UNIVERSIDADE FEDERAL DE SERGIPE
CAMPUS ALBERTO CARVALHO
DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO

TIAGO OLIVEIRA DE REZENDE

ANÁLISE COMPARATIVA DE TÉCNICAS DE SELEÇÃO DE CASOS DE TESTE PARA O PLANEJAMENTO DE TESTES DE REGRESSÃO.

ITABAIANA
2011

UNIVERSIDADE FEDERAL DE SERGIPE
CAMPUS ALBERTO CARVALHO
DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO

TIAGO OLIVEIRA DE REZENDE

ANÁLISE COMPARATIVA DE TÉCNICAS DE SELEÇÃO DE CASOS DE TESTE PARA O PLANEJAMENTO DE TESTES DE REGRESSÃO.

Trabalho de Conclusão de Curso submetido ao Departamento de Sistemas de Informação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Msc. Marcos Barbosa Dósea

ITABAIANA
2011

Oliveira, Tiago.

Análise Comparativa de Técnicas de Seleção de Casos de Teste para o Planejamento de Testes de Regressão / Tiago Oliveira de Rezende – Itabaiana: UFS, 2011. 80f.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal de Sergipe, Curso de Sistemas de Informação, 2011.

1. Técnicas de Seleção de Casos de Teste. 2. Teste de Software. 3. Sistemas de Informação. I. Análise comparativa de técnicas de seleção de casos de teste para o planejamento de testes de regressão.

TIAGO OLIVEIRA DE REZENDE

ANÁLISE COMPARATIVA DE TÉCNICAS DE SELEÇÃO DE CASOS DE TESTE PARA O PLANEJAMENTO DE TESTES DE REGRESSÃO.

Trabalho de Conclusão de Curso submetido ao corpo docente do Departamento de Sistemas de Informação da Universidade Federal de Sergipe (DSIITA/UFS) como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Itabaiana, 06 de Dezembro de 2011.

BANCA EXAMINADORA:

_____

Prof(a) Marcos Barbosa Dósea, Mestre
Orientador
DSIITA/UFS

_____

Prof(a) André Luís Meneses Silva, Mestre
DSIITA/UFS

_____

Prof(a) Alcides Xavier Benicasa, Mestre
DSIITA/UFS

AGRADECIMENTOS

Ao professor e orientador Marcos Dósea, pela paciência na orientação e ensinamentos, que me levaram à conclusão deste TCC.

Aos amigos e colegas da Universidade, pelo apoio e ajuda constantes durante o desenvolvimento deste trabalho.

Aos meus pais, irmãos, sobrinho e a toda minha família que com muito carinho e apoio, não mediram esforços para que eu chegasse a essa etapa de minha vida.

A Deus, o que seria de mim sem a fé que eu tenho nele.

Enfim, a todos que de maneira direta ou indireta ajudaram a concluir esse trabalho de graduação.

“Tudo vale a pena se a alma não é pequena”
(FERNANDO PESSOA)

OLIVEIRA, Tiago. **ANÁLISE COMPARATIVA DE TÉCNICAS DE SELEÇÃO DE CASOS DE TESTE PARA O PLANEJAMENTO DE TESTES DE REGRESSÃO.** 2011. Trabalho de Conclusão de Curso – Curso de Sistemas de Informação, Departamento de Sistemas de Informação, Universidade Federal de Sergipe, Itabaiana, 2011.

RESUMO

Um dos principais mecanismos para atingir a qualidade do software é a utilização de técnicas que possam testar características de um produto. As características que podem ser exploradas são as funcionalidades desenvolvidas, segurança, desempenho e usabilidade do software. Para isso, utilizam-se técnicas de teste de software para atingir uma determinada qualidade do produto desenvolvido. Para facilitar o planejamento e determinar o procedimento e os dados de testes, são criados os casos de testes. À medida que avança o desenvolvimento do software e versões do software são implantadas, torna-se inviável o teste de regressão completo de todos os casos de teste executados manualmente. O tempo é fator determinante para seleção dos casos de teste que poderão ser executados. Existe uma variedade de técnicas para seleção de casos de teste mais prioritários, garantindo um grau de cobertura da funcionalidade. Os critérios a serem abordados na seleção de casos de teste são o histórico de execução, o tempo disponível para execução dos casos de teste e o impacto desses defeitos no negócio. Neste trabalho é realizado um comparativo de algumas dessas principais técnicas de seleção de casos de teste apresentadas na literatura em um projeto real de desenvolvimento de software. Para facilitar essa avaliação foi realizada a implementação dessas técnicas em uma ferramenta de gerenciamento de casos de testes amplamente utilizada pelo mercado

Palavras-chave: **Teste de Regressão, Seleção de Casos de Teste, Redução de Tempo e Custo.**

ABSTRACT

One of the main mechanisms to achieve the software quality is the utilization of techniques which are able to test the characteristics of a product. The characteristics that can be explored are the developed features, security, performance and software usability. For this, it is utilized software testing techniques to achieve a determined quality of the developed product. To ease the planing and determine the procedure and the test datum, the test cases are created. As the software development advances and versions of the software are implanted, the complete regression of all the manually executed test cases becomes impossible. Time is a determinant factor to the selection of test cases that will be able to be executed.There is a variety of techniques to select the more priority test cases, which guaranties a degree of functionality coverage. The criteria to be addressed in the selection of the test cases are the execution historic, the available time to the execution of the test cases, and the impact of these defects on the business. In this work, it is performed a comparative of some of the main techniques of test case selection presented in literature at a real project of software development. To facilitate this exam, it was performed the implementation of those techniques in a tool of test cases management widely used in the market.

Keywords: **regression test, selection of test cases, reduction of time and cost.**

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

RUP	Rational Unified Process
UFS	Universidade Federal de Sergipe
CIn	Centro de Informática
UFPE	Universidade Federal de Pernambuco
CPD	Centro de Processamento de Dados
FURPS	*Functionality, Usability, Reliability, Performance, Supportability*

SUMÁRIO

1. INTRODUÇÃO

A atividade de desenvolvimento de software está cada vez mais complexa, já que os sistemas estão se tornando cada vez mais importantes e mais exigidos pela sociedade. Por isso a grande preocupação em construí-los de forma que não causem falhas durante a utilização, garantindo assim uma alta qualidade do produto desenvolvido.

Teste de Software consiste na verificação dinâmica do comportamento de um programa através de um conjunto finito de casos de teste devidamente selecionados a partir de um domínio de execuções usualmente infinito. (SWEBOOK, 2004)

Durante o desenvolvimento do software muitas mudanças são realizadas, alterações de requisitos pelo usuário, alteração de funcionalidades. Depois de realizadas as alterações no software, este precisa ser testado novamente, pois novos erros podem aparecer a partir das alterações realizadas, para isso é realizado o Teste de Regressão. Os testes de Regressão voltam a testar os segmentos já testados após a implementação de uma mudança em outra parte do software. (BASTOS, RIOS, CRISTALLI, MOREIRA, 2007).

O Teste de Regressão será feito através da execução de todos os testes em uma nova versão do software, que foram executados na versão antiga e depois comparar os resultados. (SWEBOOK, 2004)

Para garantir as características de qualidade que o software desenvolvido necessita possuir, deve-se retestar o sistema assim que novas modificações forem adicionadas no produto. Os testes de regressão são caros, mas quando executados podem reduzir o custo de manutenção do software. No entanto, devido à pressão pela entrega do sistema os testadores detêm pouco tempo para executá-los.

1.1. Problema

Os projetos de desenvolvimento de software na maioria das vezes não possuem planejamento e quando possuem os cronogramas são mal planejados em relação a custo e prazo das entregas a serem feitas do software.

Durante o desenvolvimento do software, quando novas versões são disponibilizadas, deveriam ser executados todos os casos de testes que foram utilizados na

versão anterior, o chamado teste de regressão. No entanto, devido às restrições de tempo e custo do software nem sempre todos os casos de teste são reexecutados.

Para solucionar o problema da falta de tempo e recursos para execução de testes de regressão, foram criadas técnicas de seleção de casos de teste, a fim de priorizar os casos de teste que possuem uma maior probabilidade de incidência de erros e que garantam uma maior cobertura possível das funcionalidades com um menor número de casos de teste possível.

Entretanto a variedade de técnicas disponíveis na literatura dificulta a escolha de qual delas utilizar no processo de testes. Para escolher qual técnica utilizar, deve-se verificar quais critérios cada uma utiliza para realizar a seleção dos testes. Além disso, não existem ferramentas de gerenciamento de processos de testes que implementem qualquer uma dessas técnicas, tornando praticamente inviável sua utilização num ambiente real de desenvolvimento de software.

1.2. Solução Proposta

O objetivo deste trabalho é analisar diversas técnicas de seleção de casos de teste a serem aplicadas nos Testes de Regressão em um projeto de desenvolvimento real de software. Para facilitar a aplicação das técnicas foi realizada a implementação destas em uma ferramenta de gerenciamento de processos de testes amplamente utilizada pelo mercado

1.3. Contribuições

As principais contribuições desse trabalho são:

- Análise comparativa de técnicas de seleção de casos de teste para o planejamento de testes de regressão;
- Desenvolvimento das técnicas de seleção de casos de teste em uma ferramenta de gerenciamento de testes amplamente utilizada no mercado;
- Avaliação da ferramenta e das técnicas de seleção em um projeto real de desenvolvimento de software.

1.4. Organização deste Trabalho

Este trabalho encontra-se dividido da seguinte forma:

- No Capítulo 2 é feita uma abordagem dos assuntos e técnicas referentes a Testes de Software;
- O conteúdo do Capítulo 3 é composto de algumas técnicas de seleção de caso de teste encontradas na literatura;
- O Capítulo 4 contém a introdução de algoritmos para seleção de casos de teste desenvolvida em uma ferramenta de gerenciamento de testes;
- O conteúdo referente ao Capítulo 5 é mostrado uma análise comparativa entre as técnicas de seleção de casos de teste desenvolvidas na ferramenta através dos resultados obtidos;
- No Capítulo 6 é apresentada a conclusão deste trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo são abordados os principais conceitos, importância e técnicas relacionadas à área de Testes de Software. Na Seção 2.1 é introduzido o conceito de Engenharia de Software sendo a área a que pertence os Testes de Software. Na Seção 2.2 é apresentada a importância da qualidade no contexto de softwares. E finalmente, na Seção 2.3, são apresentados os conceitos essenciais para o entendimento deste trabalho relacionados aos Testes de *Software*.

2.1. Engenharia de Software

A Engenharia de Software é uma disciplina de engenharia relacionada com todos os aspectos de produção de software, desde os estágios iniciais de especificação do sistema até sua manutenção, depois que este entrar em operação (SOMMERVILE, 2007).

Tal disciplina evoluiu significativamente, procurando estabelecer técnicas, critérios, métodos e ferramentas para a produção de software em consequência da crescente utilização de sistemas computacionais em praticamente todas as áreas da atividade humana, o que provoca uma crescente demanda por qualidade e produtividade (DELAMARO, MALDONADO, JINO; 2007).

Os engenheiros de software adotam uma abordagem sistemática e organizada em seu trabalho, a fim de produzir software com alta qualidade (SOMMERVILE, 2007). Para garantir a qualidade do software desejada, algumas características, apresentadas na Seção 2.2 devem ser analisadas.

2.2. Qualidade de Software

Qualidade é uma característica ou atributo de alguma coisa mensurável. Porém um software que é uma entidade intelectual é mais difícil de caracterizar do que objetos físicos. No desenvolvimento de software, a qualidade do projeto abrange os requisitos, as especificações e o projeto de sistema (PRESSMAN, 2002).

Os softwares devem atender a certos atributos de qualidade e, para isso, precisamos identificar seus requisitos de qualidade que complementam os requisitos

funcionais, de desempenho, de custo e de cronograma, normalmente especificados para o desenvolvimento do software (DELAMARO, MALDONADO, JINO; 2007).

As atividades de Controle de Qualidade ajudam a identificar defeitos no software e começam desde o início do desenvolvimento até que o teste do sistema esteja completamente terminado e o sistema seja implantado. Já Garantia da Qualidade tem como função garantir que os processos e produtos de software estejam de acordo com os requisitos e os planos estabelecidos.

Uma metodologia de qualidade que faz parte do *Rational Unified Process* (RUP) é a FURPS (*Functionality, Usability, Reliability, Performance, Supportability*) (SOMMERVILLE, 2007), que engloba algumas características que devem estar presentes durante o desenvolvimento do software. Para atender à categoria Funcionalidade o software deve estar de acordo com o que foi especificado pelo usuário. A Usabilidade representa a facilidade de uso do sistema pelos usuários. Um teste bastante utilizado para verificar esse tipo de característica no software é o teste de usabilidade que foca na facilidade de uso da aplicação pelos usuários.

A característica de Confiabilidade garante a integridade dos dados, confiabilidade da estrutura de dados e da aplicação. O Desempenho garante o quanto é rápido o processamento das funcionalidades do sistema. Testes de Carga e de Desempenho são realizados nessa categoria.

A Suportabilidade representa a capacidade de um software funcionar em diversas configurações de hardware e software. Os tipos de testes realizados para garantir a suportabilidade são os testes de configuração e instalação do software.

Na seção seguinte serão abordadas algumas atividades que podem ser realizadas para que o software desenvolvido atinja certo nível de qualidade aceitável pelo usuário.

2.2.1. Verificação e Validação

Durante e depois do processo de desenvolvimento, o software precisa ser verificado para certificar-se de que atende às necessidades do cliente. Verificação e Validação (V & V) é a denominação dada a esses processos de análise e é realizado em cada estágio do processo de software, desde as revisões dos requisitos determinados até os testes funcionais do produto.

RIOS e MOREIRA (2003) definem Verificação e Validação da seguinte forma:

- **Verificação**: realizar inspeções/revisões sobre os produtos gerados pelas diversas etapas do processo de teste.
- **Validação**: avaliar se o sistema atende aos requisitos do projeto. Os testes unitários, de integração, de sistema e de aceitação podem ser classificados como testes de validação.

O papel da verificação envolve verificar se o software está de acordo com o que foi especificado pelo usuário, ou seja, se atende aos requisitos funcionais e não-funcionais do sistema. Já validação tem como objetivo assegurar que o sistema atende às necessidades do usuário.

Várias atividades podem ser realizadas durante o desenvolvimento do software para que possam garantir uma maior qualidade ao produto desenvolvido. Podem-se citar atividades referentes à Verificação como as revisões de requisitos, revisões de modelos, inspeções de código no qual o sistema é revisto para encontrar erros ou anomalias. Atividades de validação consistem na realização de Teste de Software.

2.3. Teste de Software

O Processo de Desenvolvimento de Software envolve diversas atividades dentre elas a de Testes de Software. A atividade de Testes consiste de uma análise dinâmica do produto e é uma atividade relevante para a identificação e eliminação de erros que persistem. (DELAMARO, MALDONADO, JINO; 2007).

Myers (1979) define Teste de Software como sendo o processo de executar programas com o objetivo de encontrar erros. Já Rios (2007) relata que testar é verificar se o software está fazendo o que deveria fazer, de acordo com seus requisitos, e se não está fazendo o que não deveria fazer.

Um ponto crucial na atividade de teste, independentemente da fase, é o projeto e/ou a avaliação da qualidade de um determinado conjunto de casos de teste utilizado para o teste de um produto, dado que, em geral, é impraticável utilizar todo o domínio de dados de entrada para avaliar os aspectos funcionais e operacionais de um produto em teste. O glossário de termos de Teste de Software define casos de teste como sendo um conjunto de valores de entrada / *inputs*, precondições de execução, resultados esperados e pós-condições de execução desenvolvidas para um determinado objetivo ou condição de teste. Os casos de teste têm como objetivo exercitar o caminho de um determinado programa ou verificar o atendimento a

um requisito especifico. O objetivo é utilizar casos de teste que tenham alta probabilidade de encontrar a maioria dos defeitos com um mínimo de tempo e esforço, por questões de produtividade. (DELAMARO, MALDONADO, JINO; 2007).

Na próxima seção são detalhadas as etapas do processo de teste de software que devem ser realizadas durante a execução dos testes.

2.3.1. Processo de Teste de Software

O ciclo de vida do Processo de Teste de Software é composto por diversas etapas. A Figura 1 explica as etapas do ciclo de vida de um processo de teste, conhecido como Modelo 3P x 3E. Tal modelo é assim denominado, pois nele contém as seguintes etapas: Planejamento, Procedimentos Iniciais, Preparação, Especificação, Execução e Entrega.

