Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

Análise Comparativa entre Técnicas de Priorização Geral de Casos de Teste no Contexto do Teste Baseado em Especificação

João Felipe Silva Ouriques

Campina Grande, Paraíba, Brasil
Janeiro - 2012

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Análise Comparativa entre Técnicas de Priorização Geral de Casos de Teste no Contexto do Teste Baseado em Especificação

João Felipe Silva Ouriques

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande – Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linhas de Pesquisa: Engenharia de Sofware

Patrícia Duarte Lima Machado
(Orientadora)

Emanuela Gadelha Cartaxo
(Co-orientadora)

Campina Grande – Paraíba – Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

O939a Ouriques, João Felipe Silva.

Análise comparativa entre técnicas de priorização geral de casos de teste no contexto do teste baseado em especificação / João Felipe Silva Ouriques. - Campina Grande, 2012.

108f.: il.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

Orientadoras: Prof. Patrícia Duarte Lima Machado e Prof. Emanuela Gadelha Cartaxo.

Referências.

1. Teste Baseado em Modelo. 2. Priorização. 3. Experimento. I. Título.

CDU 004.415.538 (043)

"ANÁLISE COMPARATIVA ENTRE TÉCNICAS DE PRIORIZAÇÃO GERAL DE CASOS DE TESTE NO CONTEXTO DO TESTE BASEADO EM ESPECIFICAÇÃO"

JOÃO FELIPE SILVA OURIQUES

DISSERTAÇÃO APROVADA EM 05/12/2011

PATRICIA DUARTE DE LIMA MACHADO, Ph.D
Orientador(a)

EMANUELA GADELHA CARTAXO, D.Sc
Orientador(a)

FRANKLIN DE SOUZA RAMALHO, Dr.
Examinador(a)

JULIANO MANABU IYODA, Ph.D
Examinador(a)

CAMPINA GRANDE - PB

Resumo

A atividade de teste de *software* consome uma grande parte dos recursos disponíveis para a realização do projeto de desenvolvimento de um sistema. Para lidar com este alto custo, pesquisadores buscam formas de selecionar melhor quais os casos de teste que serão executados e uma delas é a priorização de casos de teste. No contexto do Teste Baseado em Modelo, as atividades de teste são conduzidas partindo de modelos comportamentais que expressam as ações possíveis no sistema, respostas e condições a serem satisfeitas, mesmo antes de o sistema ser desenvolvido. A priorização de casos de teste em estágios iniciais do processo de desenvolvimento de *software* é pouco explorada na literatura e neste trabalho ela é analisada através de uma revisão sistemática e um estudo experimental. A revisão sistemática conduzida mostrou quais técnicas estão disponíveis e como avaliar seu desempenho. Através do experimento, foi possível identificar uma ordem de desempenho entre as técnicas existentes, e verificar que podem existir evidências em comum sugeridas por outros autores em outros contextos.

Palavras-Chave. Teste Baseado em Modelo, Priorização, Experimento.

Abstract

The software test activity consumes a great part of the available resources to the realization of a development process of an application. To deal with this high costs, researchers develop ways of better selecting test cases that will be executed, and one of them is the test case prioritization. In the model-based testing context, the test activities are conducted taking the behavioral models that express possible actions, responses and conditions to be satisfied of the system, even before of your implementation. The test case prioritization in early stages of the software development process is less explored in the literature and, in this work, such context is analyzed through a sistematic review and an experimental study. The conducted sistematic review revealed which techniques is available and how to evaluate their performances. Through the experiment, it was possible identify a performance order between the considered techniques and verify that there exists evidences in common suggested by other authors in other contexts.

Keywords. Model-Based Testing, Prioritization, Experiment.

Agradecimentos

Nestas próximas linhas, gostaria de agradecer a algumas pessoas que tornaram possível a realização deste trabalho, que me inspiraram e que me ajudaram, mesmo que de maneira inconsciente, a prosseguir.

Agradecer primeiramente a **Deus**, por me dar a força para levantar todas as manhãs, por me conservar com saúde e por colocar pessoas maravilhosas no meu caminho.

A meus pais, **Valdenice e Flaviano**, e meu irmão **Paulo Victor**, por sempre apoiarem cada passo de minha jornada e por serem fortaleza e fonte de exemplos para minha vida.

À minha esposa **Raquel**, por compreender os momentos nos quais estive compenetrado demais no trabalho e por batalhar junto comigo com o mesmo objetivo.

A minhas orientadoras **Profª. Patrícia Duarte** e **Emanuela Cartaxo** por me motivarem a dar mais de mim a cada dia neste trabalho e por darem ótimas contribuições e direcionamentos para a minha vida acadêmica, mas que acabaram por influenciar também minha vida pessoal.

A **Francisco Neto**, pois compartilhamos conhecimentos envolvidos nas nossas pesquisas que aconteceram ao mesmo tempo, por meio de nossas discussões e trocas de experiências.

Aos membros de minha banca de proposta de dissertação **Prof. Franklin Ramalho** e **Profª. Raquel Lopes** por me alertar sobre alguns problemas que haviam na pesquisa em curso.

A todo o pessoal que conviveu comigo no laboratório **Everton**, **Ana Emília**, **Augusto**, **Diego** e **Adriana** por tornar os dias mais divertidos e o clima mais agradável a cada dia trabalhado. A todos os colegas do SPLab, pelo ambiente amistoso e o espírito de equipe.

Aos professores e funcionários da COPIN/UFCG por sua capacidade de fazer o curso dar certo e pela prestação do serviço.

E ainda à **CAPES** e ao **INES** pelo suporte financeiro oferecido ao longo deste trabalho.

Epígrafe

"Toda vez que eu dou um passo, o mundo sai do lugar."

Siba.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos e Códigos-Fonte

Capítulo 1

Introdução

Os processos de desenvolvimento de software destinam uma boa parte dos recursos para aferir e aumentar a qualidade do software sendo desenvolvido. Através de tarefas de verificação e validação tal objetivo pode ser atingido [Som06]. Tais tarefas podem atuar sobre diversos artefatos produzidos ao longo da realização do processo como documentos de requisitos, diagramas entidade-relacionamento e o próprio código-fonte do sistema sendo desenvolvido.

O custo do teste de *software* no orçamento é em torno de 50% [Bei90] e, por esse motivo, muitas pesquisas vem sendo desenvolvidas para lidar com os altos custos da atividade. Um exemplo disso é o Teste Baseado em Modelos - TBM. Esta abordagem se baseia no fato de que, ao expressar os requisitos do sistema a ser desenvolvido, modelos comportamentais podem ser formulados e, partindo deles, casos de teste podem ser gerados e executados automaticamente [UL07].

Além de se propor a reduzir os custos, o TBM também se presta a reduzir o tempo destinado à etapa de teste de software [UL07]. Entretanto algumas limitações podem ser percebidas, por exemplo, ele requer mais experiência do testador, quando comparado com a abordagem manual de projeto de testes, pois ele precisa estar apto a abstrair o comportamento do sistema que ainda será construído para só depois disso modelá-lo [UL07].

A redução do custo e do tempo originada pelo TBM advém da geração automática dos casos de teste a partir de modelos. Porém, ainda assim, o número de casos de teste gerados é alto devido ao fato de que os algoritmos de geração de casos de teste serem baseados em busca exaustiva, mais precisamente busca em profundidade [UL07] e quase sempre alguma restrição de tempo ou custos inviabiliza a execução completa desses casos de teste [DJK+99].

Pesquisadores investigam formas de lidar com esse problema e, como solução, pode-se selecionar uma determinada porcentagem do conjunto de casos de teste, segundo algum critério, como, por exemplo, a cobertura de transições, de requisitos, dentre outros - Seleção de Casos de Teste [CNM07]; reduzir a quantidade de casos de teste, mantendo satisfeito um conjunto de requisitos de cobertura - Redução do Conjunto de Teste [HGS93, JH03]; ou mesmo reordenar a execução desses casos de teste com a finalidade de aumentar a chance de, por exemplo, encontrar mais defeitos o quanto antes no processo de teste - Priorização de Casos de Teste [JG08, JH03]. Essas abordagens ainda podem ser combinadas de modo a afinar ainda mais o processo, por exemplo, é possível, após um processo de seleção de casos de teste, realizar uma priorização com o objetivo de fazer com que os casos de teste selecionados revelem os defeitos mais rapidamente.

Das três abordagens supracitadas, a priorização de casos de teste é a única que assegura que não há eliminação de nenhum caso de teste do conjunto gerado, o que implica que não há redução da capacidade de revelar defeitos do conjunto de casos de teste [JG08, JH03], e por este motivo ela foi o alvo da presente pesquisa.

Segundo Beizer [Bei95] existem duas estratégias de teste, a primeira é o Teste de Caixa-Branca (*White-Box Testing*), onde os casos de teste são baseados no conhecimento das estruturas internas do código-fonte do Sistema Sob Teste - SST. A segunda é o Teste de Caixa Preta (*Black-Box Testing*), em que apenas os requisitos do SST são exercitados. Neste, existem algumas formas diferentes de expressar tais requisitos [Bei95], e uma delas é através de modelos comportamentais, que são o ponto inicial do TBM e alvo da presente pesquisa.

A priorização de casos de teste aplica-se em ambos os contextos, sendo que o baseado em código-fonte é mais abordado na literatura, e sendo frequentemente associado ao Teste de Regressão[1] [KKT08, KK09, RUCH01] e por isso Rothermel et al. [RUCH01] propõem uma classificação:

- Priorização geral: ela pode ser utilizada nas etapas iniciais do processo de teste de *software*: em testes de integração, teste de sistema, teste de aceitação (como classificados, por exemplo, por Sommerville [Som06]).

- Priorização em teste de regressão: usada quando já se tem evolução do sistema a ser

[1]Modalidade de teste que avalia sistemas após modificações [EMR00].

testado e guiado por informações de alterações ou ainda de relatórios de defeitos encontrados em testes anteriores.

A maioria das técnicas de priorização tem seu funcionamento orientado por elementos e propriedades do código-fonte do SST [ERKM04, SMP08a, JZCT09, RUCH01], como por exemplo, instruções de código, chamadas a funções e procedimentos e estruturas condicionais. Poucos trabalhos focam na priorização de casos de teste no contexto do TBM [KSSM09, SM09] e ainda existem trabalhos que contém técnicas de priorização de casos de teste mistas, ou seja, direcionadas aos modelos de aplicações e orientadas pelas modificações no sistema, mas estas são condicionadas por alterações no código-fonte do sistema [KKT08, KK09].

Alguns estudos comparativos entre técnicas de priorização de casos de teste foram encontrados: para técnicas de priorização geral, são apresentados estudos somente no contexto do código-fonte do SST [JZCT09, RUCH01]; e, para técnicas de priorização de casos de teste focadas em teste de regressão foram apresentados estudos com TBM [KKT08, QNXZ07]. Todos os trabalhos aqui citados guiam as comparações das técnicas de priorização de casos de teste pela capacidade de revelar defeitos dos conjuntos de casos de teste propostos pelas técnicas participantes. Lima [dL09] propõe uma técnica de priorização geral no contexto do TBM que, com base no reuso de dados, tem o objetivo de reduzir o tempo total de execução.

Durante a realização de uma revisão sistemática, alguns trabalhos que propõem novas técnicas de priorização geral de casos de teste no TBM foram identificados, mas trabalhos que realizassem um estudo empírico envolvendo técnicas de priorização geral de casos de teste no contexto do TBM não foram identificados, de modo que pouco se sabe sobre este contexto. Desta forma, a lacuna que o presente trabalho procura preencher é **a falta de conhecimento com relação à detecção de defeitos na priorização geral de casos de teste no contexto do teste baseado em modelos.**

Dentre os resultados obtidos com este trabalho, é possível verificar o bom desempenho das técnicas baseadas no paradigma aleatório adaptativo no TBM, o que se repete no contexto do teste baseado no código-fonte, mas existem diferenças conceituais que devem ser consideradas para a sugestão de uma generalização, por exemplo:

- Os casos de teste gerados em abordagens de TBM contém redundância devido a forma

que são gerados, através de algoritmos de busca exaustiva em profundidade, o que aumenta a chance de ter muitos casos de teste revelando o mesmo defeito;

- Existe uma diferença no nível de abstração das duas abordagens, e isto impacta diretamente na relação entre os casos de teste e os defeitos. No nível do código-fonte, existe a tendência desta relação ser de um para um, pois um defeito é uma sequência específica na linha de execução do código. Já no nível dos modelos, por um trecho de um caso de teste poder ser executado em outros casos de teste, um mesmo defeito pode ser revelado por vários casos de teste;
- No nível dos modelos, os defeitos são representados de maneira abstrata e podem representar um ou vários defeitos no código-fonte da aplicação em questão.

Com essas considerações, é possível verificar que a confirmação dos resultados são apenas um indício de que os resultados podem ser generalizados, mas para isto, é necessário mais investigação levando em consideração os fatores listados.

1.1 Objetivos

O objetivo geral deste trabalho é investigar técnicas de priorização geral de casos de teste através da condução de um estudo experimental, no teste de caxa preta, no contexto do TBM, com a finalidade de indicar qual técnica das consideradas apresenta melhor desempenho com relação a sua capacidade de revelar defeitos.

Para atingir o referido objetivo geral, se faz necessário satisfazer os seguintes objetivos específicos:

- Conduzir uma revisão sistemática com a finalidade de analisar a literatura e investigar as técnicas de priorização geral de casos de teste em TBM e métricas de avaliação das mesmas;
- Definir um projeto de estudo experimental com o objetivo de avaliar as técnicas elencadas, de acordo com as formas investigadas de avaliação de técnicas;
- Executar o estudo experimental previamente definido;

- Analisar os dados adquiridos no estudo experimental e indicar a técnica, dentre as consideradas que melhor se comporta com relação às formas de avaliação consideradas.
- Relacionar com resultados obtidos por outros autores com a finalidade de tentar sugerir generalizações dos resultados.

Aqui temos dois instrumentos importantes para o sucesso do trabalho: o projeto do experimento e o conjunto de técnicas de priorização geral selecionadas como amostra. Através dos resultados do experimento e, por consequência, das hipóteses postuladas, as conclusões serão tecidas e a confiança nos resultados deste experimento depende da representatividade das técnicas selecionadas para a literatura.

1.2 Metodologia

Com relação à metodologia aplicada neste trabalho, um experimento preliminar foi feito no contexto desta mesma pesquisa [OCM10] que utilizou um projeto experimental semelhante, mas com um subconjunto diferente de técnicas e outro modelo de aplicação envolvidos.

Com o projeto do experimento preliminar analisado na etapa anterior, foi conduzido um outro estudo experimental, o instrumento principal para chegar no resultado da presente pesquisa, comparando técnicas de priorização geral de casos de teste expressivas da literatura no contexto do TBM com relação à sua capacidade de revelar defeitos, usando o projeto experimental definido anteriormente. O experimento prevê uma fase de planejamento, e dentre as suas tarefas, as variáveis a ser analisadas no estudo precisam ser definidas. As variáveis são um conjunto de técnicas de priorização de casos de teste e medidas de avaliação de conjuntos ordenados de casos de teste.

