo assim, poderia ter sido marcado mais casos de teste para executar.



DADOS DOS TESTES	
Quantidade de TC:	55
Tempo estimado de teste p/ cada TC (Hora)	0,13
Quantidade de Bugs	20
Quantidade de Bugs Externos	25

CALCULO ROI	
Sobre Gastos Reais	10945%
Sobre regra 10 Myers	6911491%

CDst	CT + CDct	Total
R$ 21.000,00	R$ 14.578,67	R$ 6.421,33
R$ 4.200.000,00	R$ 145.258,67	R$ 4.054.741,33

Figura 5. ROI aplicado em teste de software com defeito.

Com os valores mencionados acima, podemos perceber que a correção de um defeito mesmo sem medir o tamanho do impacto desse, quase sempre será vantajoso executar testes no software. Agora isso fica mais claro ainda quando é calculado o ROI sobre a regra 10 de Myers, onde ele mede não só o custo para a correção, mas também o impacto ocasionado por esse defeito, desde o cliente que pode ter ficado parado até a mobilização dos funcionários da empresa para encontrar o defeito e resolver a situação o quanto antes.

6. Conclusão

Nos dias de hoje as empresas buscam o melhor para seus clientes, no caso de uma empresa de software, o melhor é um sistema pronto par atender a necessidade do cliente e com qualidade. E um dos pontos que definem a qualidade do produto, é o teste, pois é a partir dele que será exercitado o sistema para saber se realmente ele faz o que deve fazer. Hoje em dia as empresas vêm buscando criar o setor de testes, e as que já têm, vem buscando o aperfeiçoamento desses. Porém quando falamos em teste, estamos falando em custo, e muitas vezes é um custo muito elevado, gerando no fim prejuízo ao invés de lucro. Como resultado final, foi criado um modelo de teste, essa desenvolvida em planilha eletrônica, a fim de documentar os casos de teste nela para gerar o cálculo do ROI, assim o analista de teste vai saber informar quais testes serão executados, evitando que o investimento traga custos, assim podendo até gerar lucro.

Referencias

CARTAXO, Emanuela Gadelha. (2006) “Geração de casos de teste funcional para aplicações de celulares”, 146 f. Dissertação (Mestrado) - Ufcg, Campina Grande. Disponível em: <http://docs.computacao.ufcg.edu.br/posgraduacao/dissertacoes/2006/Dissertacao_EmanuelaGCartaxo.pdf>. Acesso em: 18 set. 2011.

GITMAN, Lawrence J. (2001) “Princípios de administração financeira”. 2.ed. Porto Alegre: Bookman. 610 p. ISBN 857307776x

MYERS, Glenford J. (2004) “The Art of Software Testing”. New York: John Wiley & Sons, Second Edition.

PEREIRA, Leonardo Dos Santos;REIMER, Leonardo Ribeiro; ROCHA, Luciano Guimarães. (2006) “Plano de Negócio em Tecnologia”. 56 f. - Ucg, Goiania. Disponível em: <http://aldeia3.computacao.net/greenstone/collect/trabalho/index/assoc/HASHc840.dir/doc.doc>. Acesso em: 12 nov. 2011.

PEZZÈ, Mauro; YOUNG, Michal. (2008) “Teste e análise de software: processo, princípios e técnicas”. Porto Alegre: Bookman. 512 p. ISBN 9788577802623 (broch.)

PRESSMAN, R.S. (1992) “Software Engineering: Practitioner's Approach”, McGRAW-HILL.

PRESSMAN, R. S. (2002) “Engenharia de Software”. 5 ed. Rio de Janeiro: McGraw-Hill, 843 p.

RIOS, Emerson; MOREIRA FILHO, Trayahú R. . (2006) “Teste de software”. 2. ed. rev. e ampl Rio de Janeiro: Alta Books. 222p. ISBN 8576081288 (broch.)