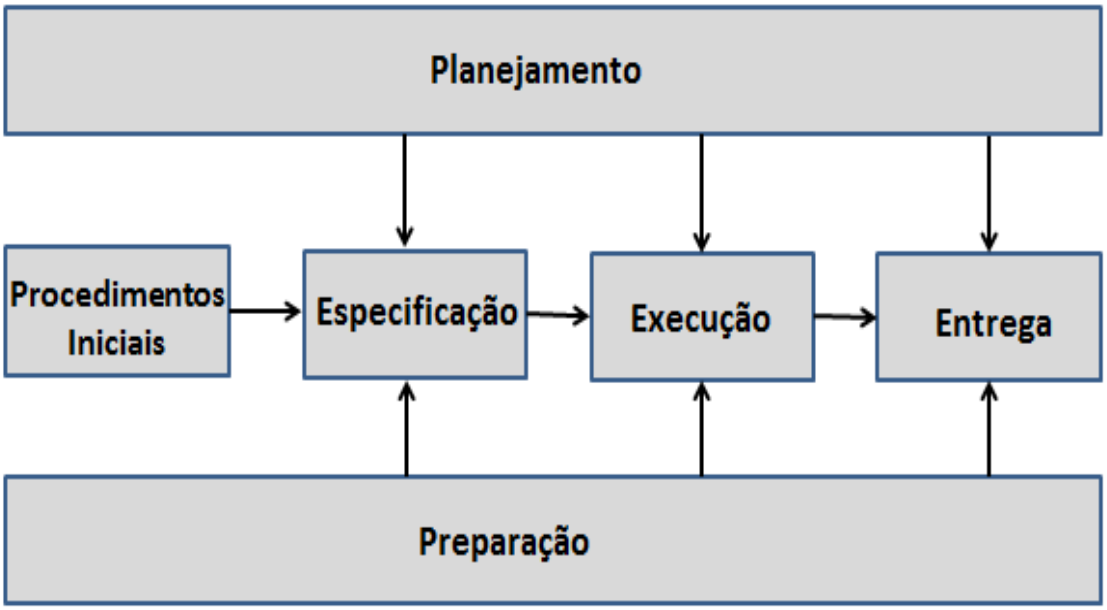


Figura 1 - Modelo 3P x 3E do ciclo de vida do processo de teste.
Fonte: Adaptado de BASTOS; RIOS; CRISTALLI; TRAYAHÚ, 2007.

Durante a etapa de **Procedimentos Iniciais** é elaborado um estudo sobre os requisitos do negócio do sistema a ser desenvolvido, a fim de garantir que o mesmo esteja completo. Um plano deve ser elaborado com todas as atividades que serão executadas durante o processo de teste.

Durante a etapa de **Planejamento** são elaborados o Plano de Teste e a Estratégia de Teste a serem utilizados, com o objetivo de diminuir os principais riscos do negócio a partir da criação da análise de riscos do projeto de testes. O objetivo da etapa de **Preparação** é preparar o ambiente de teste como, ferramentas de automação, pessoal, hardware, software para que os testes sejam executados como o planejado.

Na etapa de **Especificação** ocorre a elaboração e revisão dos casos de teste e dos roteiros de teste. Essas atividades são elaboradas durante o decorrer do projeto de teste, ou seja, elaborados a partir de quando a equipe de desenvolvimento liberar partes do sistema. A etapa de **Execução** é a qual em que os testes são realizados, obedecendo estritamente aos roteiros e os casos de teste que foram elaborados na etapa anterior.

Na última etapa denominada **Entrega**, os testes serão finalizados depois de executados todos os casos de teste e relatados todas as ocorrências relevantes para a melhoria da qualidade do software.

A próxima seção aborda a importância da execução dos testes desde as fases iniciais do processo de desenvolvimento do software.

2.3.2. Conceito “V” de Teste de Software

O ciclo de vida de testes pressupõe que os testes são realizados durante todo o processo de desenvolvimento do software. Os produtos que são desenvolvidos durante o processo devem ser revisados para que sejam criadas condições necessárias para as atividades posteriores como a implementação, identificando defeitos o mais cedo possível. A Figura 2 ilustra o conceito “V” de Teste de Software.

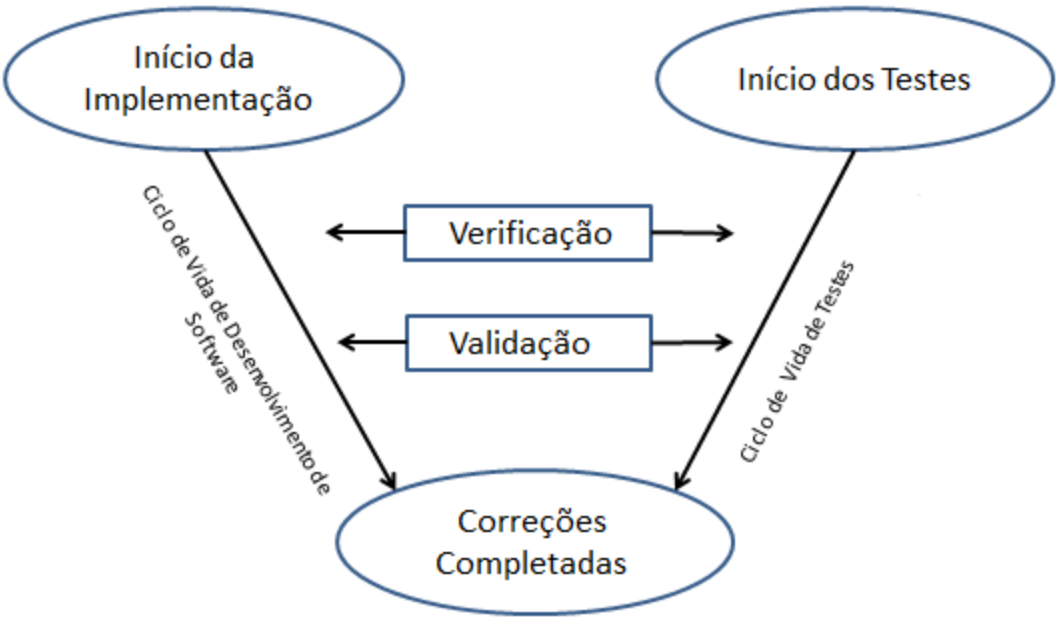


Figura 2 - Conceito "V" de Teste de Software
Fonte: Adaptado de BASTOS; RIOS; CRISTALLI; TRAYAHÚ, 2007.

Uma das vantagens devido à utilização de Testes desde as etapas iniciais do desenvolvimento de software é a detecção precoce de defeitos. A Figura 3 mostra que quanto mais tarde é a correção de defeitos, maior será o custo para corrigi-los. Defeitos encontrados durante a produção tendem a custar muito mais que defeitos encontrados em modelos de dados e em outros documentos do projeto de software (BASTOS, RIOS, CRISTALLI, MOREIRA, 2007).

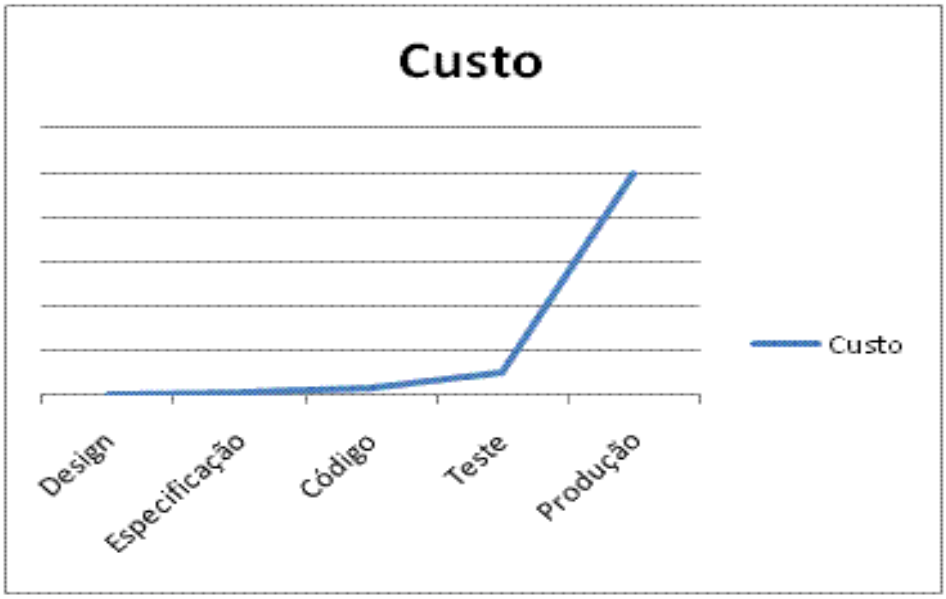


Figura 3 - Regra 10 de Myers.
Fonte: MYERS, 1979.

Existem vários exemplos de falhas que ocorreram devido a uma falha no software e que causaram impactos muito negativos para a sociedade. A seguir será relatado uma falha no foguete Ariane 5 e mostrando a importância de utilizar testes durante o desenvolvimento do software, a fim de reduzir a chance de ocorrer uma falha durante a execução do mesmo.

Em 1996, após o lançamento do foguete Ariane 5 o mesmo se autodestruiu pelo motivo de um erro no software. A anomalia interna do software ocorreu durante a execução de uma conversão de dados de um número de 64 bits em ponto flutuante para um inteiro de 16 bits com sinal. O resultado foi um operando inválido relacionado ao alinhamento inercial do foguete.

2.3.3. Níveis de Teste de Software

Os testes são executados em diferentes níveis do desenvolvimento do software. A Figura 4 ilustra os quatro níveis de teste de software. Os níveis de testes abordados são os seguintes: unitário, integração, sistema e de aceitação. Desde as fases iniciais de desenvolvimento do software já podem ser realizados testes. A seguir são apresentando os objetivos e atividades que devem ser desenvolvidas em cada nível dos testes.

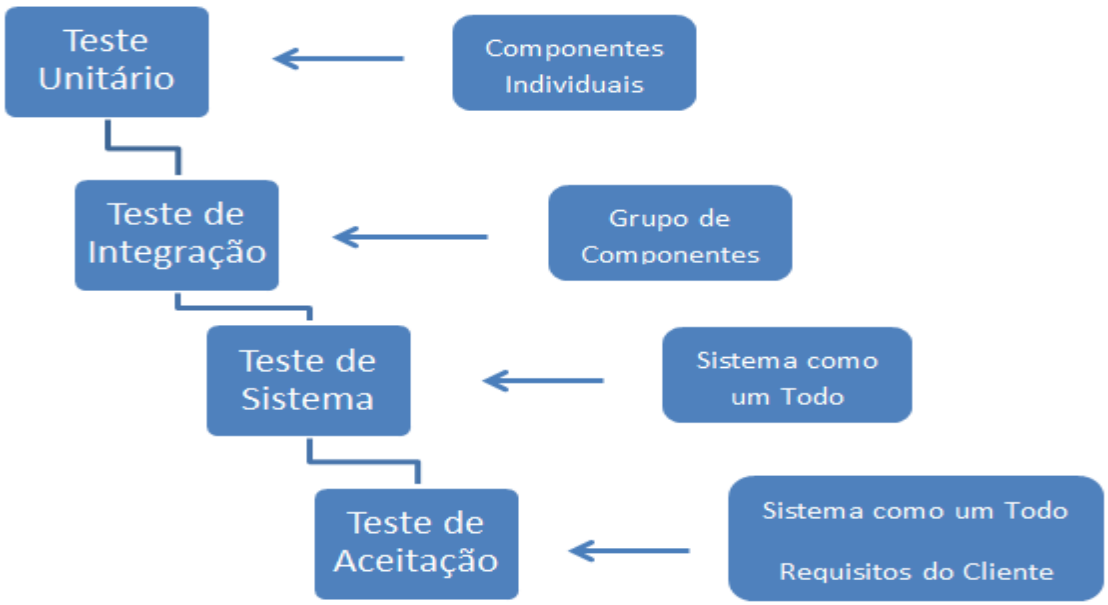


Figura 4 - Níveis de Teste de Software.
Fonte: Adaptado de MYERS, 1979

Teste Unitário

São os primeiros testes possíveis de serem realizados em um software. Cada unidade de código é testada individualmente, ou seja, para cada componente individual do sistema desenvolvido, deve ser criado um teste unitário. Entende-se por unidade de código, como sendo a um método em linguagens orientadas a objeto. Esse tipo de teste é normalmente executado pelo próprio desenvolvedor do código.

Teste de Integração

É o processo de verificação da interação entre componentes ou módulos do software (SWEBOOK, 2004). É uma atividade que tem como objetivo verificar se realmente os componentes ou módulos do sistema comunicam-se da maneira correta. Neste nível, os testes ainda são realizados normalmente pelos desenvolvedores.

Poder-se-ia imaginar que testes de integração não seriam necessários quando as unidades isoladas foram criteriosamente testadas. Entretanto, o teste isolado de uma unidade não garante que a sua combinação com outras unidades irá se comportar adequadamente.

Além disso, é importante que testes de integração sejam realizados somente após os testes de suas unidades terem passado (BURNSTEIN, 2002).

Teste de Sistema

O teste de sistema está preocupado com o comportamento do sistema como um todo (BURNSTEIN, 2002). Diferente dos testes unitários e de integração, os testes de sistema são preferencialmente executados em um ambiente separado, idêntico ao ambiente de produção. Neste sentido, são também durante este nível que são realizados os testes funcionais, dos quais são selecionados os que irão compor as suítes de teste de regressão. O glossário de termos de Teste de Software define suítes de teste como sendo o conjunto de vários casos de teste para um componente ou sistema sendo testado, no qual a pós-condição de um teste é frequentemente utilizada como precondição para o próximo. O objetivo é para garantir que o sistema funcione de acordo com suas exigências.

Neste nível, espera-se que as unidades tenham sido integradas com sucesso e o sistema (ou módulo/subsistema) esteja pronto para ser testado por uma equipe que desconheça seus aspectos técnicos (caixa-preta). (BURNSTEIN, 2002)

É durante esse nível que são realizados os testes de carga, de performance, confiabilidade entre outros, caso não haja uma equipe específica para realizar as atividades. Também são realizados os testes funcionais, dos quais irão ser selecionados para compor as suítes dos testes de regressão.

Teste de Aceitação

Teste de aceitação é o processo de comparar o programa com os requisitos iniciais e necessidades atuais do cliente. É usualmente realizado por um grupo de usuários finais e normalmente não é considerado como uma responsabilidade da equipe de desenvolvimento. O teste de aceitação é realizado para determinar se o produto satisfaz as necessidades do cliente (MYERS, 1979).

A seguir serão apresentados os principais tipos de testes que são realizados nos softwares para garantir certo nível de qualidade do produto desenvolvido.

2.3.4. Tipos de Testes

Existem na literatura vários tipos de testes que são aplicados em condições idênticas ou próximas das do ambiente de produção. A seguir serão apresentados alguns tipos de testes que podem ser utilizados nessa fase de testes, inclusive o teste de regressão que é a base para o desenvolvimento deste trabalho.

Teste Funcional

Myers (1979) define teste funcional como um processo de tentar encontrar discrepâncias entre o programa e sua especificação externa. Uma especificação "externa" é uma descrição precisa do comportamento do programa do ponto de vista do usuário. Testes funcionais são testes caixa-preta por natureza, uma vez que o foco está nas entradas e saídas apropriadas para cada função (BURNSTEIN, 2002).

O objetivo dos testes funcionais é a partir das entradas em cada funcionalidade, verificar suas saídas respectivas se realizam de maneira correta o que deve ser feito.

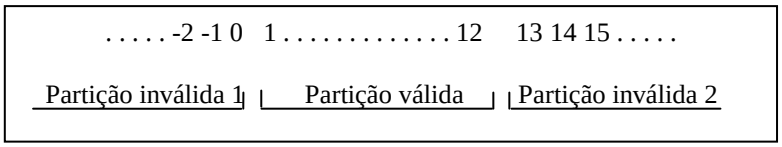
Durante esses tipos de testes alguns métodos são utilizados para geração de casos de teste. São eles: Particionamento em Classes de Equivalência; Análise de Valor-Limite; Grafo de Causa-Efeito (NASCIMENTO, 2008).

A técnica Particionamento em Classes de Equivalência divide o domínio das entradas em classes de dados. Cada classe válida e inválida representa um conjunto de elementos que repercutem num mesmo comportamento do sistema.

Após definir as classes de equivalência para cada condição de entrada, casos de teste devem ser criados primeiramente para as classes de equivalência válidas, até que todas sejam cobertas. Em seguida, deve ser criado um caso de teste para cada classe de equivalência inválida, pois o comportamento do sistema para cada uma dessas classes deve ser verificado individualmente (MYERS, 1979).

Muitos erros ocorrem quando se utiliza valores de fronteira como entrada, então a técnica de Análise de Valor-Limite realiza a escolha de valores limítrofes na definição de casos de teste. Inicialmente, deve-se saber quais os conjuntos de classificações compreendidos pelo sistema e depois levantar valores de fronteira para esses conjuntos de dados.

Para um exemplo onde os valores de entrada são meses do ano expressos em valores inteiros, o parâmetro de entrada "mês" deve ter as seguintes partições:



Quadro 1 : Exemplo de Classes de Equivalência e Valor Limite

No exemplo do Quadro 1, existem valores de limite em 0, 1, 2 e 11, 12 e 13 e ainda no mesmo exemplo, os valores são separados em duas partições inválidas e uma partição válida.