Para fazer uma escolha representativa de quais técnicas e métricas comporiam o experimento um outro estudo foi conduzido, uma revisão sistemática. Através desta, foi feita uma varredura na literatura e as técnicas e métricas sugeridas foram selecionadas para o experimento.

1.3 Contribuições do Trabalho

A reduzida quantidade de estudos experimentais na Ciência da Computação não permite consolidar a quantidade de métodos e tecnologias propostas [Fei06]. A maioria dos trabalhos que versam sobre a priorização de casos de teste se atém ao contexto do teste baseado em código-fonte, inclusive com a condução de experimentos, como já foi discutido anteriormente, e pouco conhecimento existe no que tange a priorização de casos de teste em TBM, mais precisamente a priorização geral.

Como TBM se propõe a atuar sobre modelos comportamentais de aplicações e os mesmos estão disponíveis mesmo antes do desenvolvimento do sistema em questão, é possível que não haja muitas informações que possam ser utilizadas num procedimento de priorização de casos de teste, e é aí que a priorização geral de casos de teste atua. Desta forma, o experimento conduzido durante a presente pesquisa carrega algumas outras contribuições em sua realização:

- **Realização de revisão sistemática em priorização geral de casos de teste em MBT**: para selecionar com confiança quais as técnicas e métricas seriam utilizadas na pesquisa conduzida, esta revisão foi conduzida e além dos resultados sugeridos por ela, vários artefatos foram produzidos, por exemplo: documentos explicando as fontes de pesquisa, termos (palavras-chave) consultados e todas as etapas de seleção, lista integral de todos os artigos retornados pela pesquisa e os motivos da remoção dos que foram desconsiderados. Tal material pode ser utilizado para propiciar a possível replicação do estudo por parte de outro pesquisador;

- **Implementação de técnicas de priorização em ambiente operacional**: para a realização do experimento, as técnicas comparadas foram implementadas na ferramenta LTS-BT [CANM08] (mais informações na Seção 5.3). Da maneira que as técnicas foram implementadas, além de prover o uso no âmbito do experimento, é possível utilizá-las em um processo de *software*, pois LTS-BT é uma aplicação com o objetivo de auxiliar o processo de TBM [CANM08];

- **Realização da análise assintótica dos algoritmos das técnicas implementadas**: além da implementação das técnicas, a análise do desempenho assintótico foi feita para

cada técnica com o objetivo de prover uma visão mais genérica de como o desempenho das técnicas que compuseram o experimento diminui de acordo com o crescimento da entrada considerada. Tal análise não foi feita por seus autores, o que a torna ainda mais importante;

- **Melhor conhecimento no uso da priorização geral de casos de teste em TBM**: o TBM é uma área do teste de *software* em expansão, discutido na Seção 2.2, e o uso da priorização geral de casos de teste neste contexto acrescenta contribuições no sentido de sugerir a ordenação dos casos e teste em etapas iniciais da atividade de teste, mesmo quando o *software* sendo desenvolvido ainda não está disponível. Além do exposto, a priorização geral de casos de teste no TBM é pouco explorada, dessa forma o conhecimento sobre tal temática se torna importante na investigação de aspectos genéricos da priorização de casos de teste;

- **Condução de estudo experimental em priorização geral de casos de teste em TBM**: a realização do experimento é uma contribuição pois fornece evidências fortes para avaliar as técnicas componentes com relação a capacidade de revelar defeitos. Tais evidências corroboram alguns resultados obtidos por outros autores em contextos diferentes, que é um indício de que os resutados podem ser alargados além do escopo deste trabalho.

1.4 Organização da Dissertação

Este capítulo foi um sumário das motivações para a realização deste trabalho e o dos procedimentos metodológicos adotados para atender os seus objetivos. Este trabalho está organizado da seguinte maneira:

- O **Capítulo 2** versa sobre os temas que embasam este trabalho, que são o Teste de Software, o Teste Baseado em Modelo, a Priorização de Casos de Teste e a Engenharia de Software Experimental;

- No **Capítulo 3** são apresentados os procedimentos realizados na Revisão Sistemática sobre Priorização Geral de Caso de Teste no TBM, como a definição das questões

de pesquisa, seleção de fontes de dados e etapas de seleção, e os seus respectivos resultados;

- As técnicas sugeridas pela revisão sistemática são detalhadas no **Capítulo 4**, através de seus algoritmos em pseudo-código, discussão sobre seu funcionamento e análise de seus desempenhos assintóticos;

- No **Capítulo 5**, os experimentos conduzidos são detalhados. Nele constam todas as etapas da execução do experimento, definição, planejamento, instrumentação, análise dos resultados e avaliação da sua validade;

- No **Capítulo 6** são tecidas as conclusões acerca do trabalho como um todo, relacionando com os objetivos apresentados, dificuldades encontradas no decorrer da pesquisa, os pontos que a pesquisa suscita mais investigação e os trabalhos futuros;

Capítulo 2

Aporte Teórico

Imerso no contexto apresentado, o presente capítulo expõe os fundamentos dos grandes temas que alicerçam este trabalho. O Teste de *Software* é o tema principal deste documento. Tal etapa no processo de desenvolvimento de *software* pode ser focada no código-fonte do sistema ou ainda na sua especificação [UL07]. Esta e outras classificações do teste de *software* serão mais detalhadas mais adiante na Seção 2.1.

Várias outras atividades são executadas durante o teste de *software* com a finalidade de melhorá-lo ainda mais, tais como a seleção de casos de teste [CNM07] e a redução de conjuntos de teste [JH03], e dentre elas, delimitando ainda mais o contexto, a priorização de casos de teste - PCT, como abordado na Seção 2.3 tem importância no melhor uso dos recursos alocados, de modo a satisfazer algum critério de qualidade de maneira mais rápida.

Como o trabalho versa sobre o aumento do corpo de conhecimento a respeito de técnicas de PCT através de estudos experimentais, a Engenharia de *Software* Experimental e seus métodos também serão abordados neste capítulo, especificamente na Seção 2.4.

2.1 Teste de *Software*

Durante os processos de desenvolvimento de *software*, existem tarefas que tem por objetivo aferir a qualidade do *software* sendo desenvolvido, cujas finalidades são: verificar se o mesmo satisfaz as necessidades do cliente, conceito chamado de **validação**, e avaliar se as saídas produzidas pelo sistema estão de acordo com o que está definido através de uma especificação, que é o conceito de **verificação** [Pre01, Som06]. Nos processos atuais, a mai-

oria dos custos de avaliação do sistema são empregados depois que unidades são codificadas através do teste de *software* [Som06].

Testar um *software* representa uma anomalia interessante para o engenheiro de *software*: seu objetivo natural é revelar tantos defeitos quantos forem possíveis, ou seja, "destruir" o que ele acabou de "construir" [Pre01]. Algumas definições de teste de *software* são encontradas na literatura, mas vale ressaltar duas delas:

- Teste de *software* é o processo de executar um programa com a finalidade de encontrar erros, de acordo com Myers [Mye04].
- Teste de *software* é um processo concorrente no ciclo de vida da engenharia de *software*, e tem como finalidade medir e melhorar a qualidade da aplicação que está sendo testada, segundo Craig e Jaskiel [CJ02].

A primeira definição é deveras simplória, no sentido de que afirma que o único objetivo da atividade de teste de *software* é encontrar problemas no programa sendo desenvolvido, pois é possível obter muito mais informação com a execução de casos de teste. Por isso, a segunda definição é melhor aceita por ser mais completa e alinhada aos processos de desenvolvimento de *software* atuais.

A unidade básica de toda a atividade de teste de *software* é o caso de teste [Pre01]. Em essência, as técnicas de teste de *software* tem por objetivo escolher quais os casos de teste serão executados, exercitando o sistema sob teste. Assim, um caso de teste é composto por [Jor95]:

- **Entradas**
  - **Pré-condições:** expressam as condições necessárias para o início da execução do caso de teste;
  - **Passos:** as ações que devem ser desempenhadas no sistema sob teste;
- **Saídas**
  - **Resultados esperados:** indicam quais as respostas que devem ser exibidos pelo sistema sendo testado quando são executados os passos (entrada);

– **Pós-condição:** as condições que devem ser verificadas e satisfeitas ao final da execução do caso de teste.

No início do processo de execução dos casos de teste, o testador condiciona o sistema sendo testado de modo a satisfazer todas as condições iniciais, em seguida executa cada passo e armazena os resultados obtidos e finalmente verifica se a pós-condição foi satisfeita. Após isso, os resultados obtidos são confrontados com os resultados esperados pelo caso de teste. Se todos os resultados obtidos foram iguais aos esperados, o caso de teste passou, caso contrário falhou.

A construção dos casos de teste leva em consideração o grau de conhecimento das estruturas internas do sistema no qual se pretende praticar as atividades do teste de *software* e, quanto a este fator, ele pode ser classificado da seguinte forma [CJ02, Som06, Mye04]:

- ***Black Box*** ou funcional: o teste que é executado conhecendo apenas as funcionalidades que o *software* foi projetado para fazer, portanto direcionado para os requisitos da aplicação;
- ***White Box*** ou estrutural: o teste que é feito tendo pleno conhecimento dos aspectos procedurais e caminhos lógicos do *software*.

Nenhuma das duas abordagens mencionadas acima é dita superior, pois ambas tem áreas distintas de atuação e se complementam. No caso em que o elemento a ser testado é uma unidade em baixo nível de abstração, considerando as estruturas lógicas do código-fonte, o teste estrutural é frequentemente usado e quando o elemento é um grande componente, com várias funcionalidades, ou mesmo o sistema como um todo, considerando os resultados de suas funcionalidades, com frequência a abordagem funcional é empregada [Bei95].

Estratégias de teste funcional por serem baseadas nos requisitos da aplicação a ser testada, tem a grande vantagem de que os casos de teste podem ser derivados antes mesmo de o *software* ser desenvolvido, pois são gerados a partir de documentos que especificam os requisitos do sistema, bem como seu comportamento [Bei95], artefatos estes disponíveis desde o início do processo de desenvolvimento.

Tais documentos de especificação sendo compostos usando uma notação formal, ou mesmo semi-formal, podem prover algum grau de automação no processo de teste, redu-

zindo os custos e o esforço empregados. Tal premissa é a base do tema abordado na seção seguinte, o teste baseado em modelo.

2.2 Teste Baseado em Modelo

O Teste Baseado em Modelo - TBM é uma abordagem funcional de teste que, em linhas gerais, se baseia em modelos de aplicações para conduzir o processo de teste, como por exemplo a atividade de geração de casos de teste, de avaliação dos resultados obtidos dentre outras [EFW01] e em consonância com Utting e Legeard [UL07], TBM é a automatização do teste funcional.

O processo de TBM, sumarizado na Figura 2.1, tem início com a criação do modelo abstrato da aplicação, tendo os requisitos do sistema como base para esta primeira etapa. Este modelo é dito abstrato por ser menor e mais simples que o sistema que será testado, pois este ainda não foi sequer escrito [UL07]. Na próxima etapa, os casos de teste são gerados com base no modelo escrito na etapa anterior, dando origem, como visto na Figura 2.1, a um conjunto de casos de teste.

A próxima grande etapa é a execução dos casos de teste. Eles podem ser executados manualmente, como feito no contexto deste trabalho, ou automaticamente. A execução automática é viabilizada por um procedimento de criação de roteiros de teste chamado concretização, e através de um adaptador, que é uma interface entre o roteiro e o sistema sob teste - SST, eles são executados. Logo depois que os casos de teste são executados manual ou automaticamente, os resultados são armazenados e, como última atividade do processo, estes são analisados.

Esta abordagem tem como benefícios reduzir o custo e o tempo empregado com as atividades de teste, aumentar a qualidade dos testes gerados, entre outros [UL07]. E como limitações, ainda segundo Utting e Legeard [UL07], pode-se dizer que TBM requer mais experiência do testador quando comparado com o projeto de teste manual, pois para gerar modelos de boa qualidade, o testador precisa conhecer muito bem o sistema para conseguir modelá-lo de forma adequada, além de acertar a granularidade desejada do modelo. Dessa forma, vale também ressaltar que a qualidade dos casos de testes gerados é diretamente dependente da qualidade do modelo.

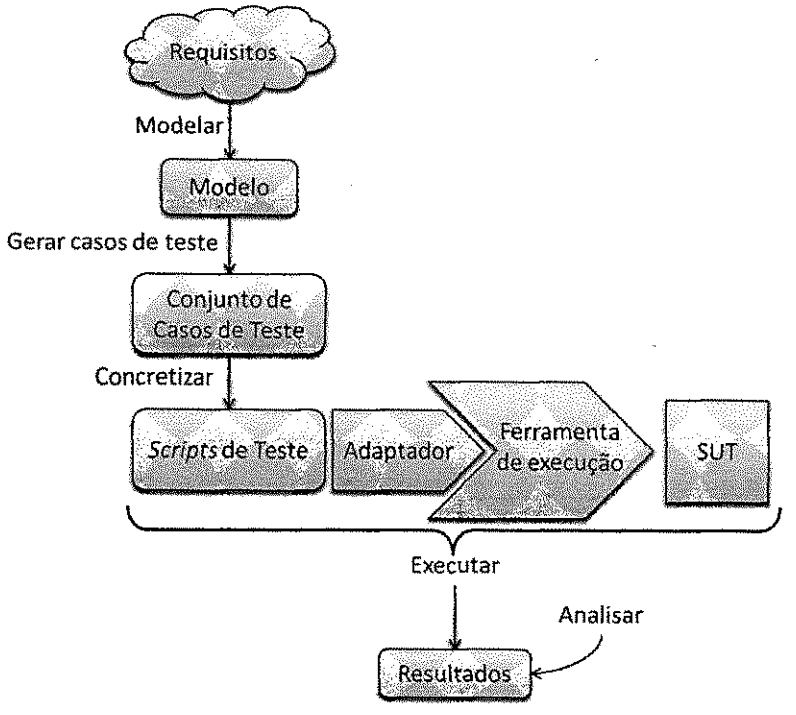


Figura 2.1: Abordagem de Teste Baseado em Modelo

É possível sintetizar os requisitos de uma aplicação em diversos tipos de linguagens, como grafos [Bei95], tomando como exemplo os grafos de fluxo de controle, grafos de fluxo de dados, árvores de chamada, grafos de máquina de estados finitos e outros, ou mesmo a linguagem *Unified Modeling Language* - UML através de alguns de seus diagramas, como por exemplo, diagrama de sequência, diagrama de máquina de estados ou mesmo diagrama de atividades [UL07]. Neste trabalho, as aplicações a serem utilizadas nos estudos experimentais foram modeladas em UML através de diagramas de atividades UML.

O diagrama de atividades é um diagrama responsável por modelar aspectos comportamentais do sistema e é composto essencialmente por [OMG04]:

- **Nó inicial**: marca o início de uma atividade;
- **Nó de ação**: passo realizado por algum ator no sistema. A responsabilidade de sua execução também pode ser definida na própria ação;
- **Nó de decisão**: simboliza um ponto que depende de uma decisão, através de uma condição ou de uma ação;

- **Nó de ramificação**: simboliza uma separação do fluxo principal em fluxos concorrentes;
- **Nó de final de fluxo**: marca o final de um fluxo específico da atividade;
- **Nó de final da atividade**: marca o final da atividade;
- **Partição de Atividade**: separa a responsabilidade da execução das ações;
- **Fluxo de controle**: elemento que interliga nós de controle (nó inicial, de ação, de decisão, de ramificação de final de fluxo e de final de atividade), marcando a continuação das ações;
- **Fluxo de objeto**: elemento que interliga nós de controle, trafegando objetos explícitos.

No contexto do TBM, dando prosseguimento ao fluxo apresentado na Figura 2.1, com a aplicação já modelada é possível derivar os casos de teste fazendo uso de algum algoritmo de busca, munido de um critério de seleção de casos de teste ou de cobertura estrutural, pois normalmente a quantidade de casos de teste é infinita [UL07].

2.2.1 Critérios de Seleção

O critério de seleção tem por finalidade delimitar a área do modelo em que o algoritmo de geração deverá atuar e gerar casos de teste. É possível enumerar alguns tipos de critérios de seleção [UL07]:

- **Cobertura estrutural**: são definidos critérios estruturais a serem satisfeitos, por exemplo: cobrir todos os estados/nós/ações, todas as transições ou ainda passar por cada *loop* (laço) no modelo apenas uma vez;
- **Cobertura de requisitos**: cobrir cada requisito com pelo menos um caso de teste;
- **Propósito de teste**: explicitar quais locais do modelo devem ser explorados [JJ02], também conhecido como definição explícita de casos de teste [UL07].

Após o processo de geração de casos de teste, um conjunto de casos de teste estará disponível para ser executado, mas o custo associado a esta execução pode exceder os recursos

alocados para a tarefa. Para tentar reduzir os custos com a execução de casos de teste, algumas abordagens tem sido propostas: seleção de casos de teste, redução de conjuntos de teste e a priorização de casos de teste. No contexto deste trabalho, sugerir uma ordem de execução para os casos de teste gerados, visando um determinado critério, é a forma adotada para utilizar melhor os recursos destinados para o processo de teste de *software* e a esta sugestão dá-se o nome de priorização de casos de teste.

2.3 Priorização de Casos de Teste

Uma vez que todos os casos de teste foram escritos ou mesmo derivados automaticamente em uma abordagem de TBM, a ordem que os mesmos foram postos inicialmente pode não ser interessante para a execução, por diversos motivos, como por exemplo consumir muito tempo entre cada um deles efetuando trocas de contextos com a finalidade de satisfazer pré-condições ou mesmo demorar muito para revelar defeitos.

A priorização de casos de teste - PCT tem por finalidade posicionar os casos de teste que satisfazem um determinado requisito de interesse mais próximo do início da ordem de execução. A Figura 2.2 exprime uma motivação, tomando-se como objetivo a detecção de defeitos mais rapidamente. As técnicas abordadas neste trabalho tem como requisito achar tais defeitos mais cedo.

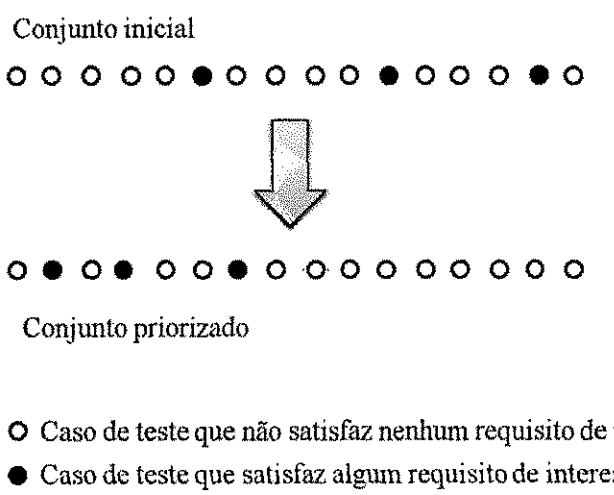


Figura 2.2: Motivação da aplicação da priorização de casos de teste

As técnicas de PCT tem por finalidade reordenar casos de teste com base em algum

critério de cobertura (i.e. transições do modelo, nós de escolha, requisitos) ou mesmo outro critério de qualidade definido, estabelecendo uma nova sequência de execução. Dessa forma, é possível definir o problema da priorização de casos de teste da seguinte forma [EMR00, JG08, JH03]:

Sejam:

- Um conjunto de casos de teste T;
- O conjunto PT de todas as permutações dos casos de teste de T;
- $f : PT \rightarrow \mathbb{R}$ uma função-custo de avaliação que mapeia uma permutação nos reais;

Encontrar um $T' \in PT$ tal que $\forall(T'' \in PT)$ com $T' \neq T''$, $f(T') \geq f(T'')$.

Esta função f tem como finalidade comparar as permutações, de acordo com a finalidade da priorização, seja custo (não necessariamente financeiro), detecção de defeitos ou qualquer outro objetivo.

A priorização de casos de testes é muito relacionada na literatura com contexto do teste de regressão [KKT08, RUCH01, dL09], por isso Rothermel et al. [RUCH01] propõem uma classificação:

- Priorização geral: ela pode ser utilizada nas etapas iniciais do processo de teste de *software* em testes de integração, teste de sistema, teste de aceitação como classificados, por exemplo, por Sommerville [Som06].
- Priorização em teste de regressão: usada quando já se tem evolução do sistema a ser testado e guiada por informações de alterações ou ainda de defeitos encontrados em testes anteriores.

Este trabalho está direcionado para o caráter geral da priorização de casos de teste, de modo que o termo **Priorização Geral de Casos de Teste** é utilizado ao longo do trabalho para denotar este contexto.

2.4 Engenharia de Software Experimental

A atividade de desenvolvimento de *software* não é fácil. Envolve esforço humano e métodos bem definidos, tanto para projetos de pequenas dimensões quanto para grandes projetos, e

ao longo dos tempos a demanda por sistemas cresce.

Com a finalidade de minimizar os problemas sofridos (a falta de qualidade do produto final, não atendimento aos requisitos do cliente, custos mal dimensionados, dentre outros) por uma grande quantidade de projetos de desenvolvimento, um conjunto de práticas foi criado e no final da década de 60, o termo Engenharia de *Software* foi cunhado para abarcar todas estas práticas como padrões e assim, dar suporte ao desenvolvimento de *software* [WRH+00]. Com o amadurecimento da engenharia de *software*, surge a necessidade de compreender seus relacionamentos internos e seus componentes [BSH86] e com isso se faz necessária a realização de pesquisas de cunho mais científico, efetuando medições, comparações e observações.

A pesquisa baseada na observação de fenômenos é chamada de pesquisa empírica. Dependendo do propósito e do grau de controle, é possível classificar a pesquisa empírica em três tipos: o *survey*, o estudo de caso e o experimento. A seguir cada um dos tipos é brevemente explanado.

O ***survey*** é muito aplicado em pesquisas nas ciências sociais [Bab90] e, no contexto da engenharia de *software*, em pesquisas que envolvem técnicas ou ferramentas que já foram utilizadas em algum momento no **passado**. Este tipo de pesquisa pode ser conduzido para descrever características de uma população, fazer asserções explicativas e ainda, ser utilizado como estudo prévio para um outro estudo melhor direcionado e mais profundo [WRH+00]. A forma mais comum de aquisição dos dados nesta modalidade de pesquisa é o formulário de questionário [WRH+00, Bab90], que pode ser respondido diretamente pelo pesquisado ou através de uma entrevista conduzida por um membro da equipe de pesquisadores.

Um outro tipo de pesquisa, o **estudo de caso**, ao contrário do *survey*, é indicado para investigar um único fenômeno **durante** o tempo que este acontece. A diferença fundamental entre o estudo de caso e o experimento (a ser tratado em seguida) é um menor controle das variáveis sob estudo, pois no estudo de caso apenas os casos em situação típica das variáveis em estudo são considerados e no experimento todas as situações das variáveis devem ser consideradas.

O **experimento** pressupõe **controle** das variáveis envolvidas e da alocação dos sujeitos às possibilidades das variáveis, os chamados tratamentos, e por isto é frequentemente realizado em laboratório. Através do experimento busca-se verificar relações de causa-efeito no mundo

real, observando variáveis que os refletem. Ele é próprio para confirmar teorias, explorar relacionamentos entre variáveis, validar modelos e medidas e ainda confirmar elementos da sabedoria popular [WRH+00].

O grande pilar da pesquisa empírica é o teste de hipóteses, realizado com a finalidade de refutar ou obter indícios de validade sobre as hipóteses formuladas com base na observação do mundo [Gau03]. Através de métodos estatísticos, as hipóteses são testadas através das variáveis associadas e, com a concepão de confiança, são refutadas ou não se consegue evidências suficientes para isto.

Tabela 2.1: Características dos tipos de pesquisa empírica [WRH+00].

	Survey	**Estudo de Caso**	**Experimento**
Controle de Execução	Não	Não	Sim
Controle nas Medições	Não	Sim	Sim
Custo da Investigação	Baixo	Médio	Alto
Facilidade de Replicação	Alta	Baixa	Alta

A Tabela 2.1 mostra uma comparação entre os três tipos de pesquisa empírica supracitados com relação à necessidade de controle na execução e na medição, custo de condução e a facilidade de sua replicação.

Na engenharia de *software*, a condução de experimentos é indicada para várias situações, dentre elas a comparação de alternativas em um processo de desenvolvimento [WRH+00, Gau03]. Para conduzí-los é necessário planejamento, controle e método. É possível sintetizar o processo através da Figura 2.3. Partindo de uma idéia, o experimento é formalizado, conduzido e conclusões são delineadas com seu resultado. Etapas de condução de um experimento são detalhadas na próxima seção.

2.5 O Processo de Experimentação

Experimentos são parte natural no processo de tomada de decisão na ciência e também na engenharia [MR03], além disso requerem controle e métodos bem definidos. Wohlin et al. [WRH+00] ilustram etapas do processo de experimentação, que pode ser sintetizado pela Figura 2.3. Nesta seção tal processo será detalhado.

2.5.1 Definição do Experimento

A definição do experimento engloba a elucidação de alguns detalhes imprescindíveis para a compreensão do experimento, são eles [WRH+00]:

- Objeto de estudo: a entidade a ser analisada no experimento. Pode ser, por exemplo, uma teoria, um produto ou um processo;
- Propósito: a finalidade de executar um experimento envolvendo o objeto de estudo. Geralmente expresso em função de um verbo no infinitivo: avaliar, medir ou mesmo comparar;
- Foco de qualidade: aspecto do objeto de estudo a ser aferido que está intimamente ligado com o propósito, como por exemplo custo, eficiência, ou uma medida inerente ao próprio objeto de estudo;
- Perspectiva: o ponto de vista que será considerado na execução do experimento, por exemplo, o ponto de vista do programador, do usuário, do gerente de requisitos ou do testador;
- Contexto: são as pessoas e os artefatos envolvidos no processo do experimento.

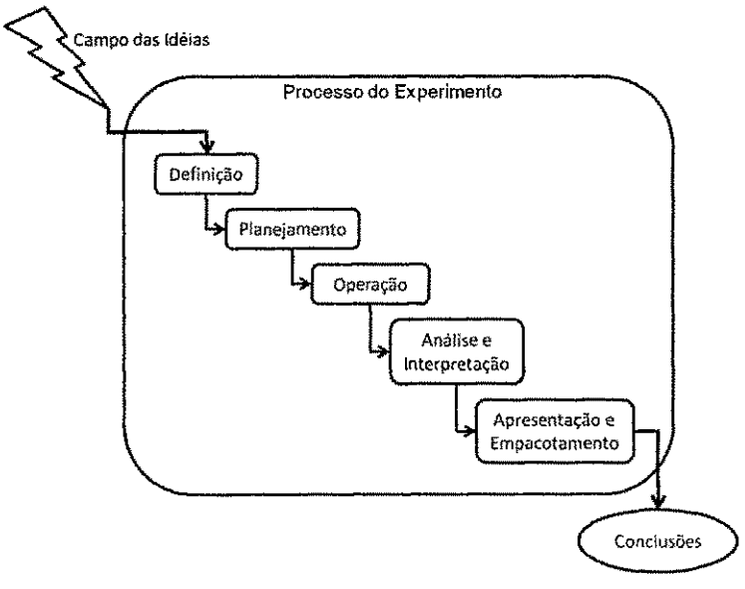


Figura 2.3: Visão geral do processo de experimentação [WRH+00].

Com tais informações devidamente claras, o experimento começa a tomar forma e pode ser detalhado.

2.5.2 Planejamento do Experimento

Nesta etapa, maiores detalhes do experimento são delineados, desde o ambiente que o mesmo será conduzido até a postulação formal da hipótese nula e da alternativa. Ainda no planejamento, a alocação de sujeitos aos objetos do experimento é definida e também é calculada a quantidade de replicações a serem feitas, o que é o chamado projeto do experimento.

Seleção de Contexto

A seleção de contexto determina em que condições o experimento será realizado. Wohlin et al. [WRH+00] enumeram quatro dimensões a serem ponderadas. São elas:

- *On-line* ou *off-line*: a primeira alternativa é um experimento que ocorre em ambiente de produção com a participação de profissionais e retrata muito bem a realidade, mas não se tem controle e é bastante caro. E a segunda alternativa, a *off-line*, é um experimento em ambiente controlado e com menos custos envolvidos;

- Estudantes ou profissionais: que tipo de sujeitos serão envolvidos no experimento? os profissionais no objeto de estudo representam melhor o contexto, mas podem não estar disponíveis para a realização do experimento. Os estudantes são mais acessíveis, mas podem não representar bem o objeto de estudo. Estudantes com alguma experiência pode ser um bom compromisso entre realismo para o contexto do experimento e acesso a sujeitos;

- Problemas simulados ou reais: as situações abordadas no experimento são simuladas ou reais;

- Específico ou geral: o presente experimento trata de um aspecto específico ou geral do objeto de estudo? Depende da capacidade de generalização dos resultados do experimento.

Seleção de Variáveis

Após a seleção do contexto, o resultado seguinte diz respeito às hipóteses, ou seja, inicialmente se enuncia uma hipótese geral norteadora do experimento e, com base na mesma e no foco de qualidade definido no planejamento do experimento, as variáveis a serem medidas no experimento são selecionadas. Tais variáveis são de dois tipos [BSH86]:

- Independente: variáveis que se pode controlar livremente em um experimento. Geralmente a seleção de variáveis independentes é feita após a seleção das variáveis dependentes. Se a variável independente será medida e relacionada diretamente com a variável dependente ela é chamada de **fator** e cada valor que ela assumir no experimento é um dito **tratamento**;
- Dependente: são as variáveis que representam o efeito dos tratamentos do experimento. Wohlin et al. [WRH+00] afirmam que frequentemente uma única variável dependente é selecionada e isto é feito com base na hipótese postulada;

Postulação de Hipóteses

No momento que as variáveis são selecionadas, as hipóteses nula e alternativa a ser testadas são formalizadas. Uma hipótese nula significa uma condição em que diferenças práticas nos tratamentos são confirmadas, mas não se tem indícios estatísticos desta diferença [WRH+00]. Desta forma se deseja refutar esta hipótese, em favor da alternativa, com o maior nível de confiança possível.