A técnica Grafo de Causa-Efeito é utilizada para complementar as técnicas Particionamento em Classes de Equivalência e Análise de Valor-Limite. Ela propõe a combinação de condições de entrada para a geração de casos de teste a partir da utilização de grafos. Esta técnica apoia na seleção sistemática de um conjunto de casos de teste efetivos (MYERS, 1979). Uma causa é uma condição de entrada e um efeito é uma condição de saída. Utilizam-se os seguintes operadores lógicos: AND, OR, NOT.

Testes Exploratórios

Bach (2009) define testes exploratórios como sendo um processo em que o testador elabora o design do teste, aprende e executa ao mesmo tempo em que explora o produto. É uma abordagem muito utilizada quando existe pouca documentação do sistema a ser testado e quando existe pressão por conta de prazo.

Bach (2009) determina que o teste exploratório deve ser utilizado nas seguintes situações:

- Necessidade de aprender a utilizar o produto rapidamente;
- Necessidade de *feedback* rápido sobre um novo produto;
- Encontrar o erro mais importante no menor espaço de tempo;
- Investigar e isolar um defeito específico.

Ao finalizar um processo de execução de teste exploratório algumas informações devem ser fornecidas, tais como: as falhas encontradas, o registro de como o produto foi testado e algumas informações do produto em si.

Para a execução desse tipo de teste, é criado um *Charter*, no qual é definido o escopo, objetivo do teste e outras informações essenciais para o produto ser testado adequadamente.

Para a correta criação do *charter* ele ainda deve conter tais informações:

- Áreas de Concentração: Áreas que deverão ser exploradas durante a execução;
- Tempo: Deve ser informado o tempo máximo e mínimo da duração do teste;
- Requisitos: Documentos que auxiliam na execução do charter;
- Configurações Gerais: Alguma configuração específica para executar os testes;
- Observações: Campo livre para o testador anotar algum imprevisto durante a execução dos testes.

A partir da criação do charter, o testador utiliza-o para executar o que está presente no charter, abordando suas áreas de concentração e as demais informações presentes nesse documento.

Teste de Segurança

Testes destinados para que os dados do sistema possam ser acessados apenas por pessoas autorizadas. Testes de segurança avaliam as características do sistema que se relacionam com a disponibilidade, integridade e confidencialidade dos dados do sistema e dos serviços. Usuários/clientes devem ser incentivados a certificar-se de que suas necessidades de segurança são claramente conhecidas, de modo que as questões de segurança podem ser abordadas pelos testadores (BURNSTEIN, 2002).

Os efeitos de falhas de segurança podem causar:

(i) perda de informações;

(ii) a corrupção de informações;

(iii) a desinformação;

(iv) violação da privacidade;

(v) negação de serviço.

Teste de Desempenho

Os testes de desempenho têm como objetivo verificar se o mesmo atende aos requisitos de desempenho determinados pelo usuário. Durante os testes de desempenho, os testadores verificam fatores de hardware e software que possam ter algum impacto durante a execução do sistema. Os testadores que realizam teste de desempenho no sistema deve possuir uma grande afinidade com o mesmo, para aperfeiçoar a alocação de recursos necessários para a sua execução. Por exemplo, testadores podem achar que eles precisam realocar memória, ou modificar o nível de prioridade das operações de determinado sistema (BURNSTEIN, 2002).

Para ajudar a alcançar níveis de desempenho adequados, você precisa garantir que as especificações operacionais são descritas durante a fase de levantamento de requisitos. Sem especificações escritas ou metas, você não sabe se seu aplicativo está de acordo com o que o usuário determina (MYERS, 1979).

Teste de Stress

Teste de stress são testes que são realizados com o sistema em condições extremas, para verificar seu comportamento quando seus recursos são alocados a partir de seus valores máximos.

Isto é particularmente importante para sistemas de tempo real na qual os eventos imprevisíveis podem ocorrer, resultando em cargas de entrada que exceda aqueles descritos nos documentos de requisitos. Os testes de estresse muitas vezes descobrem condições de corrida, *deadlocks*, esgotamento de recursos ou padrões não planejados, e perturbações no funcionamento normal do software.

Teste de Configuração

O teste de configuração permite que os desenvolvedores/testadores avaliem o desempenho do sistema e disponibilidade quando as trocas de hardware e reconfigurações ocorrem. O teste de configuração também requer muitos recursos, incluindo vários dispositivos de hardware utilizados para os testes. Se um sistema não tem requisitos

específicos para as mudanças de configuração do dispositivo, o teste de configuração de grande escala não é essencial (BURNSTEIN, 2002).

O teste de configuração tem como objetivo garantir que o software funcione de maneira aceitável quando submetido a diferentes tipos de configurações de software e hardware.

Testes de Regressão

O teste de regressão não é um nível de teste, mas são novos testes que ocorrem quando são feitas alterações para garantir que a nova versão do software esteja de acordo com o especificado. Segundo PRESSMAN (2002), testes de regressão são realizados para verificar se manutenções realizadas no sistema introduziram efeitos colaterais indesejáveis.

O teste de regressão pode ocorrer em qualquer nível de teste, por exemplo, quando os testes de unidade são executados na unidade pode passar um número de testes até que um dos testes não revele um defeito (BURNSTEIN, 2002).

Entretanto em ambiente reais é muito difícil executar o teste de regressão completo por restrições de tempo e custo. No próximo capítulo são abordadas técnicas que selecionam e priorizam os casos de testes para facilitar o planejamento de um ciclo de testes de regressão.

3. TÉCNICAS DE SELEÇÃO DE CASOS DE TESTE

Este Capítulo tem como objetivo principal fornecer o embasamento teórico com relação às principais técnicas de seleção de casos de teste encontradas na literatura, para planejamento de testes de regressão. São abordadas diferentes técnicas de seleção de testes com o intuito de realizar uma análise comparativa entre algumas delas.

3.1. Técnicas de Seleção de Casos de Teste

A medida que um software evolui em suas funcionalidades, as novas versões que são desenvolvidas precisam ser testadas novamente. Essa nova bateria de testes é denominada de Testes de Regressão. É um processo caro, pois todo o software é testado para detectar se novas falhas foram adicionadas a partir das versões que vão sendo desenvolvidas.

Em grande parte dos projetos de desenvolvimento de software, um grave risco que é observado, são os prazos de entrega de versões do software para o cliente, devido ao mau planejamento e gerenciamento das atividades que compõem o projeto. Como consequência desse grave erro que é encontrado, não há tempo suficiente para testar novamente todas as funcionalidades que foram adicionadas e alteradas nas novas versões que são desenvolvidas.

Para solucionar tal problema, existem na literatura técnicas para seleção de casos de teste baseados no histórico de execução dos ciclos de testes anteriores. O objetivo é selecionar casos de teste com maior probabilidade de encontrar erros e que tenham maior importância para o negócio. Algumas dessas técnicas são abordadas nas próximas seções.

3.1.1. Métricas e Teste baseado em Risco (*Risk Based Testing and Metrics)*

Essa técnica (AMLAND, 1999) faz uma análise dos principais critérios que revelam um maior risco de ocorrência de erros em determinados testes. Os critérios que são utilizados nessa técnica são o custo da falha por parte do vendedor e do cliente e a probabilidade de uma falha ocorrer.

Os custos da falha pelo cliente são determinados por perda de espaço no mercado e não cumprimento de regulamentos governamentais. Já por parte do vendedor pode ser o custo de manutenção do software por causa de uma função com falhas.

O cálculo do fator de exposição ao risco é dado pela seguinte fórmula apresentada no Quadro 2:

FER = P(*f*) * C(*f*)

Quadro 2 - Fórmula do cálculo do fator de exposição ao risco

Onde: FER = Fator de exposição de risco de uma função

P(*f*) = Probabilidade de ocorrência de falha em uma função

C(*f*) = Impacto de uma falha

O impacto de uma falha é calculado pela média aritmética entre o impacto de uma falha do ponto de vista do vendedor e do cliente, como mostra no Quadro 3.

C(*f*) = C(*c*) / C(*v*)

Quadro 3 : Cálculo do impacto de uma falha

Onde: C(c) = Impacto decorrente da falha na visão do cliente

C(v) = Impacto decorrente da falha na visão do vendedor

A seguir, cada um desses elementos será detalhado:

- **Impacto**

O impacto é a consequência que uma falha pode trazer para o sistema caso venha a ocorrer com o sistema em produção. O impacto decorrente de uma falha em uma funcionalidade crítica de um sistema é muito maior do que uma área que não afeta o funcionamento do sistema. O impacto é avaliado por tal técnica a partir do ponto de vista do cliente (por exemplo, perda de espaço no mercado, não cumprimento de regulamentos governamentais) e do ponto de vista do vendedor do serviço (custo de manutenção elevado do software por causa de uma função com falhas). Então, o valor do impacto de tal funcionalidade para o sistema é calculado a partir da média aritmética entre os impactos de uma falha avaliados pelo cliente e vendedor. A Tabela 1 mostra os valores que são determinados para cada nível de impacto.

Impacto decorrente de uma falha de uma função	**C(*f*)**
Baixo	1
Médio	3
Alto	9

Tabela 1 - Valores de um Impacto de uma falha.

- **Probabilidade**

Quanto maior o número de erros encontrados em uma funcionalidade do sistema, maior a probabilidade de existir erros não detectados. Quando um defeito é encontrado em um ciclo de teste e o mesmo é corrigido, é provável que mais defeitos sejam encontrados nesse componente, já que a correção destes defeitos causa alterações no código, consequentemente gerar novos defeitos.

Então, um teste que falhou na maioria das vezes em que foi executado deve ser ter uma maior prioridade do que um teste que não houve falhas em ciclos de teste anteriores. A partir da análise do histórico de resultados do teste, o percentual de falhas identificadas pelo teste em relação ao seu total de execuções é calculado. A Tabela 2 **Tabela 2** mostra o percentual de falhas identificadas.

Percentual de falhas identificadas	**F**

Menor que 25%	0
Maior ou igual a 25% e menor que 50%	1
Maior ou igual a 50% e menor que 75%	3
Maior ou igual a 75%	9

Tabela 2 - Valores do percentual de falhas identificadas.

Chen (2002) altera o cálculo da probabilidade proposto por Amland pela probabilidade da severidade, o qual combina a probabilidade de um erro acontecer com a severidade do erro encontrado, como é mostrado no Quadro 44.

$$P(f) = F \times S$$

Quadro 4 - Probabilidade da Severidade definido por Chen

Onde:

P(*f*): Probabilidade da Severidade;

S: Valor da Severidade;

F: Percentual de falhas identificadas.

A severidade da falha pode ser classificada da seguinte maneira:

Alta: Erros que bloqueiam as atividades de quem está utilizando o sistema e que provoquem perda de dados, problemas de memória;

Média: Muitas perdas de funcionalidades;

Baixa: Pouca perda de funcionalidade e que para ser corrigido necessite de pouco trabalho. Na Tabela 3 é mostrado os valores determinados para cada nível de severidade.

Severidade da falha encontrada	**S**
Alta	9
Média	3
Baixa	1

Tabela 3 - Severidade de uma falha encontrada.

A partir da severidade e a probabilidade de uma falha identificada, pode-se calcular a probabilidade da severidade, através da multiplicação entre o percentual de falhas

identificadas (F) e a severidade da falha encontrada (S). Portanto a probabilidade da Severidade P(*f*) é determinada da seguinte forma: P(*f*) = F x S. Esta relação que classifica a probabilidade da severidade é dada da seguinte forma: P(*f*) será igual a 9 quando F x S for igual a 81 ou 27, P(*f*) será igual a 3 quando F x S for igual a 9 ou 3 e P(*f*) será igual a 1 quando F x S for igual a 1 ou 0. A Tabela 4 mostra o mapeamento entre o valor da multiplicação e o valor determinado para P(*f*).

F x S	**P(*f*)**
0	1
1	1
3	3
9	3
27	9
81	9

Tabela 4 - Valores da Probabilidade da Severidade P(*f*).

Então, para finalizar o cálculo de priorização dos casos de teste, multiplica-se o valor da probabilidade da severidade pela média aritmética entre os impactos causados pela falha. A partir do resultado do cálculo de priorização, deve-se dar prioridade aos casos de teste que tenham maior valor do fator de exposição ao risco FER. Na Tabela 5 contém exemplos de casos de testes com valores fictícios para a realização do cálculo de priorização seguindo a proposta de Chen (2002). A partir dos resultados da tabela abaixo, deve-se dar uma maior prioridade ao caso de teste “Cadastrar Produto” em comparação ao restante dos casos de teste.

Caso de Teste	**C(*f*)**	**P(*f*)**	**FER = C(*f*) x P(*f*)**
Logar no Sistema	1	1	1
Cadastrar Cliente	3	3	9
Cadastrar Produto	9	3	27
Cadastrar Venda	1	9	9

Tabela 5 - Exemplos do Cálculo da Priorização.

Viana (2006) adota em seu trabalho, além do fator de exposição ao risco, também utiliza a pontuação em relação aos testes automatizados e tipos de testes como critérios de priorização.

Os testes automatizados podem ser classificados de três maneiras: Manuais, Semi-automáticos e Automáticos. Os testes **manuais** são executados manualmente pelos testadores, seguindo o Projeto de Teste. Um Projeto de Teste é um conjunto de passos e resultados esperados, baseados na especificação do sistema. Os testes **automáticos** são os testes aos quais são gerados automaticamente dados de testes. São utilizadas ferramentas para avaliar o comportamento do sistema ao fornecer entradas e verificar suas respectivas saídas. Já os testes **semi-automáticos** são considerados uma junção entre os outros dois tipos. Esse tipo de teste automatizado precisa de intervenção humana para ser executado. (Viana, 2006)

A classificação desse tipo de teste é definida da seguinte forma que é mostrada na Tabela 6:

Testes Automatizados	Peso
Manuais	1
Semi-automáticos	3
Automáticos	9

Tabela 6 - Testes Automatizados.

Outro critério utilizado por Viana (2006) são os tipos de testes. Durante a execução de teste funcional, uma entrada é inserida no sistema para que seja derivada uma saída. Sendo assim, os testes podem ser classificados em Positivos e Negativos.

Os **testes positivos** tentam demonstrar que um determinado módulo faz realmente aquilo que deve fazer, ou seja, verifica se o sistema realmente funciona da maneira especificada pelo usuário. Já os **testes negativos** tentam demonstrar que determinada funcionalidade não faz nada que não deva fazer, ou seja, são testes que tentam “quebrar” o sistema. A Tabela 7 apresenta os valores para cada tipo de teste:

Tipo de Teste	Peso
Positivo	3
Negativo	1

Tabela 7 - Peso para Tipo de Teste.

Deve-se dar preferência aos testes positivos, pois uma falha nesse tipo de teste pode criar uma confusão maior comparado a uma falha nos testes negativos.

Os critérios definidos pelo método proposto são definidos na Tabela 8.

Critérios	
Sigla	**Nome**
FER	Fator de Exposição ao Risco
TAU	Testes Automatizados
TPN	Tipos de Testes

Tabela 8 - Critérios de Priorização de Testes

Assim, o cálculo da seleção dos casos de teste é definido a seguir pelo seguinte método proposto mostrado no Quadro 5:

PCT = FER + TAU + TPN

Quadro 5 - Fórmula do cálculo da seleção dos casos de teste

Onde:

PCT = Prioridade do Caso de Teste

FER = Fator de Exposição de Risco

TAU = Testes Automatizados

TPN = Tipo de Testes (Testes Positivos ou Negativos)

Com os resultados do cálculo da fórmula acima, será montada uma lista decrescente a partir dos valores do cálculo de cada caso de teste, sendo os primeiros os candidatos cujo teste tem maior importância.

3.1.2. Estratégias de Seleção de Testes Baseado em Riscos

Schaefer (2005) prioriza os testes a partir da avaliação das partes mais importantes e críticas do sistema. As partes mais importantes do sistema são determinadas a partir dos fatores como custo decorrente de falha, áreas de grande visibilidade e áreas que são utilizadas com grande intensidade. As partes mais críticas do sistema determinam-se utilizando critérios

como complexidade, mudanças constantes, novas tecnologias e números de pessoas envolvidas.

A Figura 5 mostra a maneira como o risco é analisado. Schaefer (2005) define risco como sendo o produto do custo da falha, caso ocorra, pela probabilidade da mesma acontecer. A análise do custo da falha verifica a frequência que tal funcionalidade é executada no sistema, ou seja, caso uma funcionalidade seja muito utilizada pelo usuário, uma falha na mesma pode causar um impacto e gerar consequências para o usuário do sistema. A probabilidade de uma falha ocorrer é determinada pela quantidade de defeitos que já foram encontrados na mesma funcionalidade.