Definição de Sujeitos e Objetos

Com as hipóteses formais definidas, se faz necessário definir sujeitos e objetos do experimento. Sujeitos são as pessoas envolvidas na execução do experimento [WRH+00], por exemplo observando-as enquanto atuam em algum projeto para a coleta de dados e os objetos são artefatos, documentos que são utilizados no experimento. Também é necessário explicitar como se dará a alocação dos sujeitos aos objetos no experimento.

Projeto Experimental

O próximo passo é a definição de quantas vezes serão repetidas as execuções dos tratamentos e a organização destas execuções. Algumas práticas são importantes para projeto do experimento, a saber: aleatorização, blocagem e balanceamento.

Os métodos estatísticos utilizados na análise dos dados provenientes do experimento atuam sobre amostras independentes das variáveis consideradas [WRH+00] e para isso, é importante a inserção da **aleatorização** em todos os procedimentos, desde a alocação de sujeitos aos tratamentos até a sequência de execução dos mesmos. Além disso, dependendo do objetivo do estudo, alguns fatores podem apresentar um efeito indesejado na variável dependente, por exemplo o nível de experiência de algum grupo de sujeitos pode influenciar em uma métrica de desempenho, então coloca-se sujeitos experientes em um bloco do experimento e sujeitos sem experiência em outro bloco, o que se chama de **blocagem**, e tal atitude confere mais precisão ao resultado do experimento. E um fator importante que permeia a aleatorização e a blocagem é o **balanceamento**, por exemplo, quantidades iguais de sujeitos devem ser associadas a cada tratamento ou quantidades iguais de repetições devem ser executadas de cada tratamento.

Com todas as informações adquiridas até o presente momento, na definição e no planejamento do experimento, é possível definir o projeto do experimento. Wohlin et al. [WRH+00] apresenta quatro projetos experimentais frequentemente utilizados, diferenciados pela quantidade de fatores envolvidos e a quantidade de tratamentos considerada para cada um deles:

- Um fator com dois tratamentos;
- Um fator com mais de dois tratamentos;
- Dois fatores com dois tratamentos cada;
- Mais de dois fatores cada um com dois tratamentos.

Instrumentação

Para que um experimento seja conduzido, além dos tratamentos e sujeitos previamente selecionados, são necessários outros artefatos como documentos de especificação, roteiros de atividades para esclarecimento dos sujeitos, formulários de aquisição dos resultados, dentre

outros. Dessa forma, Wohlin et al. [WRH+00] classificam os instrumentos como sendo de três tipos: objetos, de diretrizes e de medidas.

Os **objetos** podem ser exemplificados como sendo documentos de especificação de sistemas, gráficos de andamento de processos de desenvolvimento, etc. Tais documentos são ferramentas de auxílio para a realização dos tratamentos. Já os objetos de **diretrizes** são os que descrevem procedimentos a serem realizados pelos sujeitos ao longo da operação do experimento. E os instrumentos de **medidas** são formas de obtenção dos dados referentes às variáveis dependentes que serão analisadas, por exemplo formulários de questionários ou mesmo código-fonte inserido para recolher os valores gerados pela execução do experimento.

O sucesso da realização do estudo experimental depende da correta definição e uso de tais instrumentos, pois várias ameaças à validade do experimento podem surgir como erros provocados por execução errada por parte dos sujeitos ou captação errônea dos dados, como pode ser visto na próxima seção.

Avaliação de Validade

Por fim na etapa de planejamento, a validade do experimento é avaliada através das ameaças à validade. Tais ameaças se referem a, por exemplo, a variáveis do experimento, a alocação de sujeitos, a medição ou o uso de métodos estatísticos. Wohlin et al. [WRH+00] classificam as ameaças a validade em quatro tipos: interna, externa, de construção e de conclusão, e as relacionam com os elementos do experimento de acordo com a Figura 2.4.

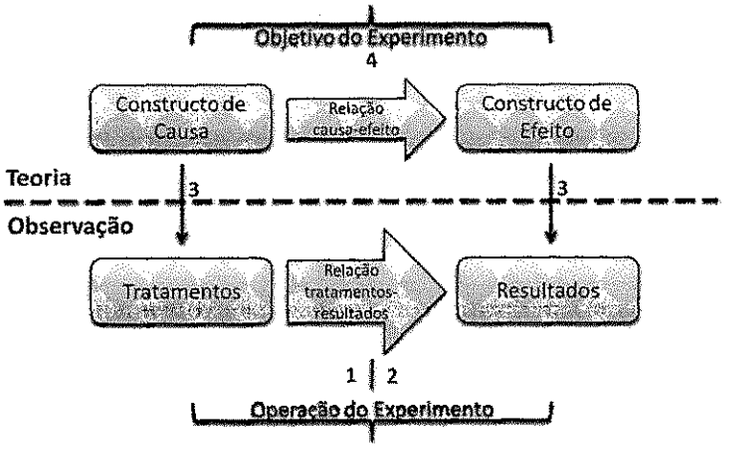


Figura 2.4: Princípios do experimento e ameaças à validade [WRH+00].

1. Validade de conclusão: refere-se ao relacionamento dos tratamentos com os resultados e pode ser ameaçada por imprecisão na análise dos dados ou mesmo por falha de suposições acerca dos dados;

2. Validade Interna: dado que existe relação entre os tratamentos e o resultado, a validade interna assegura que tal relação é causal. Ela é impactada por outros fatores que podem influenciar nesta relação ou mesmo a interação entre fatores;

3. Validade de Construção: trata da relação do mundo da teoria com o da prática, ou seja, se os tratamentos refletem bem os constructos de causa e se os resultados refletem bem os constructos de efeito. Pode ser ameaçada, por exemplo, por uma escolha errada de tratamentos ou de variável dependendente;

4. Validade Externa: versa sobre a capacidade de generalização do experimento e pode ser fragilizada, por exemplo, por uma deficiência na amostragem que represente a população de interesse.

Uma vez que o projeto do experimento foi definido e a validade do experimento avaliada, inicia-se a etapa de operação do experimento.

2.5.3 Operação do Experimento

É na etapa de operação do experimento que os dados relativos às variáveis surgem. Ela compreende três grandes atividades: a preparação, a execução e a validação dos dados.

Na preparação, todas as tarefas de preparação para a execução do experimento são desempenhadas. Os sujeitos são instruídos sobre a execução do experimento, o equipamento a ser utilizado é checado, algum *software* necessário é adquirido, instalado e testado, dentre outras atividades.

Com todos os detalhes acertados e com o projeto experimental já definido, ele é finalmente executado. Durante a execução, a coleta dos dados deve ser feita de maneira a impactar o mínimo possível na própria execução [BSH86] e estes dados devem ser organizados em arquivos de maneira a facilitar tanto a sua validação quanto a análise a ser feita sobre eles.

Ao final do processo de execução do experimento, deve ser procedida a validação dos dados obtidos. Tal validação pode consistir em verificar se os métodos de aquisição foram

de fato eficientes e, em casos que envolvem sujeitos, um seminário após a execução do experimento pode sondar se os sujeitos compreenderam bem os passos executados e por consequência, se os dados são válidos [WRH+00].

2.5.4 Análise e Interpretação dos Dados

Inicialmente é feito um procedimento de estatística descritiva com a finalidade de prover uma visualização sintética e simplificada dos dados. Alguns dispositivos estatísticos podem ser utilizados para sumarizar os dados, como por exemplo [Jai91]:

- Medidas de tendência central (média, mediana, moda): expressam através de um valor o comportamento de um grupo de dados;
- Medidas de dispersão (desvio padrão, variância): da mesma forma que as medidas de tendência central, expressam através de um valor o quão dispersos os dados estão;
- Gráfico de Dispersão: gráfico em duas dimensões que mostra os dados integralmente, representando cada ocorrência através de um ponto (exemplo na Figura 2.5).;
- Histograma: mostra os dados agrupados em categorias ou em faixas de valores (exemplo na Figura 2.6);
- Diagrama *box-plot*: congrega algumas medidas como quartis, média, mediana e alguns pontos-limite (exemplo na Figura 2.7).

Os dados coletados podem apresentar alguns valores destoantes, o que pode ser empecilho para o teste de hipótese. Tais valores devem ser investigados e caso eles não tenham influência negativa no resultado e experimento, eles devem ser removidos. O conjunto final dos dados é o artefato de entrada para os testes de hipótese.

Os testes de hipótese são as ferramentas para a tomada de conclusões com os dados do experimento. Cada teste tem suas suposições acerca dos dados que precisam ser satisfeitas para que seja possível utilizá-los com confiança. A Tabela 2.2 sintetiza alguns testes que podem ser utilizados e a sua finalidade [Jai91, Kan06, Abd07].

Todos os testes compõem um ferramental poderoso de análise de dados que, a depender das suposições de cada um dos testes e da necessidade do pesquisador, leva a conclusões precisas, sempre envolvendo um certo nível de confiança no resultado.

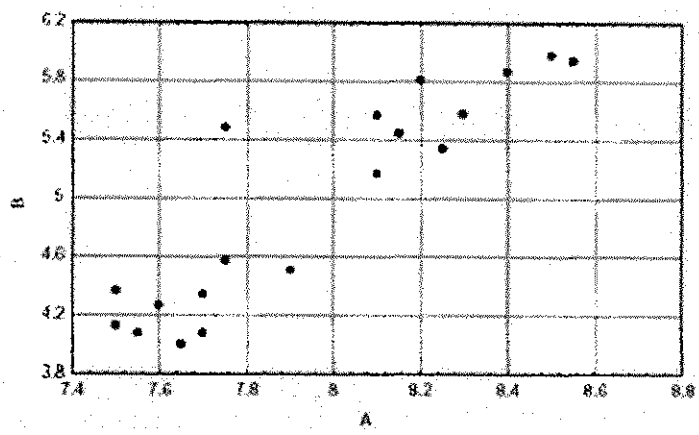


Figura 2.5: Exemplo de Gráfico de Dispersão.

Inicialmente os dados são verificados com a finalidade de comprovar se eles satisfazem as premissas para o uso dos **testes paramétricos**, que são testes realizados sobre amostras que advém populações que seguem alguma distribuição, frequentemente a distribuição normal, já os testes **não-paramétricos** não fazem tais suposições [WRH+00, Jai91]. Os testes paramétricos são testes mais poderosos e requerem amostras menores que os testes não-paramétricos, e estes por sua vez, são mais gerais quanto aos dados e por isso requerem amostras maiores.

Tabela 2.2: Sumário dos testes de hipótese.

Nome	Finalidade
Anderson-Darling	Bastante utilizado para testar se uma amostra advém de uma população que segue distribuição normal
Cochran	Utilizado para testar a igualdade de variâncias entre amostras de mesmo tamanho
Bartlett	Utilizado para testar a igualdade de variâncias entre amostras de tamanho possivelmente diferentes
Teste T	Comparação da média de duas amostras
Mann-Whitney	Alternativa não-paramétrica ao teste T
Teste T pareado	Comparação da média de duas amostras pareadas
Wilcoxon	Alternativa não-paramétrica ao teste T pareado
ANOVA	Efetuar comparações entre mais de duas alternativas
Kruskal-Wallis	Alternativa não-paramétrica a ANOVA
Tukey	Utilizado para efetuar comparações entre pares de tratamentos
Bonferroni	Alternativa não paramétrica ao teste de Tukey

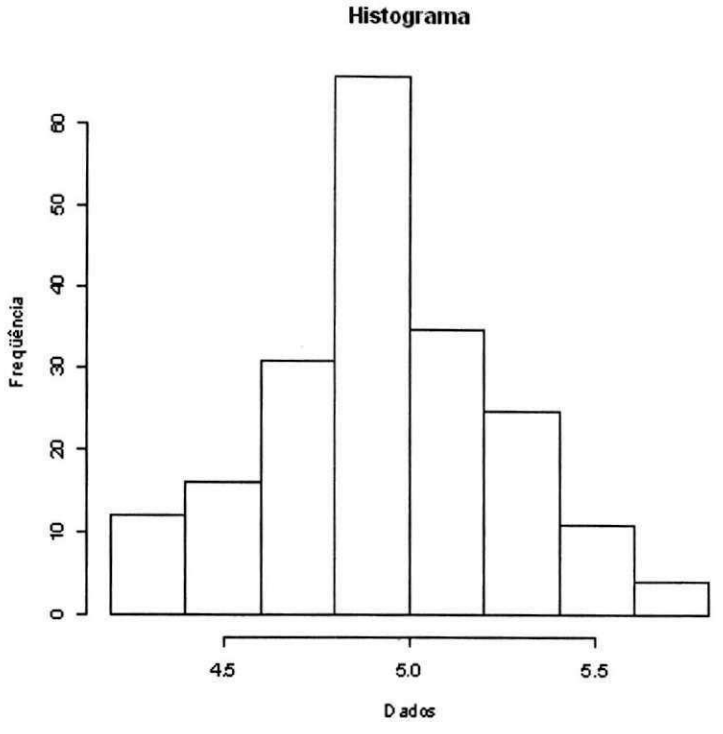


Figura 2.6: Exemplo de Histograma.

Dos testes na Tabela 2.2, os testes de Anderson-Darling, Cochran e Bartlett são utilizados para verificação de suposições nos dados, antes de ser selecionado qual o teste de hipótese será utilizado para refutar ou não a hipótese nula contra a alternativa, postuladas para o experimento. Uma vez que as suposições foram verificadas, de acordo com a quantidade de fatores e tratamentos o teste de hipótese é então escolhido.

Todos os testes de hipótese são orientados pela análise da estatística utilizada para sintetizar o seu resultado, que é o chamadp **p-valor** [Jai91]. Tal resultado é calculado pelo teste e a comparação deste valor com o nível de significância definido para o teste indica se a hipótese nula deve ou não ser refutada. Caso este p-valor seja menor que o nível de significância, a hipótese nula deve ser rejeitada face à alternativa [Kan06].

Dependendo do resultado fornecido pelo teste de hipótese, pode ser necessário que algum teste seja executado para especificar ainda mais os resultados. São os chamados testes *post-hoc*, ou pós-testes [Abd07]. Por exemplo, uma análise de variância (ANOVA) foi conduzida para comparar quatro alternativas de um fator. O teste indicou que a hipótese nula, que é a hipótese de igualdade entre as alternativas, foi refutada pela análise do p-valor, como saber

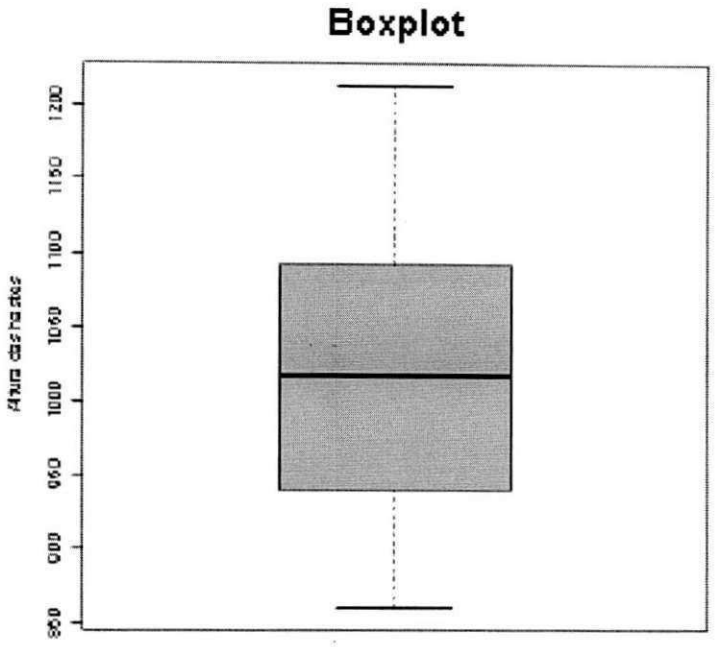


Figura 2.7: Exemplo de Gráfico de *Box-Plot*

qual delas são diferentes entre si? Para responder este tipo de pergunta é realizado um teste de múltiplas comparações, a exemplo do teste de Tukey, para a ANOVA, ou após o teste de Kruskal-Wallis, múltiplos testes de Mann-Whitney entre todos os pares de alternativas com a correção de significância de cada um dos testes, por exemplo a de Bonferroni [Abd07].

Tal procedimento de correção do nível de significância é necessário pelo fato de serem vários testes que compoem um teste total, então a significância do teste total é uma conjunção da significância dos testes individuais. Suponha que se deseje comparar três alternativas, e para isso um teste não-paramétrico rejeitou a hipótese nula de igualdade no nível de significância de 5%. Então, para fazer a comparação entre os três pares possíveis com significância também de 5%, pela correção de Bonferroni, cada teste será realizado com significância de $0.05/3 \approx 0.0167$. Então, para manter a significância geral desejada de 5%, é preciso fazer a correção das significâncias de cada teste componente [Abd07]. Com os resultados dos testes entre todos os pares de alternativas, é possível verificar as diferenças entre elas no determinado nível de significância.

Com o resultado dos testes de hipótese, as conclusões acerca do fator e seus tratamentos e a relação entre a variável dependente podem ser tomadas.

2.5.5 Apresentação e Empacotamento

Uma vez que todas as etapas da condução do experimento foram vencidas, os artefatos gerados ao longo do processo precisam ser compartilhados com a comunidade científica [WRH+00]. Todos os documentos com os resultados da etapa de definição e planejamento, arquivos com os dados obtidos e roteiros de automação da execução dos experimentos devem ser disponibilizados para que pesquisadores possam checar os resultados obtidos e, a dependender do interesse, replicar os experimentos com a finalidade de reforçar ainda mais os resultados ou refutá-los.

A publicação de artigos em periódicos ou em anais de conferências também é importante para a apresentação de resultados, pois passam pelo crivo dos revisores e qualquer detalhe que porventura tenha passado desapercebido deve ser corrigido para, assim, serem publicados. Como recurso de apresentação, ainda podem ser utilizados os relatórios técnicos.

2.6 Considerações Finais do Capítulo

Neste capítulo foram expostos os temas nos quais este trabalho está fundamentado.

A atividade de teste de *software* é o contexto deste trabalho e tem por objetivos medir e melhorar a qualidade das aplicações sendo desenvolvidas. A condução do teste no nível dos requisitos torna possível sua realização antes mesmo de que a aplicação seja desenvolvida, e de maneira automática, objetivando a redução de custos e o aumento de sua qualidade. Para a geração automática de casos de teste neste nível, é possível verificar mais critérios de cobertura e de seleção em [UL07]. Como a execução dos casos de teste pode incorrer em custos assaz elevados, procedimentos podem ser realizados para tentar lidar melhor com estes custos.

A priorização de casos de teste pode ser realizada tanto no contexto baseado no código-fonte quanto no baseado em modelos e também pode ser conduzida em conjunto a outras tarefas de redução de custos da atividade de teste de *software*, tais como seleção de casos de teste e redução de conjuntos de teste [HGS93], com o objetivo de afinar ainda mais o processo. Poucas avaliações experimentais de técnicas de teste foram realizadas e no contexto do teste baseado em modelo são ainda mais escassas.

Avaliar criteriosamente por meio de estudos experimentais é uma tarefa que requer con-

trole e rigor, necessário para a obtenção de evidências científicas acerca do fenômeno estudado. A execução de experimentos é importante para conferir o caráter científico às evidências sugeridas no contexto da engenharia de *software*. Foi exposto um esquema de condução de um experimento e o próximo capítulo, além de ter sido um grande objetivo deste trabalho em si, foi fundamental para a seleção das variáveis de pesquisa e seus tratamentos; uma revisão sistemática acerca de técnicas de priorização de casos de teste e métricas de avaliação.

Capítulo 3

Revisão Sistemática

Verificar a literatura pode ser um trabalho complicado devido à quantidade de influências que podem interferir, como autores preferidos, linhas de pensamento, grupos de pesquisa entre outros fatores. Para eliminar estas questões, é necessário seguir um método bem definido de análise de literatura. O presente capítulo detalha o procedimento de uma revisão sistemática realizada no contexto da Priorização de Casos de Teste - PCT - aplicada a conjuntos de casos de teste gerados a partir de abordagens de Teste Baseado em Modelo - TBM, e a qualquer momento durante um processo de desenvolvimento de software, ou seja, do momento em que o comportamento do sistema é especificado até quando o sistema já sofre processos de evolução durante seu uso amplo.

3.1 Revisão Sistemática

A revisão sistemática é um procedimento rigoroso e verificável de análise do Estado da Arte sobre algum tópico de interesse [Kit04]. De acordo com Kitchenham [Kit04], uma revisão sistemática é composta por algumas etapas: o planejamento, a condução e o relatório da revisão.

No **planejamento**, a necessidade que o pesquisador tem para a realização da revisão sistemática e como será o procedimento são expostos. Partindo da motivação para a realização de um processo mais sistemático de varredura da literatura, questões de pesquisa são definidas e o protocolo da revisão é desenvolvido com a finalidade de respondê-las [ESR08].

O protocolo da revisão sistemática congrega todos os passos a serem seguidos para a

realização da revisão sistemática. Ele contempla as questões de pesquisa, as fontes de informação, critérios de consulta e de seleção de fontes e também as ameaças a validade da pesquisa.

A etapa de **condução** se remete aos resultados da execução dos passos do protocolo. Nela são elencados quais trabalhos foram removidos em cada etapa da revisão e os motivos de sua remoção, os artigos reunidos e remanescentes ao final de cada etapa e os dados extraídos de cada obra. Tais dados podem ser o autor, veículo e ano de publicação, formas de avaliação da solução proposta ou mesmo a metodologia utilizada, e podem ser utilizados para, além de responder as perguntas de pesquisa, traçar perfis das pesquisas como quais os pesquisadores envolvidos, qual o recorte temporal onde a pesquisa mais se deu e os veículos de publicação mais utilizados [Kit04].

E no **relatório da revisão**, etapa final da revisão sistemática, deve-se condensar os resultados em um artigo para um periódico ou mesmo num documento de tese [Kit04]. Nas seções seguintes a revisão sistemática realizada é relatada.

3.1.1 Protocolo da Revisão Sistemática

Ao longo da presente pesquisa acerca do desempenho de técnicas de priorização geral de casos de teste baseados em modelos, um experimento preliminar foi feito com a finalidade de avaliar o projeto do experimento principal [OCM10]. Para tal estudo, algumas técnicas foram analisadas na literatura e foram comparadas à luz de métricas: **tempo de execução da técnica** e a *Average Percentage of Fault Detection* - **APFD**. A seleção das técnicas e métricas para o estudo foi feito de maneira convencional, ou seja, através de procura e leitura não sistemática de artigos de eventos e periódicos, o que torna a comparação entre técnicas de priorização geral de casos de teste menos representativa.

Para o experimento principal, relatado no Capítulo 5, se faz necessário saber quais as técnicas de priorização geral de casos de teste baseados em modelo são encontradas na literatura e quais as métricas utilizadas nas avaliações de desempenho das mesmas, para que a comparação englobe técnicas e métricas realmente representativas para a literatura. Estes fatores são os motivadores para a realização da revisão sistemática.

Uma revisão sistemática é sempre realizada com a finalidade de responder questões de pesquisa [ESR08]. Esta se propõe a responder as seguintes perguntas:

- Quais técnicas de priorização geral de casos de teste baseado em modelo foram propostas na literatura?

- Quais métricas são utilizadas na avaliação de desempenho destas técnicas?

Com a finalidade de responder as questões acima postas, mas ao mesmo tempo alargando a variedade dos resultados, meios eletrônicos foram utilizados, como sugerido por Kitchenham [Kit04]. As fontes foram os portais do IEEE Xplore[1], ACM Digital Library[2] e ScienceDirect[3]. Tais bases de dados congregam os principais veículos de publicação de trabalhos científicos relevantes na Ciência da Computação.

Os mecanismos de busca avançada de cada uma das bases de dados foram alimentados com termos de pesquisa que retornaram os resultados que serviram de matéria-prima para as etapas de seleção, que serão detalhadas mais adiante. As palavras chave utilizadas foram:

– *Test case*;

– *Test suite*;

– *Prioritization*.

Tais palavras-chave foram combinadas para formar uma expressão ligada por operadores lógicos que guiaram a seleção de fontes bibliográficas. A seguinte expressão foi usada:

test **and** (case **or** suite) **and** prioritization

A expressão poderia ser ainda mais restrita, envolvendo os termos *model-based* ou *UML-based*, por exemplo, mas incluir toda a pesquisa acerca da priorização de casos de teste, inclusive os aspectos fora do escopo desta pesquisa, foi um recurso utilizado para que não se perdesse informação relevante por inadequação ao termo da consulta.

Etapas de Seleção

Para a seleção de trabalhos, três etapas baseadas em partes diferentes foram executadas. A primeira delas foi baseada apenas no título da obra, a segunda apenas em seu resumo foi considerado e na terceira, a informação foi obtida do corpo do artigo.

[1] http://ieeexplore.ieee.org

[2] http://portal.acm.org/dl.cfm

[3] http://www.sciencedirect.com

A **primeira etapa de seleção** envolveu a leitura do título do artigo e concluir se o artigo pode ou não ser eliminado. Nesta etapa, artigos repetidos foram imediatamente removidos e os seguintes critérios foram observados:

- O título do artigo contém palavras que indiquem que ele está situado no contexto da pesquisa? Se o título se encaixa no contexto, o trabalho foi mantido;
- É possível verificar se o artigo trata do contexto baseado em código-fonte? Se sim, o trabalho foi eliminado;
- É possível verificar se o trabalho se baseia em informações de modificações, qualificando-o como para o contexto do teste de regressão? Se sim, o trabalho foi eliminado.

Para a **segunda etapa da seleção**, dois aspectos do trabalho foram analisados, o contexto da pesquisa e a sua metodologia.

- Para o contexto da pesquisa, caso a etapa anterior não tenha identificado, trabalhos que sugiram técnicas relativas ao contexto do código-fonte e dirigidos por informações de modificações (teste de regressão) foram eliminados;
- E para a metodologia da pesquisa, o trabalho precisa sugerir alguma nova técnica, preferencialmente efetuando alguma avaliação de desempenho da mesma, ou aferir eficiência de técnicas existentes na literatura através de algum estudo empírico. A repetição de técnicas em diversos trabalhos, figurando em estudos comparativos, reforça ainda mais a importância destas na literatura, pois reflete o interesse dos pesquisadores.

E, por fim, na **terceira etapa da seleção**, os trabalhos que não davam detalhes específicos de seu conteúdo ainda no resumo, e por isso não puderam ser avaliados na segunda etapa de seleção, foram avaliados utilizando os mesmos critério utilizados na etapa anterior.

Após todas as remoções feitas, respostas sobre cada trabalho fornecerão as informações necessárias para a seleção. As seguintes questões foram feitas:

- O trabalho propõe técnica(s) de priorização geral de casos de teste?

- O trabalho avalia o desempenho de uma técnica em especial? Tal estudo é rigoroso na aquisição dos dados e em sua análise?
- As métricas utilizadas são relevantes para a comparação das técnicas?

Caso o trabalho tenha resposta satisfatória em todas, o trabalho será considerado trabalho primário e terá suas informações catalogadas. Obras que sejam relatórios técnicos foram excluídos pois não passaram por revisão acadêmica pareada obrigatória e, segundo Engström, Skoglund e Runeson [ESR08], é difícil de avaliar a qualidade destes trabalhos.

Extração das Informações dos Trabalhos

De posse de todos os trabalhos selecionados, algumas informações de cada um deles foram coletadas. São elas:

- Autores;
- Ano de publicação;
- Veículo de publicação;
- Técnica(s) sugerida(s);
  - Objetivo de priorização;
  - Artefatos de entrada;
  - Elemento orientador da técnica;
  - Artefato de saída;
- Estudo empírico;
  - Técnicas envolvidas;
  - Métrica de avaliação;
  - Melhor técnica (se houver consenso).

Inicialmente, algumas informações foram obtidas para relacionar os trabalhos ao longo de tempo, entre os autores e se existe algum veículo de publicação mais representativo no contexto da pesquisa.

Em seguida, de cada trabalho foram extraídas as técnicas que o mesmo sugere e informações sobre cada uma delas serão coletadas com a finalidade de verificar se existe alguma tendência nas características das técnicas. E sobre estudos empíricos, as técnicas envolvidas e sua frequência em estudos empíricos serão catalogadas para verificar se existe alguma evidência de que alguma técnica tem melhor desempenho.

Ameaças à Validade da Revisão Sistemática

Nesta seção, as ameaças à validade da presente revisão sistemática são discutidas.

A primeira ameaça é que, por questões de disponibilidade de pessoal, apenas o próprio pesquisador efetuou a busca de trabalhos, leitura e eliminação dos mesmos ao longo das etapas de seleção. Kitchenham [Kit04] afirma que mais de uma pessoa deve realizar todo o procedimento com a finalidade de uma pessoa revisar o trabalho da outra na equipe e assim reduzir o viés na pesquisa. Para lidar com esta ameaça, cada etapa foi revisada pelo próprio pesquisador e todos os passos foram catalogados, de modo que é possível verificar todos os trabalhos e porque certos artigos foram removidos em alguma etapa de seleção.