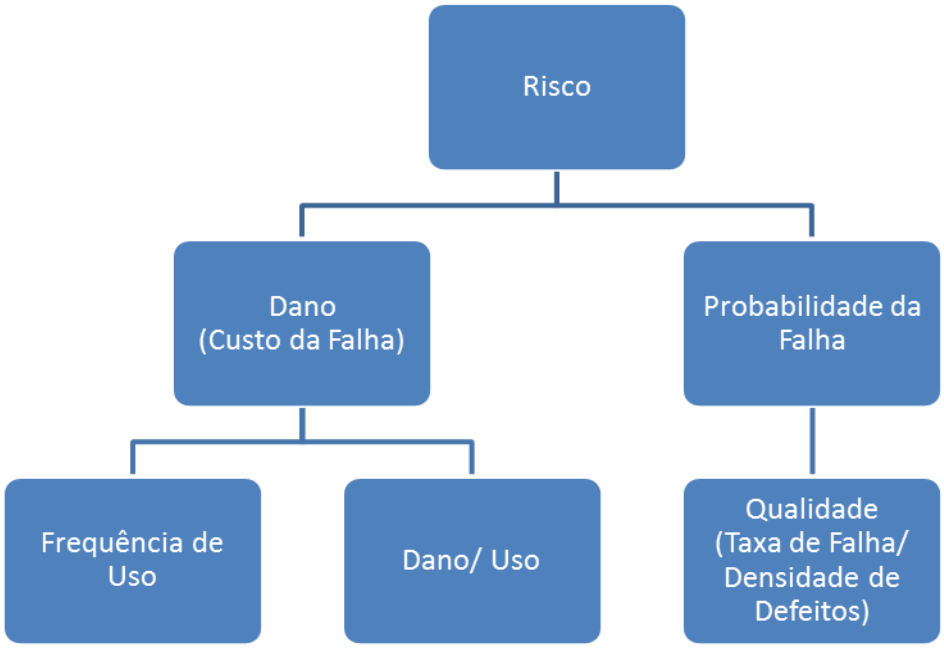


Figura 5 - Estrutura e Definição de Risco.
Fonte: Adaptado de SCHAEFER, 2005.

A partir desses fatores são atribuídos pesos em cada funcionalidade do sistema. Os valores são multiplicados pelos pesos e adicionados. Os valores mais altos desse cálculo referem-se às funcionalidades que precisam ser mais priorizadas. A Tabela 9 mostra um exemplo de cálculo de risco de algumas funcionalidades de um sistema.

Áreas	Crítica	Visibilidade	Complexa	Modificações freqüentes	RISCO
Pesos	**3**	**10**	**3**	**3**	
Ordem de registro	2	4	5	1	828
Faturamento	4	5	4	2	1116
Odem estatística	2	1	3	3	288
Gestão de relatórios	2	1	2	4	288
Desempenho de ordem registo	5	4	1	1	330
Desempenho das estatísticas	1	1	1	1	78
Desempenho do faturamento	4	1	1	1	132

Tabela 9- Cálculo do Risco
Fonte: Adaptado de Schaefer (2005).

Para realizar o cálculo do risco é atribuído um peso (1, 3 ou 10) para cada fator escolhido e uma pontuação (1 a 5) relacionada com a funcionalidade e o fator que foi escolhido.

O cálculo do risco na figura acima é calculado a partir da escolha de algumas funcionalidades (Ordem de registro, Faturamento, Gestão de relatórios, Desempenho do faturamento, etc) e dos fatores (Crítica, Visibilidade, Complexa, Modificações frequentes) que serão analisados em cada funcionalidade escolhida anteriormente além de atribuir um peso para cada fator.

Para a funcionalidade Faturamento presente na Tabela 9, o risco é calculado a partir de alguns fatores e pelos seus respectivos pesos. Os fatores da seguinte funcionalidade relacionados ao impacto de uma falha (Visibilidade e Crítica), possuem como valores 4 e 5 e pesos 3 e 10, respectivamente. Então, multiplicam seus pesos pelo valor do fator e soma seus resultados ((4 x 3) + (5 x 10)), obtendo o valor 62. Os fatores relacionados à probabilidade de uma falha ocorrer (Complexa e Modificações frequentes), possuem os valores 4 e 2 e como valores de peso 3 e 3 respectivamente para determinada funcionalidade. Multiplica-se o valor do peso pelos respectivos valores e soma seus resultados ((4 x 3) + (2 x 3)), obtendo o valor 18. Finalmente, multiplica os valores obtidos do impacto da falha da funcionalidade e da probabilidade de cada funcionalidade (62 x 18), tendo como resultado do risco da funcionalidade Faturamento, o valor 1116.

O cálculo do risco da funcionalidade Desempenho do Faturamento é realizado a partir da atribuição de valores para a funcionalidade e pesos para os fatores escolhidos correspondentes à funcionalidade. Na Tabela 9 são exibidos os fatores escolhidos com seus respectivos pesos. Para a funcionalidade Desempenho de Faturamento, os valores para os fatores "Crítica" e "Visibilidade" são 4 e 1 e seus respectivos pesos são 3 e 10. Então, o cálculo relacionado ao impacto de uma falha é determinado da seguinte forma ((4 x 3) + (1 x 10)), obtendo o valor 22. Para a mesma funcionalidade, os valores dos fatores "Complexa" e "Modificações frequentes" relacionados á probabilidade de uma falha ocorrer são 1 e 1 e seus pesos 3 e 3 respectivamente. Seu cálculo é determinado da mesma forma que o anterior, ((1 x 3) + (1 x 3)), obtendo o valor 6. O valor final do risco é determinado pela multiplicação dos resultados obtidos (22 x 9), tendo o valor 132 como valor do risco.

3.1.3. Priorização de Casos e Teste para Testes de Regressão baseado na Severidade de Falha (*Test Case Prioritization for Regression Testing based on Severity of Fault*)

Os fatores de priorização relacionadas nesta técnica são a taxa de detecção de falhas e Impacto causado pela falha (KAVITHA; SURESHKUMAR, 2010). A taxa de detecção de falhas do caso de teste é calculada a partir do número de falhas encontradas pela quantidade de execução do teste através da seguinte fórmula apresentada no Quadro 6:

RFTi = ((número de falhas) / quantidade de execuções) * 10

Quadro 6 - Fórmula da taxa de detecção de falhas

O outro critério de priorização é realizada através da classificação da severidade da falha, numa escala de 2 para a menor severidade e 10 para a severidade muito alta. Para realizar o cálculo da priorização somam-se as severidades de cada caso de teste através da seguinte fórmula exibida no Quadro 7:

$S_i = \sum SV$

Quadro 7 - Somatório dos valores das Severidades

Depois de realizado o somatório, calcula-se a proporção de severidade de cada caso de teste a partir do valor máximo da severidade. Esse cálculo é determinado a partir da fórmula exibida no Quadro 8:

$$FIi = (Si/Max(S))*10$$

Quadro 8 - Proporção de Severidade dos casos de teste

Por fim, o valor da prioridade de cada caso de teste é calculado a partir da seguinte fórmula do Quadro 9:

$$TCWi = RFTi + FIi$$

Quadro 9 - Fórmula do cálculo da Prioridade do caso de teste

Os valores de *TCWi* são ordenados em ordem decrescente e priorizados durante o processo dos testes de regressão.

Na Tabela 10 são mostrados valores referentes aos critérios utilizados por esse algoritmo para dois casos de teste. A partir dos Casos de Teste “1- Cadastrar Cliente” e “2 - Cadastrar Produto”, foram executados os testes e foram encontrados os seguintes resultados:

Caso de Teste	Nº Falhas	Nº Execuções	Severidade
1 - Cadastrar Cliente	2	9	6
2- Cadastrar Produto	3	9	10

Tabela 10 - Valores do histórico dos Casos de Testes.

$RFT_1 = (2/9) * 10 = 2,22$

$RFT_2 = (3/9) * 10 = 3,33$

$FI_1 = (6/10) * 10 = 6$

$FI_2 = (10/10) * 10 = 10$

$TCW_1 = RFT_1 + FI_1$

$TCW_1 = 2,22 + 6 = 8,22$

$TCW_2 = 3,33, + 10 = 13,33$

A partir da realização dos cálculos do algoritmo de seleção, primeiro deve-se executar o Caso de Teste “Cadastrar Produto” já que o valor de TCW_2 é maior que o de TCW_1.

3.1.4. *Test Case Selector*

A ferramenta de Seleção de Casos de Teste, *Test case Selector*, faz parte de uma cooperação entre a Motorola e o Centro de Informática da Universidade Federal de Pernambuco (CIn/UFPE) que tem como objetivo selecionar testes mais prioritários para um ciclo de regressão. A técnica utiliza 4 critérios e pesos associados a estes para obter os casos de teste mais prioritários (MAFRA; MIRANDA; IYODA; SAMPAIO). Os critérios de seleção são os seguintes:

- **Número Total de Execuções**: representa o número de vezes que determinado teste foi executado em ciclos anteriores. Se um teste é aplicável a um determinado produto e foi executado poucas vezes, maior será a sua pontuação. Este cálculo é realizado a partir da seguinte fórmula exibida no Quadro 10

$$Crit_1 = \frac{1}{execucoesCT}$$

Quadro 10 - Cálculo do Critério 1

Exemplo: A Tabela 11 mostra 3 testes fictícios, o número de execuções de cada e o resultado do critério de Número Total de Execuções.

Rank	**Teste**	**Nº de Execuções**	**Pontos**
1º	Cadastrar Cliente	50	0,02
2º	Logar no Sistema	80	0,0125
3º	Cadastrar Venda	200	0,005

Tabela 11- Resultado do Critério 1

- **Taxa de Falhas por Execução**: Este critério calcula a proporção de falhas sobre o número total de execuções de determinado caso de teste. Este critério valoriza testes que foram mais eficazes no passado. A fórmula do critério 2 é mostrada no Quadro 11:

$$Crit_2 = \frac{falhasCT}{execucoesCT}$$

Quadro 11 - Cálculo do Critério 2

Exemplo: Para os mesmos testes do exemplo anterior, assumimos valores para o número de falhas encontradas em cada caso de teste. Na Tabela 12 é exibido o resultado da aplicação do critério 2. A partir do critério acima é obtida a seguinte ordenação:

1º Cadastrar Cliente;

2º Cadastrar Venda;

3º Logar no Sistema.

Teste	Nº Execuções	Nº de Falhas	Pontos
Logar no Sistema	80	3	0,0375
Cadastrar Cliente	50	8	0,16
Cadastrar Venda	200	10	0,05

Tabela 12 - Critério 2: Falhas por execução.

- **Número de defeitos únicos encontrados:** representa o número de defeitos diferentes encontrados durante a execução do mesmo caso de teste. *Change Request* (CR) é criada para cada defeito diferente encontrado. Calcula-se o valor do critério 3 a partir da fórmula mostrada no Quadro 12:

$$Crit_3 = CRs$$

Quadro 12 - Cálculo do Critério 3

Exemplo: A Tabela 13 mostra os testes dos exemplos anteriores e a quantidade de CRs abertas por teste. A ordem por relevância para este critério é:

1º Logar no Sistema;

2º Cadastrar Venda;

3º Cadastrar Cliente

Teste	CRs	Pontos
Logar no Sistema	10	10
Cadastrar Cliente	5	5
Cadastrar Venda	3	3

Tabela 13 - Critério 3: Número de CRs abertas.

- **Nível de Regressão:** O nível de regressão é classificado de 1 a 5, quanto maior o valor mais complexo será o caso de teste e um maior número de interações com outros componentes estará presente. Um caso de teste que possui Nível de Regressão igual a 1, tem o objetivo apenas de validar os requisitos funcionais mais básicos. Já um teste com Nível de Regressão igual a 5 é complexo no qual ocorre várias iterações com vários componentes do sistema. O Quadro 13 mostra a fórmula que é determinada o valor do critério 4.

$$Crit_4 = pontosNivelRegressao_n$$

Quadro 13 - Cálculo do Critério 4

Onde, $pontosNivelRegressao_n$ são os pontos previamente atribuídos a um teste cujo Nível de Regressão é n.

Exemplo: A Tabela 14 exibe a pontuação de Nível de Regressão para cada caso de teste.

Para este cenário, dependendo da fase de desenvolvimento de software, testes com níveis 1 e 2 serão mais valorizados que testes com níveis 3, 4 e 5.

Nível Reg	Pontos
1	60
2	50
3	10
4	15
5	20

Tabela 14 - Tabela 13: Pontuações

A Tabela 15 mostra o resultado por níveis de regressão dos casos de teste.

Teste	Nível Reg.	Pontos
Logar no Sistema	4	15
Cadastrar Cliente	2	50
Cadastrar Venda	5	20

Tabela 15 - Resultado por níveis de regressão

Para o cálculo da priorização é atribuído um peso para cada critério a depender de sua importância. Após a definição dos pesos, a ferramenta *Test Case Selector*, calcula a média ponderada de todos os critérios e exibe os casos de teste em ordem decrescente de relevância. A relevância é calculada a partir da seguinte fórmula exibida no Quadro 14:

$$\text{relevância} = \frac{\sum_{i=1}^{4}(criti\ .pesoCriti)}{\sum_{j=1}^{4} pesoCritj}$$

Quadro 14 - Cálculo da relevância do caso de teste

Exemplo: A Tabela 16 mostra a atribuição fictícia de pesos para cada critério.

Critérios	Pontos
1- Número de execuções	5
2- Taxa de Falhas por execução	2
3- Número de defeitos únicos	1
4- Nível de Regressão	8

Tabela 16 - Pesos por critério

Teste	Média Ponderada
Logar no Sistema	8,13
Cadastrar Cliente	25,33
Cadastrar Venda	10,19

Tabela 17 - Resultado Final

Na Tabela 17 é mostrado o resultado da média ponderada do cálculo do algoritmo para cada caso de teste. A ordem de prioridade dos casos de teste do exemplo acima seria o seguinte: Cadastrar Cliente; Cadastrar Venda; Logar no Sistema.

3.2. Comparativo entre os critérios analisados pelas técnicas de seleção

Esta Seção verifica faz uma análise comparativa entre todas as técnicas que foram analisadas na Seção 3.1 a partir dos critérios que cada técnica verifica para selecionar os testes.

Na Tabela 18 é exibido um comparativo entre os critérios que cada técnica utiliza para selecionar os testes mais prioritários para um novo ciclo de testes.

Critérios / **Técnica**	**Severidade**	**Prioridade**	**Quantidade de Falhas**	**Quantidade de Execuções**	**Tipos de Testes**	**Nível de Regressão**
Risk Based Testing and Metrics	**X**	**X**	**X**	**X**	**X**	
Estratégias de Seleção de Testes Baseado em Riscos	**X**	**X**				
Test Case Selector			**X**	**X**		**X**
Test Case Prioritization for Regression Testing based on Severity of Fault	**X**		**X**	**X**		

Tabela 18 - Comparativo entre os critérios analisados pelas técnicas

4. FUNCIONALIDADE DESENVOLVIDA

Para auxiliar na execução das atividades de desenvolvimento de software, existem várias ferramentas que ajudam na execução das tarefas de execução dos testes e, além disso, aumentam a produtividade dos profissionais que as utilizam em seu dia-a-dia. Uma ferramenta que é amplamente utilizada por grandes empresas de desenvolvimento de software é o TestLink (TESTLINK, 2011). Este Capítulo apresenta as principais características do TestLink, e as justificativas de escolhê-la para facilitar a análise comparativa entre as técnicas de seleção de casos de teste.

4.1. Ferramenta TestLink

Na execução das atividades de Teste de Software também existem ferramentas que auxiliam na realização dos testes. Uma ferramenta de gerenciamento de testes bastante utilizada por grandes empresas no mundo é o TestLink. Dentre os objetivos e funcionalidades presentes no TestLink, pode-se destacar os seguintes:

- Especificação dos casos de teste e suítes de testes que foram mencionados no Capítulo 2 deste trabalho;
- Execução dos testes;
- Registros dos resultados da execução dos testes;
- Atribuição dos casos de testes;
- Geração de métricas e relatórios.

Na Figura 6 é exibida a tela inicial da ferramenta TestLink, com permissão de administrador do projeto de testes. Ao lado esquerdo temos quatro grandes grupos de funcionalidades. O primeiro grupo é o Projeto de Testes que permite criar projetos, criar papéis para os integrantes do projeto de testes, criar e gerenciar campos personalizados em algumas funcionalidades da ferramenta. O segundo grupo consiste na funcionalidade de Administração dos usuários, que consiste em gerenciar papéis e usuários do projeto. Esta funcionalidade é restrita ao perfil de administrador do projeto. O terceiro grupo é a parte de Requisitos, o qual poderá ser gerenciado os documentos de especificação de requisitos do sistema e dos usuários do projeto. Por último, o quarto grupo é a funcionalidade de

Especificar Testes, no qual são criados os casos de teste que serão utilizados para a execução dos testes no sistema.