E a outra ameaça enfrentada foi que na terceira etapa de seleção, alguns trabalhos apesar de indexados pelos sistemas de busca, não puderam ser acessados na íntegra pelo pesquisador por restrições impostas pelos portais ao domínio da Universidade Federal de Campina Grande na internet e, destes trabalhos, seus autores também não publicaram nenhum relatório técnico que pudesse ser usado como complemento. A baixa quantidade de artigos que não puderam ser acessados integralmente reduz a influência desta ameaça ao resultado da revisão sistemática.

3.1.2 Realização da Revisão Sistemática

Logo que o protocolo foi definido, os trabalhos já puderam ser pesquisados nas fontes de pesquisa sugeridas e com as palavras-chave definidas. A Tabela 3.1 sumariza a quantidade de obras listadas por cada fonte de dados.

Adiantando um resultado da primeira etapa de seleção com a finalidade de informar o leitor, 72 artigos foram repetidos nas consultas e dois deles tinham o mesmo título em mais de uma conferência, então 245 artigos foram avaliados ao todo nesta revisão sistemática.

Tabela 3.1: Quantidades de obras resultantes da busca

Fonte	Quantidade de obras
ACM Digital Library	227
IEEE Xplore	57
Science Direct	35
Total	319

A condução foi feita de acordo com o protocolo definido anteriormente, que prevê três etapas de seleção e cada uma delas será discutida nesta seção.

Primeira Etapa de Seleção

Nesta etapa, conforme o protocolo, apenas o título foi levado em consideração para remoção; e para a verificação de trabalhos repetidos, os autores e o veículo de publicação também foram observados.

Na Tabela 3.2 figuram as quantidades de artigos removidos na primeira etapa e tais remoções tiveram como principais motivos:

- **Incompatibilidade com as palavras-chave definidas**: vários resultados que não continham as palavras-chave indicadas foram retornados;
- **Repetição do artigo**: tendo em vista que três bancos de dados foram pesquisados;
- **Síntese de textos**: quando um caderno de artigos de um evento ou um periódico inteiro foi retornado.

Tabela 3.2: Resultados da primeira etapa de seleção

Motivo	Quantidade de obras
Incompatibilidade com o Contexto da Pesquisa	130
Repetição do Artigo	52
Síntese de Textos	28

É possível verificar um número elevado de trabalhos com incompatibilidade com o tema da pesquisa em andamento, de acordo com a Tabela 3.2. Isto já era esperado, pois as

palavras-chave foram escolhidas de maneira abrangente para que um conjunto maior de resultados fosse selecionado e assim, aumentar as chances de incluir mais propriamente o subconjunto de interesse desta revisão sistemática.

Segunda Etapa de Seleção

A segunda etapa de seleção foi executada analisando o conteúdo dos resumos dos trabalhos não removidos na etapa anterior.

Os principais motivos de eliminação de trabalhos nesta etapa foram:

- **Contexto diferente do da pesquisa**: quando o artigo se situa no contexto do teste dirigido ao código-fonte das aplicações ou quando as técnicas sugeridas levam em consideração elementos provenientes de alguma evolução do sistema como alterações na sua estrutura ou execuções das suítes de teste em versões anteriores;
- **Abordagem de aspecto diferente do teste de software**: quando o artigo trata de, por exemplo, seleção de casos de teste de regressão ou modelos de custo-benefício para avaliação de técnicas.

É possível verificar quantos artigos foram removidos nesta etapa por estes motivos na Tabela 3.3.

Tabela 3.3: Resultados da segunda etapa de seleção

Motivo	Quantidade de obras
Contexto Diferente do da Pesquisa	33
Abordagem de Aspecto Diferente do Teste de Software	7
Livro	1

Dois fatos necessitam ser ressaltados: um deles é a grande quantidade de artigos tratando do contexto do código-fonte e baseado em informações históricas reflete a afirmação feita por diversos autores (entre eles: Korel et al. [KKT08], Korel e Koutsogiannakis [KK09] e Rothermel et al. [RUCH01]), de que a priorização de casos de teste é tratada majoritariamente na literatura exatamente no contexto do código-fonte; e o outro, é que um livro contendo artigos de um evento ainda foi encontrado e eliminado, porque apenas pelo título a obra indicava ser promissora e por isso não foi removida na primeira etapa.

Terceira Etapa de Seleção

Com uma quantidade reduzida de artigos a serem revisados (dada a aplicação das etapas anteriores), nesta etapa o texto completo de cada artigo foi analisado. As considerações acerca do contexto do trabalho - analisadas na etapa anterior através do resumo - foram revistas com mais detalhes nesta etapa.

Algumas questões só apareceram nesta etapa, pois somente nesta era necessária a leitura na íntegra dos trabalhos. Por exemplo, o fato de se tratar de um *short paper* com duas páginas ou o texto não estar disponível, apesar de ter sido indexado em uma plataforma de busca de trabalhos com que se tem permissão de acesso.

Além desses dois fatores, o contexto ainda foi motivo de exclusão de trabalhos, uma vez que alguns dos trabalhos não removidos na etapa anterior não davam tal detalhamento no resumo. Dessa forma, trabalhos que tratam do contexto do código-fonte ou que consideravam elementos de histórico, seja alterações de versões anteriores, faltas reveladas, tempo gasto na execução de casos de teste em etapas anteriores do processo de teste, ainda foram removidos nesta etapa.

Tabela 3.4: Resultados da terceira etapa de seleção

Motivo	Quantidade de obras
Dirigidas ao Código-fonte	20
Baseadas em Histórico	9
Teste de Interação	6
Texto Completo Indisponível	4
Short Paper	2
Contexto Diferente do da Pesquisa	2

Os resultados da terceira etapa de seleção podem ser vistos na Tabela 3.4. Sobre os resultados se faz mister salientar:

- Nas categorias das obras que tratam do contexto do código-fonte e baseadas em histórico, há trabalhos que fazem parte das duas categorias mas só foram contabilizados em uma delas. Se o fator histórico tiver sido bem enfático, o trabalho foi adicionado na categoria de baseado em histórico;

- Foi perceptível a aplicação recente da priorização de casos de teste também no contexto do teste de interação[4];
- Apesar das etapas de leitura de títulos e resumos já serem vencidas, dois trabalhos tiveram assuntos diferentes da priorização de casos de teste, por tal informação não constar nos elementos analisados, portanto apenas identificável na leitura do texto na íntegra.

Ao passar por todas estas etapas de seleção, quinze trabalhos relacionados com o contexto da pesquisa não puderam ser descartados, mas como é possível ver na Tabela 3.4, o texto completo de quatro deles não puderam ser acessados e dois deles eram artigos resumo, ou seja, não traziam informações acerca da pesquisa. Então assim, nove artigos foram lidos integralmente para a extração de informações que respondem as questões de pesquisa.

3.2 Resultados Obtidos

No protocolo foi definido que informações de cada trabalho seriam utilizadas para traçar as conclusões da revisão sistemática. Dos artigos selecionados, foi possível sumarizar os dados extraidos em duas tabelas. A Tabela 3.5 sintetiza os dados referentes às métricas utilizadas em estudos comparativos, e a Tabela 3.6 congrega informações acerca das técnicas sugeridas.

Tabela 3.5: Métricas de avaliação de suites de priorização de casos de teste sugeridas pelos artigos selecionados

Métrica	Referências
$APFD$	[TAS06]
$APFD_C$	[SMP08b]
Relacionar com outras variáveis	[KSSM09, BEG09]
$APRC$	[LBC+10]
F-measure	[Zho10]
$WPFD$	[SWO05]
$TPFD$	[KM09]

[4]Modalidade de teste que prevê que os componentes a serem testados tem diferentes configurações e as diferentes combinações de tais configurações precisam ser exercitadas

Tabela 3.6: Técnicas sugeridas pelos artigos selecionados

Referência	Técnica	Entrada	El. de Priorização	Objetivo	Observações
[SMP08b]	RiteDAP	Diag. de Atividades	Prob. de faltas e Dano	Red. de Riscos	Inclui a geração de casos de teste
[KSSM09]	STOOP	Diag. de Sequência	Elementos do diagrama	Confiança no sistema	Inclui a geração de casos de teste
[TAS06]	CBR	Casos de teste, índices e conjunto de comparações	Depende dos índices	Detecção de faltas	Índices indicados para código, mas não há restrição
[Zho10]	ART	Conjunto de casos de teste	Elementos de cobertura	Detecção de faltas	O autor afirma que pode ser utilizado para diversos diagramas
[BEG09]	NN e FCM	Grafo de sequência de eventos	Elementos de cobertura	Testar partes importantes primeiro	Método de comparação não ficou claro
[SM09]	FixedWeights	Diagramas de atividades	Pesos nos nós e arestas	Detecção de faltas	Os pesos nos nós e arestas são associados automaticamente
[SWO05]	PORT1.1	Satisfat. entre requisitos e casos de teste e valores dos fatores relacionados	Prioridade, volatilidade, complexidade de implementação e inclinação a faltas	Detecção de faltas	Inspira o trabalho de [KM09]
[KM09]	ImprovedPORT	Satisfat. entre requisitos e casos de teste e valores dos fatores relacionados	Prioridade, complexidade de implementação, mudanças, impacto de faltas, completude e rastreabilidade	Detecção de faltas	-

Alguns resultados interessantes podem ser comentados sobre os artigos selecionados pela revisão sistemática. A seguir cada um deles é brevemente discutivo.

Uma consideração preliminar importante sobre as métricas é que, dois artigos, Kundu

et al. [KSSM09] e Belli et al. [BEG09], ao invés de utilizarem métricas que relacionam os casos de teste da suíte priorizada com o objetivo da Priorização de Casos de Teste, avaliam o desempenho de outras formas. O desempenho da técnica sugerida por Kundu et al. [KSSM09] é demonstrado através de uma prova de teorema, relacionando a função utilizada para conduzir a priorização com o objetivo que é a confiança no sistema, e em [BEG09] duas abordagens de agrupamento (*clustering*) são comparadas relacionando a taxa de completude do grafo de sequencia de eventos dado como entrada, com os erros quadrados médios calculados para cada abordagem.

Com relação ao restante das métricas, seis foram resultantes após o processo de revisão sistemática. É possível verificar que algumas delas são evoluções de uma delas em especial, a APFD. A seguir cada uma delas é discutida.

A métrica *Cost-Cognizant Average Percentage of Fault Detection* - ($APFD_c$) é derivada de um trabalho de Elbaum, Malishevsky e Rothermel [EMR01]. Ela foi definida com base em duas limitações da APFD, que são a suposição de que todos os casos de teste tem custos semelhantes e que todos os defeitos tem igual severidade. Dessa maneira, a métrica considera os casos de teste com custos diferentes e os defeitos com severidades diferentes e é calculada através da expressão $APFD_C = \frac{\sum_{i=1}^{m}(f_i \cdot (\sum_{j=TF_i}^{n} t_j - \frac{1}{2}t_{TF_i}))}{\sum_{i=1}^{n} t_i \cdot \sum_{i=1}^{m} f_i}$ em que:

- T é o conjunto de casos de teste;
- $t_1, t_2, \cdots, t_n$ são os custos dos n casos de teste de T;
- $f_1, f_2, \cdots, f_m$ são as severidades dos m defeitos revelados por T;
- TF_i é o primeiro caso de teste que revela o defeito i.

Como os custos dos casos de teste e a severidade dos defeitos não são contemplados pelo escopo do presente trabalho, a métrica $APFD_C$ foi descartada.

Outra métrica pesquisada foi a *Average Percentage of Requirement Coverage* - (**APRC**), utilizada na avaliação empírica feita em [LBC+10], que mede o quão rápido os requisitos são cobertos. Ela é calculada pela expressão $APRC = 1 - \frac{N_1 + N_2 + \cdots + N_m}{n \cdot m} + \frac{1}{2 \cdot n}$, em que:

- N_i é a posição do caso de teste que satisfaz o i-ésimo requisito;

- m é a quantidade de faltas que a suíte pode revelar;
- n é a quantidade de casos de teste que compõem a suíte de testes

Ela é uma generalização da métrica $APFD$ que, se o requisito considerado forem os defeitos, a expressão torna-se a mesma. Como tal generalização não é necessária, pois o requisito considerado neste trabalho são justamente os defeitos, esta métrica não foi considerada para o experimento.

Uma outra métrica sugerida pela revisão sistemática foi a *Weighted Percentage of Faults Detected* - (**WPFD**) [SWO05] e a *Total Percentage of Faults Detected* - (**TPFD**) [KM09]. A primeira delas é medida através da área do gráfico tendo no eixo das abcissas a porcentagem da suíte executada e no eixo das ordenadas a severidade acumulada das faltas reveladas. Já a segunda delas, a TPFD, muda apenas a informação do eixo das abcissas, que passa a ser a porcentagem de requisitos cobertos. Do mesmo modo que $APFD_C$, por considerar aspectos não pertencentes ao escopo, elas foram descartadas para o experimento.

Com a exclusão destas métricas, duas foram selecionadas para compor o experimento a ser detalhado no Capítulo 5, a $APFD$ e a $F-measure$. Em caráter introdutório, a $APFD$ mede o quão rápido os defeitos são revelados pelo conjunto de casos de teste e $F-measure$ representa a quantidade de casos de teste necessários pare serem executados até que seja revelada a primeira falta, de modo que quanto menos casos de teste forem executados antes da primeira falta ser revelada, melhor é a suíte de testes. Mais detalhes sobre elas, bem como expressão de cálculo de $APFD$ e de $F-measure$, são vistos no Capítulo 4.

Analisando as obras com relação às técnicas sugeridas, algumas delas ainda se mostraram inadequadas para compor o experimento por suas suposições restringirem o projeto do experimento, principalmente por ainda requererem muita informação do especialista. Dessa maneira, as técnicas sugeridas nos trabalhos citados logo em seguida não puderam compor o experimento executado.

Li et al. [LBC+10] aborda a **geração aleatória** de relações de satisfatibilidade entre requisitos e casos de teste para viabilizar a execução simulada de técnicas de PCT, sem a sugestão de uma nova técnica. Tal método diverge da proposta do TBM que prevê que os casos de teste tenham sido gerados de algum modelo comportamental [UL07].

Sapna e Mohanty [SM09] apresentam uma técnica que atua sobre diagramas de ativi-

dade UML, apenas considerando elementos de cobertura mas não executam nenhum estudo empírico para aferir a eficiência da técnica sugerida.

Stallbaum et al. [SMP08b] propõem uma abordagem de geração de cenários de teste e de priorização destes cenários baseada em especificação, tendo como base Diagramas de Atividades UML e a inserção de fatores de risco e custos nas atividades, com a finalidade de reduzir os riscos de ocorrência de defeitos que incorram em altos danos, sejam danos financeiros ou até mesmo à vida.

Esta técnica é dependente de quão precisa é a atribuição destes riscos e custos, o que poderia representar uma ameaça à validade do experimento caso fosse adicionado, uma vez que tais informações não estão disponíveis para sua execução. Como tentativa de minimizar tal fato, o autor sugere um estudo anterior no qual um método automático de atribuição de riscos é exposto [SM07], mas este método se baseia em métricas de desenvolvimento, o que conflita com as premissas do trabalho, de que o sistema pode estar indisponível ou mesmo ainda não desenvolvido. Por estes motivos, tal técnica não compôs o experimento.

Tonella et al. [TAS06] sugerem uma abordagem orientada ao raciocínio baseado em casos para priorizar casos de teste usando a capacidade de julgamento do especialista. Ela pode ser utilizada tanto nas primeiras etapas do processo de teste ou nas etapas de evolução, dependendo apenas das funções dos índices utilizados. Tais funções tem por finalidade avaliar o desempenho de um caso de teste e podem ser relacionadas com critérios de cobertura, complexidade dos casos de teste ou até mesmo elementos extraidos de execuções anteriores.

Inicialmente um subconjunto de comparações 2 a 2 sobre qual é o melhor caso de teste é formado por um especialista, em seguida o conjunto de índices é usado para criar uma função de classificação e, assumindo que a escolha do especialista é a melhor, os casos de teste são priorizados através desta função. A dependência de uma grande quantidade de comparações a ser conduzida pelo especialista é o grande motivo que fez com que esta técnica não fosse incluida no experimento.

Já em Belli et al. [BEG09] sugere uma abordagem de priorização de casos de teste baseada em agrupamento. No artigo, duas formas são comparadas, sendo uma baseada em redes neurais não supervisionadas e outra em *fuzzy c-means*. As técnicas estão mal detalhadas e envolvem muitos outros conceitos, tais como lógica *fuzzy* e redes neurais, que, se tais técnicas fossem incluidas no experimento, provocaria um aumento no tempo de implementação

das técnicas, o que seria um problema para o escopo.

E finalmente Srikanth et al. [SWO05] propõe uma técnica de priorização de casos de teste novos e de regressão orientada a requisitos que considera quatro variáveis: volatilidade, prioridades do cliente, complexidade de implementação e a inclinação à faltas dos requisitos da aplicação. Krishnamoorthi et al. [KM09] propõe uma técnica que estende a desenvolvida por Srikanth et al. [SWO05], aumentando o conjunto de variáveis e dizendo explicitamente quais variáveis se destinam a casos de teste novos e quais a casos de teste de regressão. Em ambos os trabalhos, os casos de teste são ordenados de maneira semelhante, usando uma ponderação das variáveis consideradas.

Em ambos é possível perceber que o uso de algumas variáveis como, custo de implementação e volatilidade do requisito, requerem que o sistema, pelo menos, esteja em curso de desenvolvimento, o que viola a premissa de que o sistema pode não estar disponível. E ainda variáveis como inclinação à faltas, necessitam de que o sistema já tenha sido desenvolvido e que versões do mesmo tenham sido testadas, fato este que também viola a premissa de que não existe informação de histórico de execuções. Tais motivos fizeram com que tais técnicas fossem excluídas do experimento.

Com a exclusão destas quatro técnicas, três técnicas restaram e elas integraram o experimento com um projeto semelhante ao utilizado na primeira investigação, condensada em [OCM10]. As técnicas selecionadas são detalhadas no Capítulo 4, mas a critério introdutório, Kundu et al. [KSSM09] propõem uma técnica que toma diagramas de sequencia UML como modelo de especificação de aplicações com o objetivo de aumentar a confiança no sistema. Em [Zho10] mais uma técnica que segue o paradigma aleatório adaptativo e tem por finalidade detectar faltas mais rapidamente no processo de teste. E, finalmente, em [SM09] uma técnica que processa diagramas de atividades UML que tem por objetivo revelar faltas mais prematuramente.

3.3 Considerações Finais do Capítulo

A revisão sistemática é um método rigoroso, verificável e replicável de proceder uma varredura na literatura de um certo tópico de interesse, com a finalidade de responder a questões de pesquisa. Neste capítulo foi exposto o procedimento completo da condução de uma destas

revisões, realizada com a finalidade de sumarizar as técnicas de PCT baseado em modelos em um contexto geral, ou seja, nenhuma informação além dos próprios modelos do sistema precisa estar disponível, o que qualifica-as para serem usadas a qualquer momento do processo de desenvolvimento de *software*, até mesmo antes deste ser desenvolvido.

Desejava-se analisar a literatura com a finalidade de obter um conjunto válido de técnicas de priorização geral de casos de teste no contexto do teste baseado em modelo e de métricas de avaliação para estas técnicas. Após a sua execução, três técnicas e seis métricas foram as respostas às questões de pesquisa.

No próximo capítulo, todas as técnicas e métricas reveladas pela revisão sistemática e que figuraram no estudo experimental serão detalhadas através de explicações em alto nível sobre seu funcionamento, algoritmos em pseudocódigo e expressões de cálculo.

Capítulo 4

Técnicas e Métricas Selecionadas

De acordo com a revisão sistemática relatada no Capítulo 3, três técnicas, sendo uma com duas variações, e duas métricas foram selecionadas para compor o experimento principal relatado neste trabalho. Neste capítulo, tais técnicas e métricas são analisadas em detalhes como seus princípios básicos, expressões, algoritmo em pseudo-código e análise de desempenho assintótico do algoritmo.

4.1 Técnicas Selecionadas

As técnicas aqui relatadas, por terem sido selecionadas através de um procedimento sistemático, representa significativamente a literatura no contexto do trabalho. Nesta Seção detalhes de sua construção e funcionamento são fornecidos e discutidos.

4.1.1 Fixed Weights

O princípio básico desta técnica é atribuir automaticamente pesos aos elementos estruturais de uma árvore e com base nesses pesos, os caminhos desta árvore, que representam os casos de teste, são priorizados. Esta técnica foi definida por Sapna e Mohanty [SM09] e tem seu funcionamento básico representado pela Figura 4.1.

A conversão do diagrama de atividades em árvore é feita executando uma busca em profundidade (DFS) no diagrama de atividades origem, modificada de forma que cada laço seja executado no máximo duas vezes. Dessa forma, cada atividade aparece no máximo duas

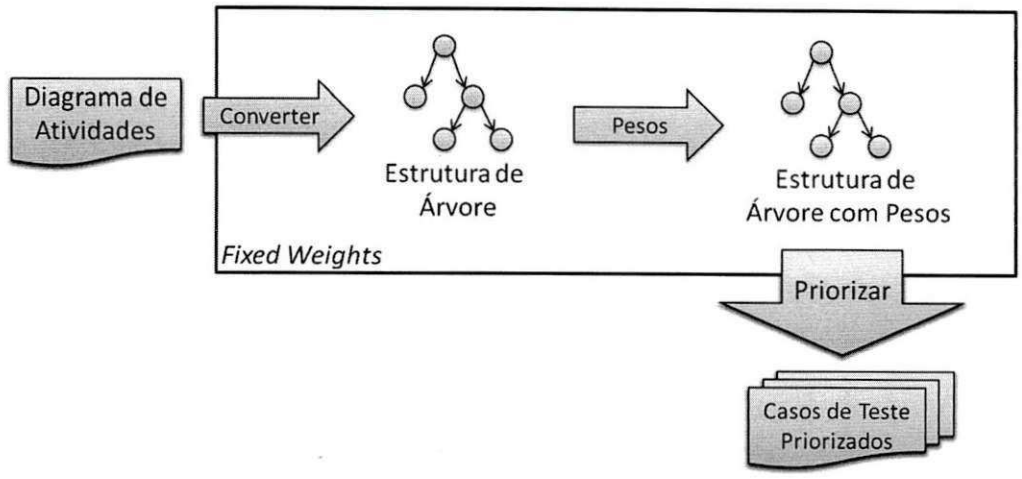


Figura 4.1: Fluxo de tarefas da técnica Fixed Weights

vezes em cada caminho da árvore. Cada nó da árvore corresponde a um nó do diagrama de atividades, com o mesmo rótulo e contendo a informação do tipo de nó correspondente (de ação, de decisão ou de ramificação) e cada aresta corresponde a um fluxo de controle ou de dados no diagrama de atividades. O Algoritmo 4.1 representa tal conversão.

Uma vez que o diagrama de atividades foi convertido em árvore, os pesos podem ser atribuídos. O peso dos nós é determinado pelo tipo de atividade correspondente no diagrama de atividades. A atribuição dos pesos dos nós $W(n)$ é feita utilizando a Equação 4.1.

$$W(n) = \begin{cases} 3, \text{ se } n \text{ é um nó de } \textbf{ramificação} \\ 2, \text{ se } n \text{ é um nó de } \textbf{decisão} \\ 1, \text{ se } n \text{ é um nó de } \textbf{ação} \end{cases} \tag{4.1}$$

Na função, o nó de ramificação é o que representa a divisão paralela do fluxo de execução e é simbolizado em UML por uma barra. Já o nó de decisão representa uma decisão que muda o fluxo de acordo com a satisfação de uma guarda e é simbolizado em UML através de um losango. Por fim, um nó de ação representa uma ação atômica executada por um objeto no sistema e é simbolizado por um retângulo com vértices arredondados.

Algoritmo 4.1 Conversão de diagrama de atividades em uma árvore

Entrada: A atividade atual A_c, a lista de atividades no caminho atual A_l, a árvore a ser gerada T e o nó atual da árvore N_c

Percorre(A_c, A_l, T, N_c)
if A_c é um Estado Inicial **then**
3: $copiaCaminho \leftarrow$ copia(A_l)
$copiaCaminho$.insere(A_c)
Percorre(Atividade filha de A_c, $copiaCaminho$, T, N_c)
6: **end if**
if T está vazia **then**
$N_c \leftarrow$ NoArvore($A_c.rotulo$, $A_c.tipo$)
9: T.adiciona(N_c)
end if
for all Fluxo F com origem em A_c **do**
12: $proxAtividade \leftarrow F.ativDestino$
if $proxAtividade$ aparece no máximo duas vezes em A_l **and** $proxAtividade$ não é um Estado Inicial **and** $proxAtividade$ não é um Estado Final **then**
$novoNo \leftarrow$ NoArvore($F.rotulo$, $F.tipo$)
15: $novaAresta \leftarrow$ ArestaArvore(N_c, $novoNo$, $F.guarda$)
T.adiciona($novoNo$)
T.adiciona($novaAresta$)
18: $copiaCaminho \leftarrow$ copia(A_l)
$copiaCaminho$.insere($proxAtividade$)
Percorre($proxAtividade$, $copiaCaminho$, T, $novoNo$)
21: **end if**
end for

Para as arestas, considere $e = [n_i, n_j]$ uma aresta qualquer da árvore, o peso $W(e)$ é calculado pela expressão $W(e) = (n_i)_{in} \cdot (n_j)_{out}$, em que $(n_i)_{in}$ é a quantidade de arestas que chegam em n_i e $(n_j)_{out}$ a quantidade de arestas que saem de n_j.

Finalmente, o peso do caminho é calculado somando os pesos dos nós e das arestas. Desta árvore resultante, cada caminho do nó raiz até as folhas representa um caso de teste. Este peso é utilizado para efetuar a priorização deles, bastando para tanto pô-los em ordem crescente de peso.

Tal atribuição dos pesos fixos para os diferentes tipos de nós foi definida por Sapna e Mohanty [SM09] com a finalidade de preferenciar os casos de teste que representam o fluxo principal da aplicação modelada. Isto é feito pela atribuição de pesos, pois casos de teste que representam fluxos principais são curtos, com poucos nós de decisão e de ramificação e ainda sem laços, o que os deixa com pesos menores e por consequência no começo da sequência. Com tais pesos calculados, a técnica organiza os casos de teste em ordem crescente de peso.

Análise Assintótica

As etapas de execução do algoritmo da técnica, de acordo com a Figura 4.1, são: conversão de diagrama de atividades em árvore, atribuição de pesos e priorização. Analisando cada uma das etapas, temos:

- Na conversão do diagrama de atividades em árvore, acontece um caminhamento usando a busca em profundidade que, de acordo com Cormen et al. [CLRS01], é feito em tempo $\Theta(A + F)$, com A sendo o número de atividades e F o número de fluxos, de dados ou de controle, do diagrama de atividades;

- A atribuição de pesos é feita em tempo linear. Para os nós, a atribuição é feita em tempo $\Theta(v)$, com v a quantidade de nós da árvore. Nas arestas, em tempo $\Theta(e)$, com e a quantidade de arestas da árvore. E para os casos de teste, assumindo mesma quantidade de nós e de arestas em cada caso de teste para o pior caso, o tempo é $O(n \cdot (v + e))$;

- E por fim a priorização é feita utilizando um *insertion sort*, que segundo Cormen [CLRS01] é feita em tempo $\Theta(n^2)$.

4.1.2 STOOP

Esta técnica tem como idéia geral a análise do aspecto estrutural do modelo da aplicação e dele calcular a métrica de priorização com a finalidade de revelar defeitos mais rapidamente, e assim aumentar a confiança no *software*. Definida em [KSSM09], ela toma como entrada um conjunto de diagramas de sequência que modelam o comportamento do sistema, e o restante das tarefas pode ser sumarizado pela Figura 4.2.

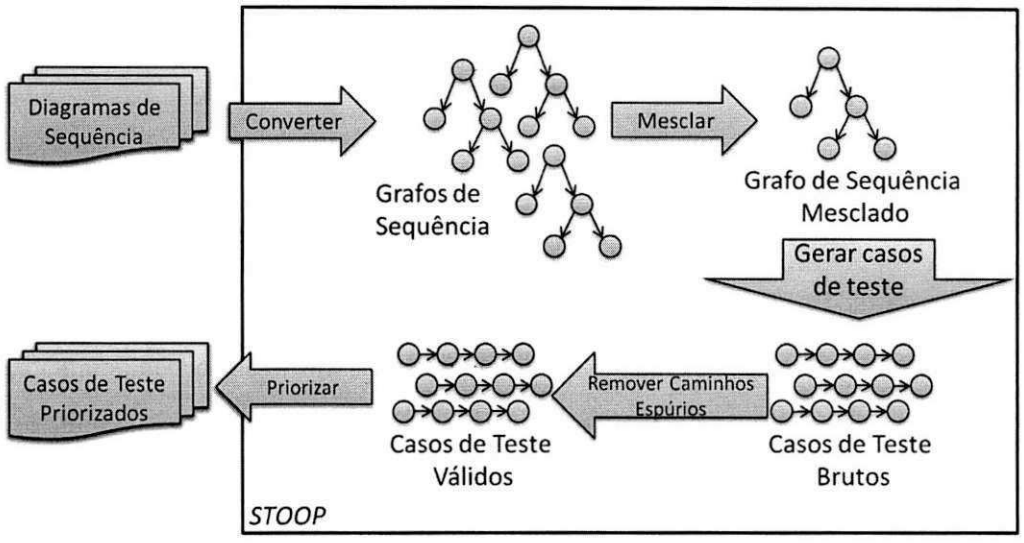


Figura 4.2: Fluxo de tarefas da técnica STOOP

O autor define os objetos do diagrama de sequência como sendo de **fronteira** aqueles com os quais o usuário pode interagir diretamente. Com essa classificação, apenas as mensagens que envolvem o usuário e objetos de fronteira são importantes para a técnica [KSSM09]. A primeira etapa dela, a conversão de diagrama de sequência para grafo de sequência é feita seguindo algumas convenções:

- $M = \{m_1, m_2, \cdots, m_n\}$ é o conjunto de mensagens recebidas por objetos de fronteira;
- $M_1' = \{m_i | m_i \in M$ e m_i é enviada por qualquer objeto que não o usuário$\}$;
- $M_2' = \{m_j | m_j \in M$ e m_j é enviada pelo usuário$\}$.