Do lado direito da tela principal temos três grandes grupos de funcionalidades. O primeiro grupo é o de gerenciar Plano de Teste, que refere-se às funcionalidades de criação dos planos de testes e *baselines*. O segundo grupo é a Execução do Teste, na qual o testador irá atribuir o valor da execução do caso de teste no sistema. Neste grupo ainda são exibidos os resultados dos testes através de relatórios, gráficos e métricas. Por último, existe a funcionalidade do Conteúdo do Plano de Teste, que será determinado quais casos de testes serão executados em determinado plano de teste e atribuí-los para os integrantes do projeto.

Na parte superior da tela é possível visualizar um menu de atalhos das principais funcionalidades do TestLink, como a Execução dos Testes, Especificação, Resultados da execução e Gerenciamento de Usuários.

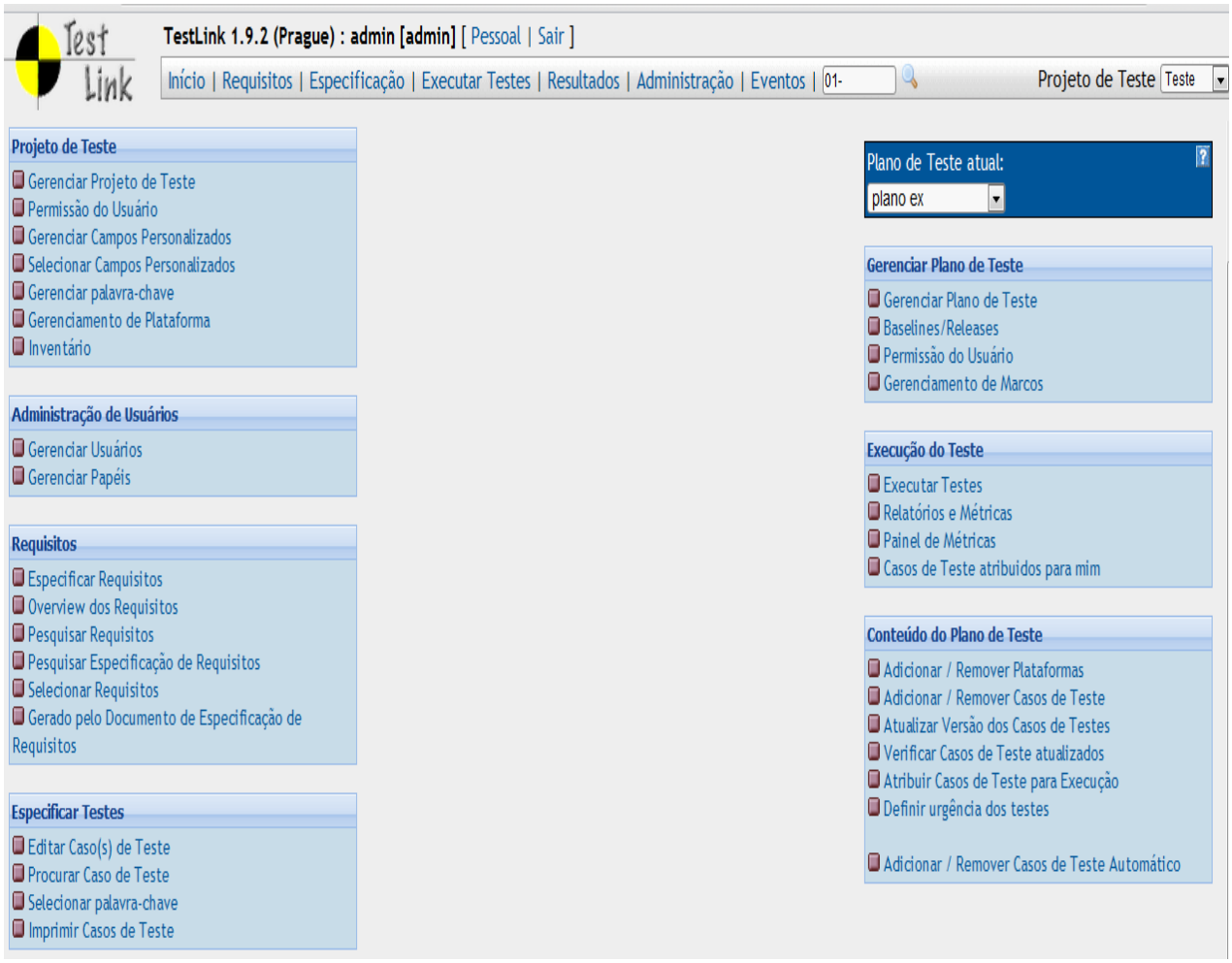


Figura 6 - Tela Inicial do TestLink.

Para que um novo projeto seja executado no TestLink, deve-se criar um Plano de Testes, que tem como objetivo organizar como serão executados os testes. No plano de testes, são descritos o escopo do sistema que será testado, as características a serem testadas, a definição do ambiente de execução, possíveis riscos para com a execução dos testes e ferramentas a serem utilizadas. Um plano de testes é formado por um conjunto de casos de teste, apresentados na Seção 1.

No TestLink, os casos de teste são organizados em suas respectivas suítes de testes, conceito discutido na Seção 2.3.3. As suítes de testes devem ser criadas para inserir os casos de testes relacionados à respectiva suíte. Depois de criado os casos de teste, estes são atribuídos aos respectivos testadores da equipe para os mesmos serem executados.

Durante a execução dos testes, os testadores realizam os passos que foram determinados nos casos de teste no sistema a ser testado e em seguida, atribuem um resultado da execução do teste. Os valores possíveis para esse resultados são os seguintes: **Passou**, caso o sistema se comportou da maneira correta que o caso de teste aborda; **Falhou**, quando o sistema se comporta de maneira inesperada; **Bloqueado**, caso o testador esteja impedido por algum motivo para executar o teste. A Figura 7 mostra a tela onde o testador registra esse resultado na ferramenta TestLink.

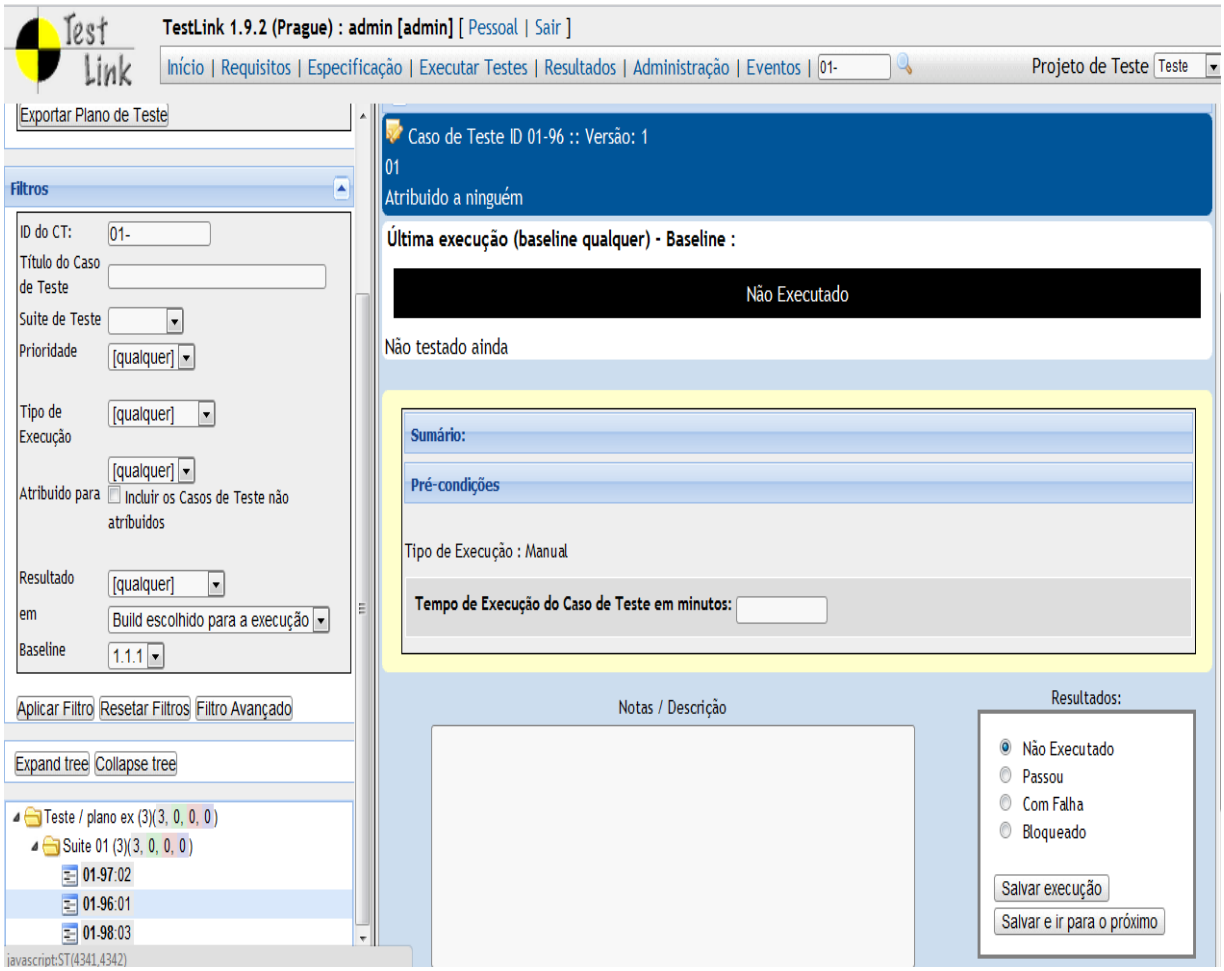


Figura 7 - Tela de Registro do Resultado da Execução de Um Caso de Teste

Como já foram abordados neste trabalho, os testes precisam ser executados novamente caso o software tenha sofrido novas alterações ou novas funcionalidades tenham sido criadas. Como na maioria das vezes não há tempo suficiente para executá-los, técnicas de seleção de casos de teste podem ser utilizadas para priorizar os principais casos de teste de um projeto baseado no histórico de execução dos mesmos. Entretanto essas técnicas não são disponibilizadas pelas principais ferramentas de gerenciamento de testes disponibilizadas pelo mercado, entre elas o Testlink.

Mesmo não disponibilizando nenhuma das técnicas, durante a execução dos testes, no TestLink são registradas informações que podem ser utilizados na seleção dos casos de teste com as técnicas de seleção. Um dos objetivos deste trabalho foi desenvolver no TestLink algumas dessas técnicas de seleção de casos de teste utilizando informações extraídas das execuções de testes na ferramenta TestLink. O objetivo principal da funcionalidade, explicada em detalhes na Seção 4.2, é a definição de um novo plano de testes

contendo os casos de testes selecionados e priorizados por uma das técnicas implementadas na ferramenta.

4.2. Desenvolvimento da Funcionalidade para Geração Automática do Plano de Teste de Regressão

Para desenvolver a funcionalidade utilizando os dados já registrados pelo Testlink, apresentado na Seção 4.1 foi realizado um levantamento das informações no banco de dados que a ferramenta registra na realização de ciclos de testes. As informações que foram identificadas foram: prioridade de um caso de teste, quantidade de falhas de um caso de teste, a partir da integração de uma ferramenta de *bugtracking* com o TestLink e a severidade de uma falha. Existem alguns algoritmos (por exemplo, *Risk Based Testing and Metrics* apresentadas no Capítulo 3, que utilizam a informação da quantidade de execuções de cada teste, no entanto, com a utilização de tal ferramenta esses dados não puderam ser adquiridos para ser calculada a Probabilidade da Severidade.

Alguns fatores foram determinantes para a escolha do TestLink para o desenvolvimento das técnicas de seleção automática de casos de testes para geração automática de um plano de teste de regressão: (a) a ferramenta possui dados históricos da execução dos testes; (b) é bastante utilizada por grandes empresas (McAfee, Intel, Symantec, Samsung, Yahoo, etc.) na automação dos mesmos e (c) é uma ferramenta *open source* (código fonte disponível).

O TestLink é um software desenvolvido na linguagem PHP (PHP, 1995) e utiliza como SGBD (Sistema de Gerenciamento de Banco de Dados) o MySQL (MYSQL, 2005). Durante a construção da funcionalidade, foram utilizadas algumas ferramentas, tais como: Netbeans para a linguagem PHP (NETBEANS, 2011), a ferramenta MySQL e o servidor web WampServer (WAMPSERVER, 2010). A Figura 8 mostra a tela da funcionalidade que foi desenvolvida para geração automática do plano de teste de regressão.

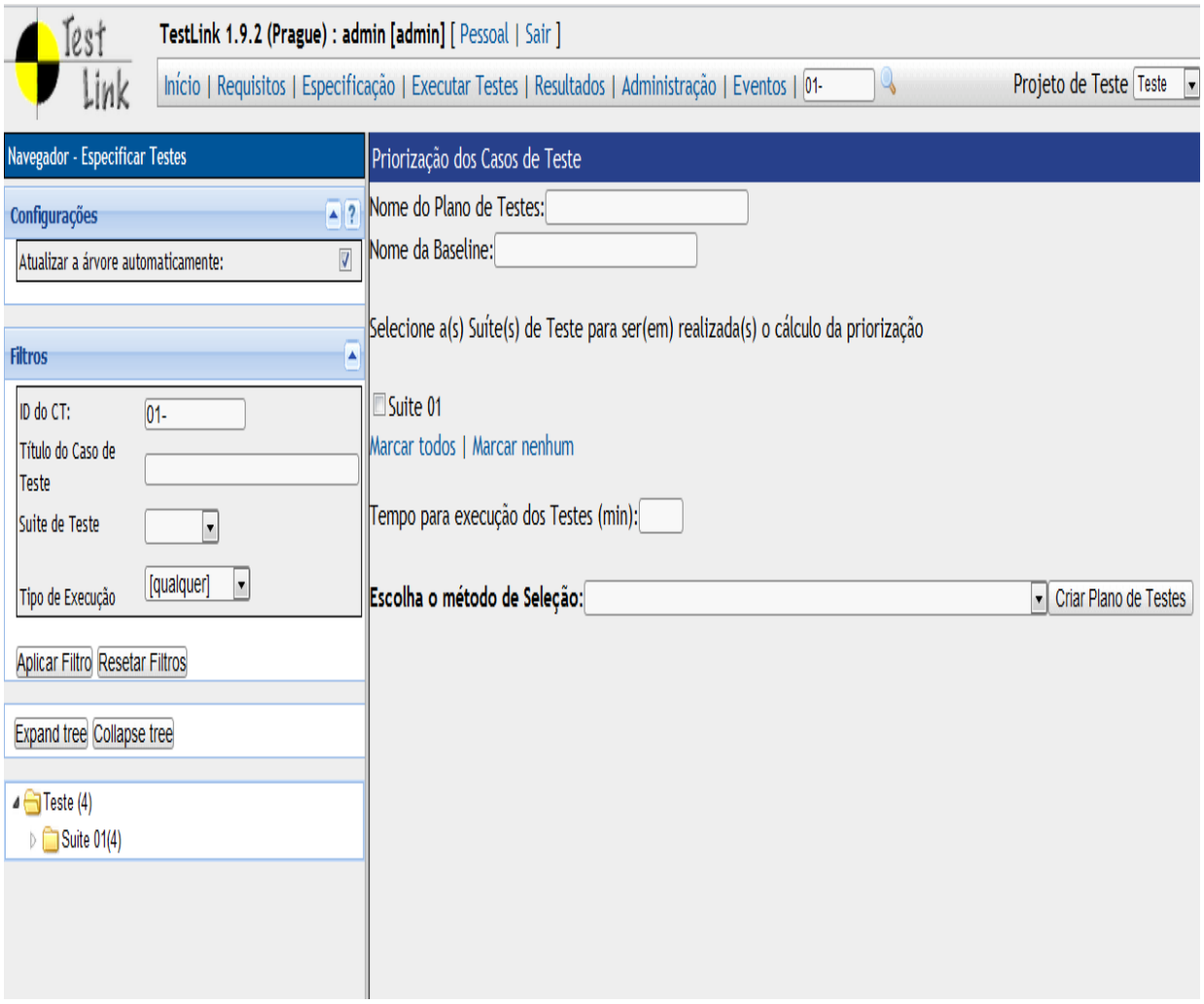


Figura 8 - Tela da Funcionalidade desenvolvida.

Para realizar a criação automática do plano de teste, deve-se inicialmente definir o nome do novo Plano de Testes e um nome da *Baseline*. Em seguida, deve-se escolher as suítes de teste do projeto que serão consideradas na aplicação das técnicas implementadas. É possível ainda definir o valor do tempo disponível para execução do teste de regressão. Finalmente deve-se escolher uma das três técnicas implementadas.

Foram desenvolvidos os seguintes algoritmos para realizar a seleção dos casos de teste: *Risk Based Testing and Metrics*; Estratégias de Seleção de Testes Baseado em Riscos e *Test Case Prioritization for Regression Testing based on Severity of Fault.*

O algoritmo *Risk Based Testing and Metrics*, apresentada na Seção 3.1.1, utiliza a quantidade de execuções de um teste para calcular a ordem de priorização dos casos de teste. Para o cálculo da probabilidade da severidade, a funcionalidade utiliza a quantidade de erros

encontrados em cada caso de teste, dividido pela quantidade de execuções do mesmo e multiplicado pela severidade do caso de teste.

Já o algoritmo Estratégias de Seleção de Testes Baseado em Riscos, apresentado na Seção 3.1.2, avalia a partir das partes mais críticas e mais importantes do sistema através de fatores que são escolhidos como critérios para priorização. Na funcionalidade desenvolvida na ferramenta, os fatores que foram utilizados foram a Prioridade e a Severidade de cada caso de teste. Para realizar o cálculo do algoritmo, é necessário atribuir o valor do peso para cada fator escolhido. Para o desenvolvimento da aplicação, o valor máximo (peso = 10) foi atribuído para os fatores que devem ser analisados em cada funcionalidade do sistema, pois ambos os fatores tem uma forte influência na seleção dos testes.

O terceiro algoritmo desenvolvido foi o *Test Case Prioritization for Regression Testing based on Severity of Fault*, apresentado na Seção 3.1.3, também utiliza como entrada para execução de seu algoritmo, a quantidade de execuções. O cálculo da taxa de detecção de falhas (RFTi) é realizado a partir da quantidade de erros que é encontrado em cada caso de teste, da quantidade de execuções do mesmo e da Severidade de cada caso de teste quando o mesmo falhou durante sua execução no sistema.

Para executar a funcionalidade desenvolvida, o usuário do sistema irá escolher um desses algoritmos para realizar a seleção dos casos de teste nas suítes selecionadas. No novo Plano de Teste criado, irá conter os casos de teste que o algoritmo selecionou. A Figura 9 mostra um exemplo do novo plano de teste criado a partir da utilização da funcionalidade desenvolvida.

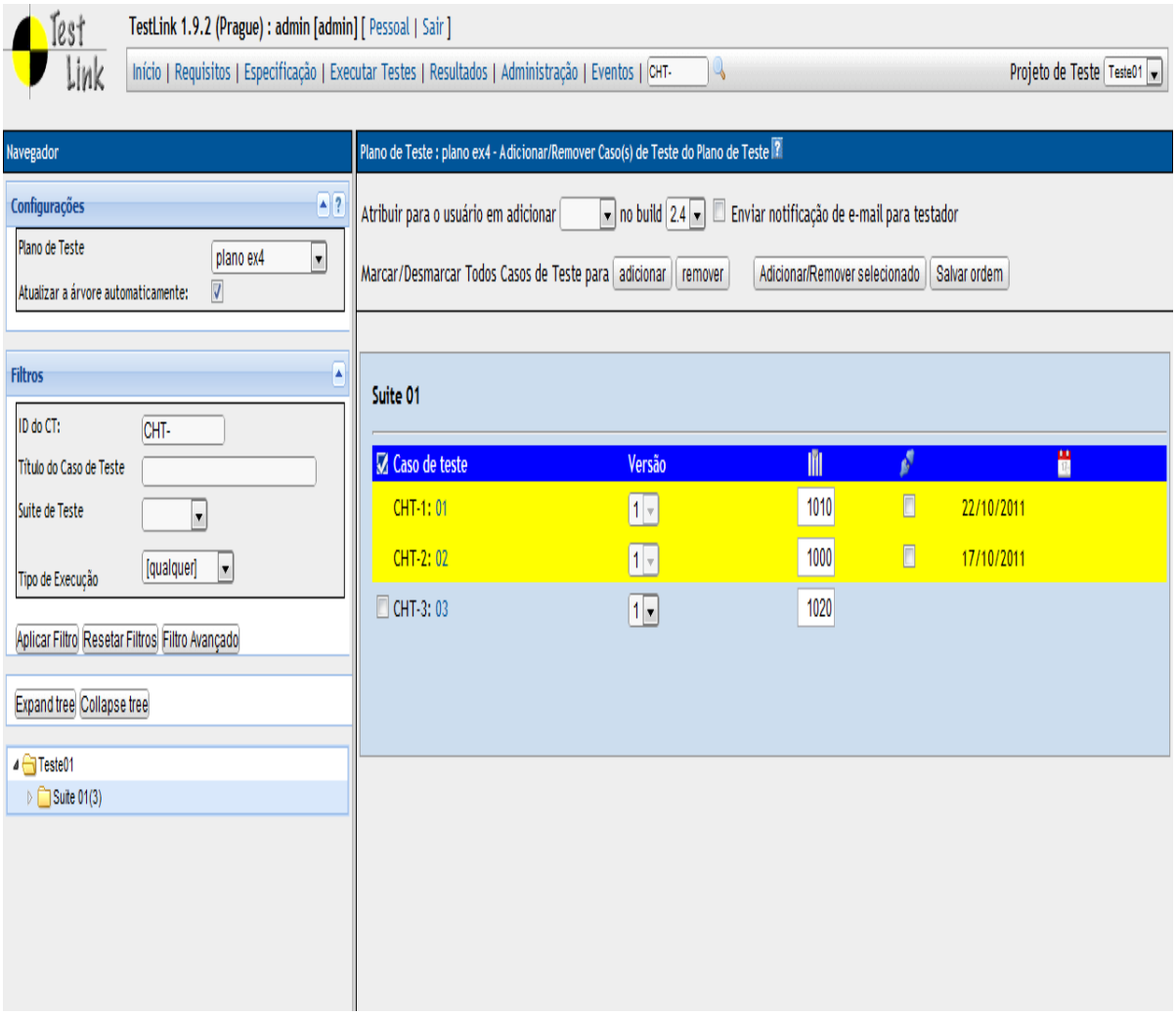


Figura 9 - Exemplo de casos de teste selecionados.

Na Figura 9 percebe-se que na Suíte01 o Caso de Teste 03 não foi selecionado pelo algoritmo de priorização, pois ele não se encontra selecionado, ou seja, este caso de teste não foi selecionado a partir do tempo disponível para realizar o novo ciclo de testes. Na mesma tela também exibe a ordem de priorização dos casos de teste, o caso de teste mais prioritário é o 02, pois o valor da prioridade é 1000, enquanto o caso de teste 01 tem valor 1010. Ainda nesta tela, o usuário já tem a opção de atribuir os casos de teste selecionados pelos algoritmos para algum testador da equipe do projeto desenvolvido.

5. ANÁLISE COMPARATIVA ENTRE AS TÉCNICAS DE SELEÇÃO DE CASOS DE TESTE

O objetivo deste capítulo é apresentar o experimento que foi realizado com a finalidade de avaliar a aplicação desenvolvida neste trabalho.

Na Seção 5.1 é mostrado o cenário do projeto de software em que a funcionalidade desenvolvida foi utilizada. Na Seção 5.2 é relatado como o experimento foi realizado no projeto de desenvolvimento de software. Já na Seção 5.3 é exibido os resultados do estudo de caso para cada algoritmo desenvolvido. Na Seção 5.4 contém a análise dos resultados para cada técnica que foi desenvolvida no trabalho. Finalmente, na Seção 5.5 é mostrada algumas considerações sobre os resultados do estudo de caso.

5.1. Contexto do Projeto Utilizado

O cenário do CPD (Centro de Processamento de Dados) da Universidade Federal de Sergipe é formado por vários projetos de desenvolvimento de sistemas. Um dos módulos existentes é o de Produção Intelectual, que integra o sistema SIGAA (Sistema Integrado de Gestão de Assuntos Acadêmicos). Este módulo tem como objetivo cadastrar as produções realizadas pelos docentes da universidade, a fim de oferecer ao gestor do departamento de Extensão, relatórios gerenciais para disponibilizar bolsas de Extensão para os professores que possuam uma maior quantidade de produções desenvolvidas pelos mesmos.

Foi realizado um estudo de caso no módulo Produção Intelectual com a funcionalidade desenvolvida no TestLink. Inicialmente foram criados os casos de teste do módulo que seria testado, sendo um total de trinta e nove casos de teste. Foi utilizada uma técnica funcional denominada de testes exploratórios para execução dos testes, apresentada na Seção 2.3.4.

Para realizar os testes de regressão no CPD, são escolhidos os testes que tem maior prioridade apenas em relação ao negócio, definidos pelos analistas das equipes de cada projeto de desenvolvimento, pois devido a prazos curtos, seria inviável executar todos os testes novamente.

Cientes de que uma seleção mais efetiva pode trazer maior economia de tempo e custo para a organização, perceberam a necessidade de se utilizar outros critérios de seleção de testes além do que já era utilizado, para garantir uma maior probabilidade de chance de serem encontrados erros e que ofereçam um menor risco para o negócio.

5.2. Relato do Experimento

Para realizar o experimento, foi selecionado o módulo Produção Intelectual, que contém trinta e nove casos de teste organizados em nove suítes de teste. Um caso de teste ao ser elaborado é classificado de acordo com algumas características que foram úteis para a utilização dos critérios de seleção de casos de teste. Foram utilizadas informações como prioridade, severidade de uma falha, quantidade de erros encontrados por cada caso de teste e tipos de testes (positivos e negativos). A Figura 8 mostra as informações no TestLink que devem ser informadas no projeto do caso de teste.

Para a realização do experimento, utilizaram-se os dados referentes ao primeiro ciclo de testes do módulo que foi escolhido. Depois de ter coletado todos os dados, foram executados os três algoritmos que foram desenvolvidos na funcionalidade criada, explicada na Seção 3.1 e foi gerada uma lista de todos os casos de teste em ordem decrescente de prioridade dos testes.

No experimento realizado todas as suítes de teste foram escolhidas para a aplicação das três técnicas implementadas.

5.3. Resultados do Estudo de Caso

A seguir será mostrado o resultado da seleção dos casos de teste do projeto escolhido, para cada algoritmo que foi desenvolvido. As técnicas desenvolvidas foram as seguintes: *Risk Based Testing and Metrics*, apresentada na Seção 3.1.1; Estratégias de Seleção de Testes Baseado em Riscos, mostrada na Seção 3.1.2 *e Test Case Prioritization for Regression Testing based on Severity of Fault*, apresentada na Seção 3.1.4.

Para cada algoritmo desenvolvido será mostrada uma tabela com a ordem decrescente de priorização dos testes.

Em cada tabela é mostrado o nome do caso de teste, o valor da severidade, prioridade do caso de teste, o total do cálculo de cada algoritmo, o tipo de teste, a quantidade

de erros encontrados em cada caso de teste e o tempo de execução em minutos. Ou seja, em cada tabela, são mostrados os critérios que cada um utiliza para a execução do cálculo da prioridade de cada caso de teste. As técnicas *Risk Based Testing and Metrics* e *Test Case Prioritization for Regression Testing based on Severity of Fault* não foram desenvolvidas exatamente o algoritmo original, pois os mesmos utilizam o critério de quantidade de execuções e esta informação não é disponível no TestLink.

Risk Based Testing and Metrics

O cálculo desse algoritmo é determinado a partir da utilização de determinadas informações históricas. Os critérios são: prioridade, severidade, quantidade de falhas por cada caso de teste e tipos de testes. A

Tabela 19 mostra o resultado do cálculo do algoritmo em ordem decrescente a partir do total do cálculo.

ID	Caso de Teste	Tipo de Teste	Severidade	Prioridade	Quantidade de Erros	Total	Tempo	Posição
288	Cadastro de Exposição ou Apresentação Artística	9	2	3	6	45	210	1
294	Cadastrar Trabalhos de Conclusão	9	2	3	6	45	150	2
290	Listar, Alterar e Excluir Exposição ou Apresentação Artística	9	2	3	6	45	150	3
298	Cadastrar, Alterar, Excluir e Listar Programação Visual	1	2	3	7	43	120	4
345	Cadastrar, Alterar, Excluir e Listar uma Orientação de Pós-Graduação.	9	2	3	5	39	150	5
292	Cadastrar, Alterar, Excluir e Listar Montagens	1	2	3	6	37	120	6
296	Listar, Alterar e Excluir Trabalhos de Conclusão	9	2	3	4	33	60	7
285	Listar, Alterar e Excluir presentações Audio Visuais	9	2	3	4	33	60	8
ID	Caso de Teste	Tipo	Severi	Priori	Quantida	Total	Tempo	Posição

		de Teste	dade	dade	de de Erros			
283	Cadastro de Apresentações Audio Visuais	9	2	3	4	33	150	9
343	Cadastrar, Alterar, Excluir e Listar um Trabalho Final de Curso.	9	1	3	7	30	150	10
347	Cadastrar, Alterar, Excluir e Listar uma Orientação de Bolsista de Iniciação Científica.	9	2	3	3	27	120	11
306	Cadastrar, Alterar, Excluir e Listar Patentes.	1	2	3	4	25	150	12
142	Listar Alterar e Excluir Produções	9	2	3	2	21	35	13
363	Validação das Produções Acadêmicas.	9	2	3	2	21	130	14
300	Cadastrar Comissões Julgadoras	9	2	3	2	21	150	15
361	Importar produções do Lattes.	9	2	3	2	21	90	16
349	Cadastrar, Alterar, Excluir e Listar uma Orientação de Estágio.	9	1	2	5	19	90	17
358	Consulta ao Acervo Digital	9	1	3	3	18	120	18
152	Cadastro de Livros	9	2	3	1	15	45	19
304	Listar, Alterar e Excluir Comissões Julgadoras	9	2	3	1	15	60	20
356	Cadastrar, Alterar, Excluir e Listar Chefia	9	1	1	6	15	210	21
146	Cadastro de Capítulo	9	2	3	1	15	55	22
50	Cadastro de artigos	9	2	3	1	15	120	23
150	Listar, Alterar e Excluir Capítulos	9	2	3	1	15	60	24
154	Listar, Alterar e Excluir Livros	9	2	3	1	15	45	25
334	Cadastrar, Alterar, Excluir e Listar Bolsas Obtidas.	9	1	1	4	13	150	26
354	Cadastrar, Alterar, Excluir e Listar Qualificação do Docente.	9	1	1	4	13	90	27
ID	Caso de Teste	Tipo	Severi	Priori	Quantida	Total	Tempo	Posição

		de Teste	dade	dade	de de Erros			
302	**Cadastrar, Alterar, Excluir e Listar Maquetes, Protótipos, Softwares e Outros**	**1**	**1**	**3**	**4**	**13**	**150**	**28**
156	**Cadastro de Participação em eventos**	**9**	**1**	**1**	**2**	**11**	**120**	**29**
160	**Cadastro de Texto Didático**	**9**	**1**	**1**	**2**	**11**	**65**	**30**
309	**Cadastrar, Alterar, Excluir e Listar Prêmio Recebido.**	**9**	**1**	**1**	**2**	**11**	**240**	**31**
158	**Listar, Alterar e Excluir Participações em Eventos**	**9**	**1**	**1**	**2**	**11**	**90**	**32**
162	**Listar, Alterar e Excluir Texto Didático**	**9**	**1**	**1**	**2**	**11**	**60**	**33**
338	**Cadastrar, Alterar, Excluir e Listar Sociedades Científicas e Culturais.**	**9**	**1**	**1**	**2**	**11**	**90**	**34**
365	**Consolidar as Validações das Produções do Chefe.**	**9**	**2**	**1**	**1**	**11**	**90**	**35**
340	**Cadastrar, Alterar, Excluir e Listar Participação em Colegiados e Comissões**	**9**	**1**	**1**	**1**	**10**	**270**	**36**
336	**Cadastrar, Alterar, Excluir e Listar Organização de Eventos.**	**9**	**1**	**1**	**1**	**10**	**90**	**37**
351	**Cadastrar, Alterar, Excluir e Listar Mini-Curso.**	**1**	**1**	**2**	**3**	**7**	**90**	**38**
332	**Cadastrar, Alterar, Excluir e Listar Visitas Científicas.**	**1**	**1**	**1**	**2**	**3**	**210**	**39**

Tabela 19- Resultado do algoritmo *Risk Based Testing and Metrics*

O cálculo do algoritmo para o caso de teste de Id igual a 288 que se encontra na posição 1, foi determinado como o mais prioritário já que este possui uma prioridade máxima, um valor de severidade médio e uma alta quantidade de erros em sua execução. O algoritmo utiliza estes critérios ditos anteriormente como sendo os mais relevantes para selecionar um teste.

O caso de teste de Id igual a 361 que se encontra na posição 16 possui exatamente os mesmos valores para os critérios prioridade e severidade do caso de teste de Id igual a 288. Houve uma considerável diferença entre suas posições devido à quantidade de erros que foram encontrados em sua execução, já que o mesmo algoritmo trata esse critério como sendo um atributo relevante para a realização da seleção dos casos de teste.

Outro caso de teste de Id igual a 332 foi determinado pelo algoritmo como o de menor prioridade para um ciclo de regressão, pois possui valores mínimos com relação à prioridade, severidade e tipo de teste. Além disso, apenas 1 erro foi encontrado em sua execução. E realmente, tal caso de teste não possui tanta importância com relação ao valor de negócio.

Test Case Prioritization for Regression Testing based on Severity of Fault

O algoritmo *Test Case Prioritization for Regression Testing based on Severity of Fault*, utiliza os seguintes critérios para a realização do cálculo da seleção dos casos de teste: quantidade de erros encontrados em cada caso de teste e a severidade de uma falha que foram explicados na Seção 3.1.4. A Tabela 20 mostra a lista em ordem decrescente de prioridade dos casos de teste.

ID	Caso de Teste	Severidade	Quantidade de Erros	Tempo (min)	Total	Posição
298	Cadastrar, Alterar, Excluir e Listar Programação Visual	2	7	120	90	1
288	Cadastro de Exposição ou Apresentação Artística	2	6	210	80	2
292	Cadastrar, Alterar, Excluir e Listar Montagens	2	6	120	80	3
290	Listar, Alterar e Excluir Exposição ou Apresentação Artística	2	6	150	80	4
294	Cadastrar Trabalhos de Conclusão	2	6	150	80	5
343	Cadastrar, Alterar, Excluir e Listar um Trabalho Final de Curso.	1	7	150	80	6
345	Cadastrar, Alterar, Excluir e Listar uma Orientação de Pós-Graduação.	2	5	150	70	7

ID	Caso de Teste	Severidade	Quantidade de Erros	Tempo (min)	Total	Posição
356	Cadastrar, Alterar, Excluir e Listar Chefia	1	6	210	70	8
285	Listar, Alterar e Excluir Apresentações Audio Visuais	2	4	60	60	9
283	Cadastro de Apresentações Audio Visuais	2	4	150	60	10
349	Cadastrar, Alterar, Excluir e Listar uma Orientação de Estágio.	1	5	90	60	11
306	Cadastrar, Alterar, Excluir e Listar Patentes.	2	4	150	60	12
296	Listar, Alterar e Excluir Trabalhos de Conclusão	2	4	60	60	13
354	Cadastrar, Alterar, Excluir e Listar Qualificação do Docente.	1	4	90	50	14
347	Cadastrar, Alterar, Excluir e Listar uma Orientação de Bolsista de Iniciação Científica.	2	3	120	50	15
334	Cadastrar, Alterar, Excluir e Listar Bolsas Obtidas.	1	4	150	50	16
302	Cadastrar, Alterar, Excluir e Listar Maquetes, Protótipos, Softwares e Outros	1	4	150	50	17
363	Validação das Produções Acadêmicas.	2	2	130	40	18
351	Cadastrar, Alterar, Excluir e Listar Mini-Curso.	1	3	90	40	19
358	Consulta ao Acervo Digital	1	3	120	40	20
300	Cadastrar Comissões Julgadoras	2	2	150	40	21
142	Listar Alterar e Excluir Produções	2	2	35	40	22
361	Importar produções do Lattes.	2	2	90	40	23
365	Consolidar as Validações das Produções do Chefe.	2	1	90	30	24
162	Listar, Alterar e Excluir Texto Didático	1	2	60	30	25
146	Cadastro de Capítulo	2	1	55	30	26
152	Cadastro de Livros	2	1	45	30	27
338	Cadastrar, Alterar, Excluir e Listar Sociedades Científicas e Culturais.	1	2	90	30	28
332	Cadastrar, Alterar, Excluir e Listar Visitas Científicas.	1	2	210	30	29
309	Cadastrar, Alterar, Excluir e	1	2	240	30	30

	Listar Prêmio Recebido.					
154	**Listar, Alterar e Excluir Livros**	**2**	**1**	**45**	**30**	**31**
50	**Cadastro de artigos**	**2**	**1**	**120**	**30**	**32**
158	**Listar, Alterar e Excluir Participações em Eventos**	**1**	**2**	**90**	**30**	**33**
160	**Cadastro de Texto Didático**	**1**	**2**	**65**	**30**	**34**
156	**Cadastro de Participação em eventos**	**1**	**2**	**120**	**30**	**35**
304	**Listar, Alterar e Excluir Comissões Julgadoras**	**2**	**1**	**60**	**30**	**36**
150	**Listar, Alterar e Excluir Capítulos**	**2**	**1**	**60**	**30**	**37**
340	**Cadastrar, Alterar, Excluir e Listar Participação em Colegiados e Comissões**	**1**	**1**	**270**	**20**	**38**
336	**Cadastrar, Alterar, Excluir e Listar Organização de Eventos.**	**1**	**1**	**90**	**20**	**39**

Tabela 20- Resultado do algoritmo *Test Case Prioritization for Regression Testing based on Severity of Fault*

A partir do cálculo do algoritmo *Test Case Prioritization for Regression Testing based on Severity of Fault*, foi determinado que o caso de teste de Id igual a 298 possui uma maior prioridade em relação aos demais. Tal algoritmo chegou a este resultado ao analisar os critérios de severidade e quantidade de erros. Em tal caso de teste foi encontrado muitos erros em sua execução e o mesmo possui um valor da severidade considerável (Severidade = 2). E realmente, tal caso de teste possui um importante valor de negócio para o sistema no qual foi realizado os testes.

O caso de teste de Id igual a 363 possui o mesmo valor para o critério de Severidade do caso de teste analisado anteriormente, o que diferiu foi a quantidade de erros encontrados. Neste foi encontrado apenas 2 erros durante sua execução. Como este algoritmo utiliza apenas esses dois critérios como relevantes, tal caso de teste foi selecionado para a posição 18.

Já o caso de teste de Id igual a 336 que foi encontrado apenas 1 erro e possui valor de severidade mínimo, o algoritmo determinou o mesmo como sendo o de menor prioridade para um possível ciclo de regressão. E realmente, tal caso de teste não possui uma importância tão relevante, caso alguma falha ocorra nessa funcionalidade com o sistema em produção, não haverá um grande impacto para o negócio.

Estratégias de Seleção de Testes Baseado em Riscos

O algoritmo Estratégias de Seleção de Testes Baseado em Riscos seleciona os seguintes critérios para realizar o cálculo da seleção dos casos de teste: severidade de uma falha e prioridade de um caso de Teste. Para cada fator escolhido, o valor é multiplicado por um peso. Na Tabela 21 é exibida a lista decrescente a partir do valor do cálculo do algoritmo de seleção.

ID	Caso de Teste	Severidade	Prioridade	Tempo (min)	Total	Posição
50	Cadastro de artigos	2	3	120	50	1
152	Cadastro de Livros	2	3	45	50	2
285	Listar, Alterar e Excluir Apresentações Audio Visuais	2	3	60	50	3
296	Listar, Alterar e Excluir Trabalhos de Conclusão	2	3	60	50	4
363	Validação das Produções Acadêmicas.	2	3	130	50	5
142	Listar Alterar e Excluir Produções	2	3	35	50	6
154	Listar, Alterar e Excluir Livros	2	3	45	50	7
288	Cadastro de Exposição ou Apresentação Artística	2	3	210	50	8
298	Cadastrar, Alterar, Excluir e Listar Programação Visual	2	3	120	50	9
300	Cadastrar Comissões Julgadoras	2	3	150	50	10
345	Cadastrar, Alterar, Excluir e Listar uma Orientação de Pós-Graduação.	2	3	150	50	11
146	Cadastro de Capítulo	2	3	55	50	12
292	Cadastrar, Alterar, Excluir e Listar Montagens	2	3	120	50	13
290	Listar, Alterar e Excluir Exposição ou Apresentação Artística	2	3	150	50	14
304	Listar, Alterar e Excluir Comissões Julgadoras	2	3	60	50	15
347	Cadastrar, Alterar, Excluir e Listar uma Orientação de Bolsista de Iniciação Científica.	2	3	120	50	16
150	Listar, Alterar e Excluir Capítulos	2	3	60	50	17
283	Cadastro de Apresentações Audio Visuais	2	3	150	50	18
294	Cadastrar Trabalhos de Conclusão	2	3	150	50	19
306	Cadastrar, Alterar, Excluir e Listar Patentes.	2	3	150	50	20
361	Importar produções do Lattes.	2	3	90	50	21
343	Cadastrar, Alterar, Excluir um Trabalho Final de Curso.	1	3	150	40	22

ID	Caso de Teste	Severidade	Prioridade	Tempo (min)	Total	Posição
302	Cadastrar, Alterar, Excluir e Listar Maquetes, Protótipos, Softwares e Outros	1	3	150	40	23
358	Consulta ao Acervo Digital	1	3	120	40	24
351	Cadastrar, Alterar, Excluir e Listar Mini-Curso.	1	2	90	30	25
365	Consolidar as Validações das Produções do Chefe.	2	1	90	30	26
349	Cadastrar, Alterar, Excluir e Listar uma Orientação de Estágio.	1	2	90	30	27
156	Cadastro de Participação em eventos	1	1	120	20	28
309	Cadastrar, Alterar, Excluir e Listar Prêmio Recebido.	1	1	240	20	29
334	Cadastrar, Alterar, Excluir e Listar Bolsas Obtidas.	1	1	150	20	30
158	Listar, Alterar e Excluir Participações em Eventos	1	1	90	20	31
336	Cadastrar, Alterar, Excluir e Listar Organização de Eventos.	1	1	90	20	32
356	Cadastrar, Alterar, Excluir e Listar Chefia	1	1	210	20	33
354	Cadastrar, Alterar, Excluir e Listar Qualificação do Docente.	1	1	90	20	34
160	Cadastro de Texto Didático	1	1	65	20	35
338	Cadastrar, Alterar, Excluir e Listar Sociedades Científicas e Culturais.	1	1	90	20	36
162	Listar, Alterar e Excluir Texto Didático	1	1	60	20	37
332	Cadastrar, Alterar, Excluir e Listar Visitas Científicas.	1	1	210	20	38
340	Cadastrar, Alterar, Excluir e Listar Participação em Colegiados e Comissões	1	1	270	20	39

Tabela 21- Resultado do algoritmo Estratégias de Seleção de Testes Baseado em Riscos

A partir do cálculo do seguinte algoritmo, foi determinado que o caso de teste de Id igual a 50, possui prioridade com relação aos demais, pois o mesmo possui valor correspondente à prioridade como sendo máximo e severidade um valor médio, já que tal algoritmo analisa apenas esses dois critérios para seleção. O caso de teste de Id 50 realmente possui uma prioridade bastante relevante para o negócio, já que uma falha nessa funcionalidade pode acarretar em um considerável impacto.

O caso de teste de Id igual 343 encontra-se na posição 22 determinada pelo algoritmo. Tal posição foi assim determinada pois o caso de teste possui um valor mínimo

para a severidade e um valor máximo para a prioridade. Então, pode-se concluir que, caso tal funcionalidade falhe com o sistema em produção, não irá sofrer nenhum impacto para o negócio.

Já o caso de teste de Id igual a 340 não possui tanta importância comparado à grande maioria dos testes, já que o mesmo possui valores mínimos tanto para severidade quanto para a prioridade. E realmente, tal caso de teste não possui um considerável valor agregado para o negócio, já que caso uma falha ocorra nessa funcionalidade, não acarretará em grandes impactos.

5.4. Análise dos Resultados

O resultado do cálculo dos três algoritmos desenvolvidos foi comparado com os casos de teste que seriam mais prioritários em um Teste de Regressão, selecionados por um analista da equipe do projeto utilizado no estudo de caso. O analista da equipe apenas utiliza a prioridade de um teste em relação ao negócio. Para realizar a análise, foram observados os 20 primeiros casos de teste que os algoritmos exibiram em seus resultados. A seguir será feita uma análise dos resultados de cada método de seleção de testes desenvolvido neste trabalho.

- ***Risk Based Testing and Metrics***

Ao verificar os resultados do primeiro algoritmo (*Risk Based Testing and Metrics*) na Tabela 18, foi observado que dos vinte primeiros casos de teste, apenas dois não teriam nenhuma prioridade quando comparada à seleção realizada pelo analista da equipe do projeto de desenvolvimento de software. Então, a partir dessa análise, o primeiro algoritmo desenvolvido obteve uma similaridade de 90% na comparação realizada com os casos de testes do analista da equipe.

Os casos de teste divergentes dos selecionados pelo membro da equipe foram selecionados pela ferramenta, pois a quantidade de erros encontrados nos mesmos foi alta em relação aos demais testes. Como o algoritmo leva em consideração a quantidade de erros em um caso de teste, então os mesmos obtiveram uma elevada pontuação.

Portanto, o algoritmo pode selecionar algum caso de teste que não tenha uma alta importância em relação à prioridade em relação ao negócio, mas que tenham ocorrido muitas falhas em seu histórico de execução.

Este resultado tem uma grande relevância, já que o algoritmo selecionou os testes que tenham maior risco de encontrar erros, que tenham uma grande importância para o negócio e que possuam uma alta severidade caso uma falha aconteça. Utilizá-los prioritariamente em um ciclo de regressão, diminuiria a chance de encontrar erros com o sistema em ambiente de produção, podendo gerar impactos negativos para o negócio.

A partir de uma análise realizada pelo analista de testes do projeto, este verificou que os testes selecionados, realmente possuem uma relevante importância tanto para o negócio quanto para possuir uma maior chance de encontrar novos erros nessas funcionalidades. Além disso, utilizar apenas um critério de seleção, como era anteriormente utilizado, não garante totalmente uma redução do impacto negativo que uma falha possa causar para o negócio.

O analista de testes da equipe priorizaria todos os casos de teste que tenham valores de prioridade e severidade elevados, como exemplo, o caso de teste de Id 294, o mesmo possui valor 3 e 2 de prioridade e severidade respectivamente. Além destes critérios, o algoritmo utiliza a quantidade de erros encontrados. Portanto, esta técnica seleciona além dos testes que tenham alta prioridade e severidade, os testes que falharam na maioria das vezes durante a execução.

- ***Test Case Prioritization for Regression Testing based on Severity of Fault***

Na comparação realizada com o segundo algoritmo desenvolvido (*Test Case Prioritization for Regression Testing based on Severity of Fault*), verificou-se que o algoritmo selecionou dezesseis casos de teste que seriam prioritários, ou seja, foram selecionados quatro casos de teste que não possuem nenhuma prioridade. Então, o algoritmo teve uma similaridade de 80% em relação a lista desenvolvida pelo analista da equipe. O Gráfico 1 resume os resultados mostrando a similaridade entre os resultados que o algoritmo selecionou em relação aos selecionados pelo membro da equipe.

Tal algoritmo também selecionou alguns casos de teste não prioritários pelo mesmo motivo da técnica explicada anteriormente. Foram selecionados alguns testes pelo motivo de terem sido encontrados muitos erros em execuções anteriores. Como o algoritmo utiliza também como critério a quantidade de erros, então este selecionou alguns casos de teste que falharam na maioria de suas execuções.

O resultado deste algoritmo possui certa relevância, já que este selecionou os testes que possuam uma grande chance de falhar e que gerem um impacto negativo caso uma falha venha acontecer em futuras execuções. Utilizá-los em um ciclo de regressão, diminuiria a chance de encontrar erros com o sistema em ambiente de produção, podendo gerar impactos negativos para o negócio.

Como já foi dito anteriormente, o analista de testes prioriza os testes com maior importância para o negócio. No entanto, este algoritmo prioriza um teste através do valor de sua severidade, ou seja, o impacto que uma falha pode causar ao negócio e, além disso, verifica a quantidade de erros durante a execução.

- **Estratégias de Seleção de Testes Baseado em Riscos**

Por último, o resultado do algoritmo Estratégias de Seleção de Testes Baseado em Riscos, apresentado no Gráfico 1 obteve 100% similaridade com a lista do analista da equipe. Todos os casos de teste selecionados possuem relevância para o negócio e devem ser priorizados em um novo ciclo de testes.

O resultado da similaridade entre os resultados é justificado pelos critérios utilizados pelo algoritmo: severidade e prioridade de um teste em relação ao negócio que são os mesmos utilizados pelo analista de testes da equipe.

Esta técnica de seleção chegou a resultados mais próximos dos quais o analista de testes selecionaria, já que este algoritmo utiliza apenas como critérios de seleção, a prioridade e severidade de uma falha. No entanto, diferentemente das demais técnicas já analisadas, esta não aborda a probabilidade de chance de um caso de teste falhar em novas execuções.

Com relação à priorização dos testes, este algoritmo analisa apenas os critérios da severidade e prioridade de um teste, de maneira idêntica aos testes que o analista priorizaria, observando ambos os critérios.

O Gráfico 1 mostra um resumo dos resultados da execução dos três algoritmos desenvolvidos, a quantidade de casos de teste selecionados por um integrante da equipe e a quantidade de casos de teste prioritários gerados pela ferramenta.

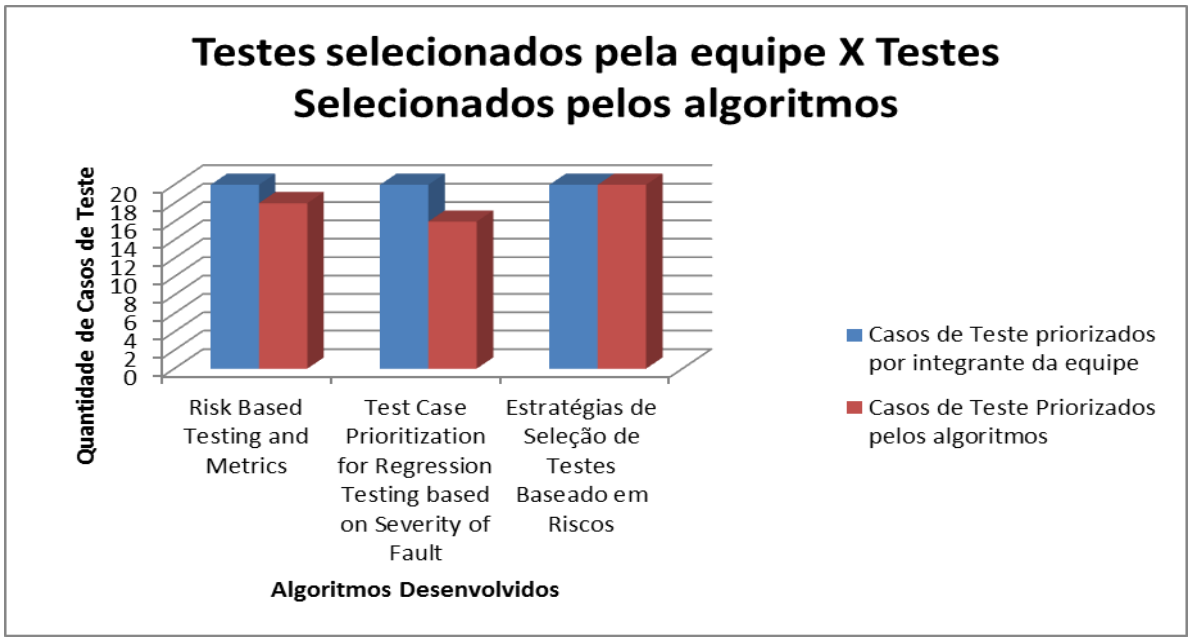


Gráfico 1 - Testes selecionados pela equipe X Testes Selecionados pelos algoritmos

A Tabela 22 mostra o comparativo entre as prioridades das três técnicas de seleção de caso de teste desenvolvidas nesse trabalho e a lista de prioridades analisada pelo analista de testes da equipe do projeto. O comparativo é realizado através da posição que cada técnica priorizou e que foi apresentado na Seção 3.3.

Técnicas de Seleção / Caso de Teste	Risk Based Testing and Metrics	Test Case Prioritization for Regression Testing based on Severity of Fault	Estratégias de Seleção de Testes Baseado em Riscos	Analista de Testes da Equipe do Projeto
Cadastro de Exposição ou Apresentação Artística	**1**	**2**	**8**	**8**
Cadastrar Trabalhos de Conclusão	**2**	**5**	**19**	**22**
Listar, Alterar e Excluir Exposição ou Apresentação Artística	**3**	**4**	**14**	**9**
Cadastrar, Alterar, Excluir e Listar Programação Visual	**4**	**1**	**9**	**10**

Técnicas de Seleção / Caso de Teste	Risk Based Testing and Metrics	Test Case Prioritization for Regression Testing based on Severity of Fault	Estratégias de Seleção de Testes Baseado em Riscos	Analista de Testes da Equipe do Projeto
Cadastrar, Alterar, Excluir e Listar uma Orientação de Pós-Graduação.	5	7	11	6
Cadastrar, Alterar, Excluir e Listar Montagens	6	3	13	18
Listar, Alterar e Excluir Trabalhos de Conclusão	7	13	4	21
Listar, Alterar e Excluir Apresentações Audio Visuais	8	9	3	4
Cadastro de Apresentações Audio Visuais	9	10	18	13
Cadastrar, Alterar, Excluir e Listar um Trabalho Final de Curso.	10	6	22	14
Cadastrar, Alterar, Excluir e Listar uma Orientação de Bolsista de Iniciação Científica.	11	15	16	7
Cadastrar, Alterar, Excluir e Listar Patentes.	12	12	20	2
Listar Alterar e Excluir Produções	13	22	6	5
Validação das Produções Acadêmicas.	14	18	5	12
Cadastrar Comissões Julgadoras	15	21	10	15
Importar produções do Lattes.	16	23	21	16
Cadastrar, Alterar, Excluir e Listar uma Orientação de Estágio.	17	11	27	27
Consulta ao Acervo Digital	18	20	24	17
Cadastro de Livros	19	27	2	3
Listar, Alterar e Excluir Comissões Julgadoras	20	36	15	19
Cadastrar, Alterar, Excluir e Listar Chefia	21	8	33	20
Cadastro de Capítulo	22	26	12	11
Cadastro de artigos	23	32	1	1
Listar, Alterar e Excluir Capítulos	24	37	17	23
Listar, Alterar e Excluir Livros	25	31	7	24

Técnicas de Seleção / Caso de Teste	Risk Based Testing and Metrics	Test Case Prioritization for Regression Testing based on Severity of Fault	Estratégias de Seleção de Testes Baseado em Riscos	Analista de Testes da Equipe do Projeto
Cadastrar, Alterar, Excluir e Listar Qualificação do Docente.	**27**	**14**	**34**	**29**
Cadastrar, Alterar, Excluir e Listar Maquetes, Protótipos, Softwares e Outros	**28**	**17**	**23**	**26**
Cadastro de Participação em eventos	**29**	**35**	**28**	**30**
Cadastro de Texto Didático	**30**	**34**	**35**	**31**
Cadastrar, Alterar, Excluir e Listar Prêmio Recebido.	**31**	**30**	**29**	**32**
Listar, Alterar e Excluir Participações em Eventos	**32**	**33**	**31**	**33**
Listar, Alterar e Excluir Texto Didático	**33**	**25**	**37**	**34**
Cadastrar, Alterar, Excluir e Listar Sociedades Científicas e Culturais.	**34**	**28**	**36**	**35**
Consolidar as Validações das Produções do Chefe.	**35**	**24**	**26**	**36**
Cadastrar, Alterar, Excluir e Listar Participação em Colegiados e Comissões	**36**	**38**	**39**	**37**
Cadastrar, Alterar, Excluir e Listar Organização de Eventos.	**37**	**39**	**32**	**38**
Cadastrar, Alterar, Excluir e Listar Mini-Curso.	**38**	**19**	**25**	**28**
Cadastrar, Alterar, Excluir e Listar Visitas Científicas.	**39**	**29**	**38**	**39**

Tabela 22 - Comparativo entre as Prioridades dos Casos de teste

O caso de teste “Cadastro de Exposição ou Apresentação Artística” é um caso de teste que tem uma alta prioridade e severidade e foi encontrada também uma alta quantidade de erros em sua execução. Então, os algoritmos *Risk Based Testing and Metrics* e *Test Case Prioritization for Regression Testing based on Severity of Fault*, que também utilizam a quantidade de erros como critério de seleção, este caso de teste teve alta prioridade, posições 1 e 2 respectivamente. Já o algoritmo Estratégias de Seleção de Testes Baseado em Riscos que apenas utiliza a prioridade e severidade como critérios, obteve igual valor em sua priorização comparado com a seleção feita pelo analista de testes, ambos na posição 8.

Outro exemplo do caso de teste “Cadastro de Apresentações Audio Visuais” é um caso de teste tem uma alta prioridade e foi encontrado menos erros do que o caso de teste mostrado anteriormente. Assim, os algoritmos *Risk Based Testing and Metrics* e *Test Case Prioritization for Regression Testing based on Severity of Fault*, que também utilizam a quantidade de erros como critério de seleção, obteve um valor de priorização inferior ao caso de teste anterior, posições 9 e 10 respectivamente, já que foram encontrados menos erros. Já que o algoritmo Estratégias de Seleção de Testes Baseado em Riscos e a seleção feita pelo analista de testes utiliza apenas os critérios de prioridade e severidade, priorizaram o caso de teste nas posições 18 e 13 respectivamente.

O caso de teste “Listar, Alterar e Excluir Capítulos” apenas foi encontrado 1 erro em sua execução e o mesmo também possui uma alta prioridade relacionado ao negócio. Então, os algoritmos *Risk Based Testing and Metrics* e *Test Case Prioritization for Regression Testing based on Severity of Fault*, não priorizaram efetivamente o caso de teste, posições 24 e 37 respectivamente, pois foi encontrado apenas um erro no primeiro ciclo de testes. Já o algoritmo Estratégias de Seleção de Testes Baseado em Riscos e a seleção realizada pelo analista de testes da equipe, priorizou tal caso de teste em uma posição inferior aos casos de teste apresentados anteriormente, já que este não possui um valor tão significativo, comparado com os anteriores.

Já o caso de teste “Cadastrar, Alterar, Excluir e Listar Visitas Científicas” possui valores mínimos de prioridade e severidade e o teste falhou 2 vezes durante sua execução. Então, como todas as técnicas utilizam a prioridade e severidade de uma falha como critérios, além de serem critérios analisados pela equipe, todos os algoritmos não priorizaram este caso de teste, já que não possui valor significativo para o negócio.

Vale ressaltar que houve uma considerável semelhança entre as priorizações dos algoritmos *Risk Based Testing and Metrics* e *Test Case Prioritization for Regression Testing based on Severity of Fault*, já que ambos utilizam a quantidade de erros na seleção dos testes. Já a técnica Estratégias de Seleção de Testes Baseado em Riscos houve uma certa semelhança com os testes selecionados pelo analista de testes da equipe, pois ambos utilizam apenas os critérios de prioridade e severidade de uma falha.

5.5. Considerações Finais

Este capítulo apresentou os resultados dos algoritmos desenvolvidos em um projeto de software escolhido, com o objetivo de avaliá-los a eficiência dos métodos propostos para seleção e priorização de casos de teste.

Com base na observação deste experimento, verificou-se em termos de similaridade com os testes selecionados pelo analista, que o terceiro algoritmo, apresentado na Seção 3.1.2, foi o mais similar em relação à priorização dos testes devido a utilizar apenas os critérios de prioridade e severidade. No entanto, devem-se levar em consideração os critérios analisados por cada técnica utilizada. A primeira técnica desenvolvida, apresentada na Seção 3.1.1, analisa uma maior quantidade de critérios do que a técnica então, apesar de ter obtido uma menor similaridade com os testes selecionados pelo analista, este algoritmo possui uma melhor efetividade em sua seleção dos testes.

Portanto, apesar de possuir uma menor taxa de similaridade, o primeiro algoritmo desenvolvido aborda critérios mais abrangentes que podem ser essenciais para uma escolha dos testes mais prioritários nos Testes de Regressão. Ou seja, o critério da quantidade de erros deve ser bastante considerável, pois o mesmo é um dos princípios de Teste de Software definidos por Myers (1979).

Tal princípio aborda que a probabilidade de encontrar mais erros numa seção de um programa é proporcional ao número de erros já encontrados naquela seção. Então, um caso de teste que encontrou mais erros durante uma execução, possui uma maior chance de encontrar novos ou os mesmos erros encontrados no ciclo de testes anterior.

A partir da comparação feita entre as técnicas desenvolvidas e a análise realizada pelo analista de testes, houve um distanciamento entre as priorizações determinadas pelos algoritmos. O principal motivo de tal distanciamento determinado pelas técnicas foi a utilização do critério da quantidade de erros encontrados em execuções anteriores dos testes em dois algoritmos, os quais foram relatados na Seção 5.4.

Portanto, pode-se ressaltar que é essencial na priorização de testes, utilizar informações históricas da execução dos testes. Então, utilizar informações como a quantidade de erros, importância de determinado teste para o negócio e severidade de uma falha, são critérios que devem ser utilizados em conjunto e com pesos iguais entre esses critérios, pois

todos esses critérios são determinantes para realizar de maneira efetiva a priorização dos testes em novas execuções.

6. CONCLUSÃO

Realizar testes em projetos de desenvolvimento de software é uma abordagem que garante certa redução de custo e tempo. Para isso, deve-se haver um planejamento adequado, determinando a execução de testes durante toda a fase de desenvolvimento do produto. O mau planejamento reflete também na fase dos Testes de Regressão.

Como na maioria dos casos não há planejamento adequado, torna-se inviável executar todos os casos de teste utilizados na versão anterior. No entanto, devido às restrições de tempo e custo do software, nem sempre todos os casos de teste são reexecutados em um novo ciclo.

Como solução para o seguinte problema, existem diversas técnicas de seleção de casos de teste a serem aplicados nos Testes de Regressão. Neste trabalho foram desenvolvidas algumas dessas técnicas em uma ferramenta de gerenciamento de testes, a fim de facilitar sua aplicação em projetos de desenvolvimento de software.

A partir da utilização das técnicas desenvolvidas na ferramenta, verificou-se que as técnicas mostraram uma grande efetividade na seleção dos testes. As técnicas implementadas selecionaram uma grande maioria dos testes que um analista de testes da equipe selecionaria para um novo ciclo de testes. Além de selecionar testes que não seriam selecionados pelo analista, mas que tem uma grande chance de encontrar erros em um novo ciclo de testes.

6.1. Trabalhos Futuros

Com relação a trabalhos futuros, abaixo são feitas algumas sugestões para a continuidade no estudo sobre técnicas de seleção de casos de teste para um ciclo de Regressão:

- Criação de uma nova técnica de seleção de casos de teste baseado em informações históricas de execução de testes ao utilizar outros critérios fundamentais para uma ideal seleção dos testes;
- Desenvolvimento da nova técnica em uma ferramenta amplamente utilizada no mercado;

- Validação de novas técnicas de seleção de casos de teste em vários projetos reais de desenvolvimento de software através de um estudo de caso;

REFERÊNCIAS

AMLAND, Stale. **Risk Based Testing and Matrics**. 5th International Conference EuroSTAR, 1999. Barcelona, Spain.

BACH, J. **Session-Based Test Management**, 2009. Disponível em: <http://www.satisfice.com/articles/sbtm.pdf>. Acesso em: 15/11/2011.

BASTOS, Aderson; RIOS, Emerson; CIRSTALLI, Ricardo; TRAYAHÚ, Moreira. **Base de Conhecimento de Teste de Software**. 2ª Ed. São Paulo: Martins, 2007.

BESSON, Stephane. **A Strategy for Risk-Based Testing**. 2002.

BURNSTEIN, Ilene. **Practical Software Testing**. Springer. 2002.

CHEN, Yanping, **Specification-based Regression Test Selection with Risk Analysis**, Dissertação de Mestrado, University of Ottawa, Canada, December 2002

DELAMARO, Márcio Eduardo; MALDONADO, José Carlos; JINO, Mário **Introdução ao Teste de Software**. Rio de Janeiro, 2007.

DURANTE, Felipe. Qualidade como Subproduto do Negócio. Testexpert. http://www.testexpert.com.br/?q=node/330. Último acesso: 15/11/2011.

GLOSSÁRIO PADRÃO DE TERMOS UTILIZADOS EM TESTE DE SOFTWARE. 2007. Versão 1.3br. Produzido pelo "Glossary Working Party" International Software Testing Qualification Board.

KAVITHA R.; SURESHKUMAR,Dr. N. **Test Case Prioritization for Regression Testing based on Severity of Fault**. International Journal on Computer Science and Engineering, 2010.

MAFRA, Juliana; MIRANDA, Breno; IYODA, Juliano; SAMPAIO, Augusto. **Test Case Selector: Uma Ferramenta para Seleção de Testes.** Centro de Informática. Universidade Federal de Pernambuco.

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

MYSQL. Disponível em: http://www.mysql.com/. Versão 1.2.17. 2005. Último Acesso em: 09/08/2011.

NASCIMENTO, André Luís Sena. **Técnicas de Caixa Preta de Teste de Software**. 2008.

NETBEANS. Disponível em: www.netbeans.org/. Versão 7.0. 2011. Último Acesso: 25/08/2011.

PHP. Disponível em: http://www.php.net/. 1995. Último Acesso: 02/08/2011.

PRESSMAN, Roger S. **Engenharia de Software**. McGraw Hill, 5ª Edição, 2002.

RIOS, Emerson. **Análise de Riscos em Projetos de Teste de Software**. Castelo Rio de Janeiro: Alta Books, 2008.

RIOS, E. & MOREIRA, T. **Teste de Software**. Rio de Janeiro, Alta Books, 2003.

SCHAEFER, Hans. **Risk Based Testing. Strategies for Prioritizing Tests Against deadlines**. 2005.

SOMMERVILLE, Ian. **Engenharia de Software**. 2007. 8ª Edição.

SWEBOK - Guide to the Software Engineering Body of Knowledge, 2004.

TESTLINK. Disponível em: http://www.teamst.org/. Versão: 1.9.2. 2011. Último Acesso: 01/07/2011.

VIANA, Virginia M. A. **Um Método para Seleção de Testes de Regressão para Automação**. Dissertação de Mestrado, Universidade Federal de Pernambuco, 2006.

WAMPSERVER. Disponível em: http://www.wampserver.com/en/. 2010. Último Acesso: 01/07/2011.