Então:

- Para cada mensagem de $m_1 \in M_1'$, um nó é criado no grafo de sequência com o texto da mensagem como rótulo;
- Para cada mensagem $m_2 \in M_2'$ uma aresta é criada no grafo de sequência entre os nós que representam as mensagens anteriores e posteriores a m_2 com o texto da mensagem como rótulo;
- Caso existam duas mensagens $m_3, m_4 \in M_1'$ que não tenha nenhuma mensagem do usuário entre elas, uma aresta sem rótulo é criada entre os nós que as representam no grafo de sequência.

Tal conversão pode ser representada em pseudocódigo pelo Algoritmo 4.2. Após cada diagrama de sequência recebido como entrada ser convertido em um grafo de sequência, eles são mesclados, dando origem a um único grafo de sequência. O procedimente de mesclagem é iniciado considerando o primeiro grafo de sequência como sendo o já mesclado e o restante deles vão sendo um a um mesclados com o primeiro. A mescla de dois grafos de sequência, SG_1 e SG_2, é definida da seguinte forma:

- Seja E_c, o conjunto de arestas comuns a SG_1 e SG_2;
- As arestas de E_c são removidas do conjunto de arestas de SG_2;
- As arestas $V_i, V_j \in SG_2$ tais que V_i ou V_j pertencem ao conjunto de nós de SG_1 são transferidas para o conjunto de arestas de SG_2

No Algoritmo 4.3 é possível acompanhar o procedimento de mescla dos grafos de sequência. Após isso é o momento de percorrer o grafo mesclado com a finalidade de gerar os casos de teste. O autor utiliza o conceito de **caminho básico** que é um caminho que parte do nó inicial até o nó final, em que cada nó só aparece no máximo duas vezes. Então, um laço só é executado uma única vez. Uma busca em profundidade munida deste critério de continuidade é realizada e os caminhos básicos resultantes são os casos de teste gerados.

Algoritmo 4.2 Conversão de Diagrama de Sequência para Grafo de Sequência
Entrada: Um diagrama de sequência SD
Saída: Um grafo de sequência SG

$SG \leftarrow$ novoGrafoSequencia
$mensagensRecebidasFronteira$ = { }
for all Mensagem M de SD **do**
  if $M.destino$ = sistema **and** $M.origem$!= usuário **then**
    $mensagensRecebidasFronteira$.insere($M$)
  end if
end for
for i = 0 ; $i <$ $mensRecebidasFronteira$.tamanho - 1 ; $i++$ **do**
  $mensOrigem \leftarrow mensagensRecebidasFronteira[i]$
  $mensDestino \leftarrow mensagensRecebidasFronteira[i+1]$
  $noOrigem \leftarrow SD$.getNo($mensOrigem$.mensagem)
  $noDestino \leftarrow SD$.getNo($mensDestino$.mensagem)
  $mensDeUsuario \leftarrow SD$.mensEntre($mensOrigem$, $mensDestino$)
  $aresta \leftarrow$ NovaAresta($noOrigem$, $noDestino$, $mensDeUsuario$.mensagem)
  SG.insereAresta($aresta$)
end for
return SG

Algoritmo 4.3 Mesclagem dos grafos de sequência
Entrada: A lista dos grafos de sequência a serem mesclados SGl
Saída: O grafo de sequência resultante SG

```
    SG ← SGl[0]
    for i = 1; i < SGl.tamanho; i++ do
3:      C ← arestasComuns(SG, SGl[i])
        arestas ← SGl[i] − C
        while arestas > 0 do
6:          a ← getArestaComNoDePara(arestas, SG.nos)
            noOrigem ← SG.getNo(a.getOrigem().rotulo)
            noDestino ← SG.getNo(a.getDestino().rotulo)
9:          arestaAd ← NovaAresta(noOrigem, noDestino, a.rotulo)
            SG.insereAresta(arestaAd)
            arestas.remove(arestaAd)
12:     end while
    end for
    return SG
```

Não é difícil perceber que, com a mescla dos grafos de sequência, podem aparecer caminhos que não estavam especificados pelos grafos de sequência originais, então após a geração de casos de teste ocorre a remoção de caminhos espúrios, cujo procedimento é detalhado pelo Algoritmo 4.4. Ela consiste em:

- Determinar os nós v_c do grafo mesclado com mais de uma aresta chegando e saindo deles (linha 1);
- Para cada caso de teste que passe por v_c, verificar se as arestas anterior e posterior são sequenciais em pelo menos um grafo de sequência original, ou seja antes de ser mesclado (linha 5));
- Caso algum caso de teste contenha v_c e as arestas anterior e posterior não sejam sequenciais em nenhum dos grafos de sequência originais, este caso de teste contém um caminho espúrio e é removido do conjunto de casos de teste (linha 6).

Algoritmo 4.4 Remoção dos caminhos espúrios do conjunto de casos de teste

Entrada: O conjunto de casos de teste gerados TC, o grafo de sequência mesclado SG e os grafos de sequência originais SGl

$nosMult \leftarrow SG$.getNosMultiplos()
for all No n de $nosMult$ **do**
3: $paresInvalidos \leftarrow$ getParesInvalidos(n, SGl)
for all Caso de teste t de TC **do**
if caminhoInvalido(t, n, $paresInvalidos$) **then**
6: TC.remove(t)
end if
end for
9: **end for**

Por fim o conjunto de casos de teste é priorizado pondo os seus elementos em ordem descrescente com relação a métrica AWPL, que é definida pela expressão $AWPL(p_k) = \frac{\sum_{i=1}^{m} eWeight(e_i)}{m}$, tomando o caso de teste $p_k = \{e_1, e_2, \cdots, e_m\}$ e $eWeight(e_i)$ sendo a quantidade de casos de teste que contém a aresta e_i.

Análise Assintótica

Da mesma forma que foi feito para a técnica anterior, a análise considerada nesta Seção pode ser feita em etapas que, de acordo com a Figura 4.2, são a conversão de diagramas de sequência para grafos de sequência, mesclagem de grafos de sequência, geração de casos de teste, remoção de caminhos espúrios e priorização. Detalhando tem-se:

- Para a conversão de diagramas de sequência em grafos de sequência, os n diagramas de sequência utilizados para especificar o sistema são convertidos. Cada conversão percorre as mensagens do diagrama de sequência, de modo que o desempenho é $\Theta(M)$, com M sendo a quantidade de mensagens do diagrama de sequência convertido. E no pior caso, assumindo que todos os diagramas contém o mesmo tamanho, a conversão de todos os diagramas de sequência em grafos de sequência e feita em tempo $O(n \cdot M)$;

- Já na mesclagem dos grafos de sequência, a execução do procedimento depende da quantidade de grafos a serem mesclados e da quantidade de arestas de cada grafo que partem ou chegam em um nó do grafo resultado. O melhor caso da situação é quando se tem apenas um grafo a ser convertido, o que resulta em um tempo $\Omega(1)$ (constante) e no pior caso, em que uma quantidade g de grafos são mesclados e de cada um deles todas as arestas, precisam ser incorporadas (assumindo um número igual de arestas e, o tempo de execução é $O(g \cdot e)$;

- A geração dos casos de teste é feita através de uma busca em profundidade no grafo de sequência resultante da mesclagem dos grafos de sequência obtidos da conversão de cada diagrama de sequência, e de acordo com Cormen [CLRS01] tal geração é realizada em tempo $\Theta(v + e)$, com v sendo o número de nós do grafo e e o número de arestas;

- A remoção de caminhos espúrios depende da quantidade de nós no grafo de sequência mesclado que tenham mais de uma aresta de entrada e mais de uma de saída e da quantidade de casos de teste gerados. Supondo um quantidade c de nós com mais de uma entrada e saída e uma quantidade n de casos de teste gerados, a remoção de caminhos espúrios é feita em tempo $\Theta(c \cdot n)$;

- E por fim, para a realização da priorização o algoritmo de ordenação *insertion sort* que segundo Cormen [CLRS01] é executado em tempo $\Theta(n^2)$, com n sendo a quantidade de casos de teste restante após o processo de remoção de caminhos espúrios.

4.1.3 Técnica Aleatório-Adaptativa

O conceito de aleatório-adaptativo foi concebido para aumentar a eficiência quando comparada com o método totalmente aleatório. Quando se recorre a solução puramente aleatória, se deseja amostrar o espaço de soluções de maneira rápida e sem viés, mas, segundo [CKMT10], apoiado em resultados de pesquisas que afirmam existir um certo padrão de defeitos, o conceito aleatório-adaptativo tem como objetivo espalhar as escolhas uniformemente pelo espaço de soluções [JZCT09, CLM04].

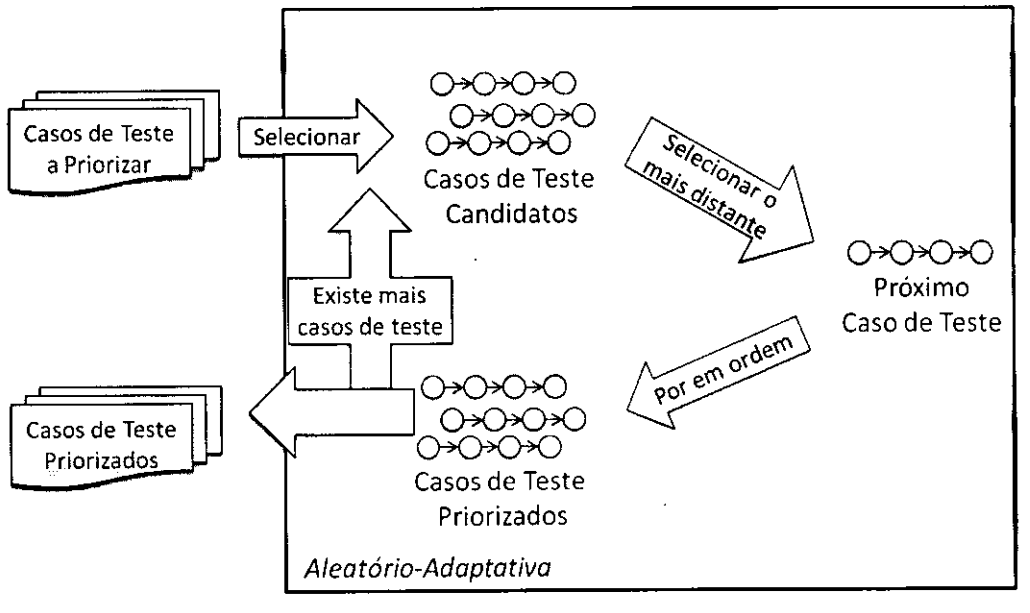


Figura 4.3: Fluxo de tarefas da técnica Aleatório-Adaptativa

A base para o seu funcionamento é a escolha de um conjunto de casos de teste candidatos

a serem selecionados e uma função de distância. Iterativamente, o caso de teste pertencente ao conjunto de candidatos mais "distante" dos já selecionados é o próximo a ser posto em ordem e isso é repetido até que o critério de parada seja satisfeito, de acordo com o fluxo ilustrado pela Figura 4.3. Essa noção de distância é genérica o suficiente para que seja definida livremente para os mais diferentes propósitos e, nesta seção, duas maneiras diferentes de cálculo de distâncias serão exploradas.

Algoritmo 4.5 Formação do conjunto de casos de teste candidatos

Entrada: O rastreador dos *branches* R e o conjunto TC dos casos de teste a serem priorizados

Saída: O conjunto C dos casos de teste candidatos

```
    aumentaCobertura ← verdadeiro
    conjCandidato ← { }
3:  branchesTotal ← { }
    repeat
      aleatorio ← TC.removeAleatorio()
6:    if aumentaCobertura(R.getBranches(aleatorio), branchesTotal) then
        branchesTotal.adiciona(R.getBranches(aleatorio))
        conjCandidato.adiciona(aleatorio)
9:    else
        aumentaCobertura ← falso
      end if
12: until TC.tamanho = 0 or !aumentaCobertura or conjCandidato.tamanho = 10
    return conjCandidato
```

A função que elege o conjunto de casos de teste candidatos pode ser representada pelo Algoritmo 4.5. Nele é possível ver na linha 5 que cada elemento a ser colocado no conjunto de casos de teste candidatos e tomado aleatoriamente e em seguida é testado se ele aumenta a cobertura de *branches* dos casos de teste já selecionados. Se aumentar, ele passa a integrar

os candidatos, caso contrário a condição de que deve aumentar a cobertura será atribuída como falsa e o algoritmo irá retornar o conjunto de casos de teste candidatos. Além desta condição, o conjunto deve conter no máximo dez casos de teste, valor obtido empiricamente por Chen et al. [CLM04] e Zhou [Zho10].

Para o cálculo da distância, uma função precisa ser selecionada de forma a medir a distância entre dois elementos integrantes da solução [JZCT09]. Duas técnicas aleatório-adaptativas foram sugeridas como resultado da revisão sistemática para compor o experimento que foi realizado, e a diferença das duas é justamente a função que avalia a distância entre os casos de teste a serem priorizados; uma delas é a função da **distância de Jaccard** e a outra é a **distância de Manhattan**.

A distância de Jaccard calcula a distância entre dois conjuntos e, no contexto do trabalho de Jiang et al. [JZCT09], o caso de teste é considerado como sendo um conjunto ordenado de arestas e então a função é utilizada. Considerando p e c dois casos de teste e $B(p)$ e $B(c)$ como sendo o conjunto de *branches* cobertos pelos casos de teste p e c respectivamente, a função da distância pode ser expressa por $J(p,c) = 1 - \frac{|B(p) \cap B(c)|}{|B(p) \cup B(c)|}$. Esta função é calculada através do Algoritmo 4.6.

Algoritmo 4.6 Cálculo da distância de Jaccard

Entrada: O rastreador dos *branches* R e os casos de teste p e c

Saída: A distância de Jaccard entre os dois casos de teste

$numBranchesComum \leftarrow |R.\text{getBranches}(p) \cap R.\text{getBranches}(c)|$

$numBranchesDiferentes \leftarrow |R.\text{getBranches}(p) \cup R.\text{getBranches}(c)|$

3: **return** $1 - \frac{numBranchesComum}{numBranchesDiferentes}$

Algoritmo 4.7 Cálculo da distância de Manhattan

Entrada: O rastreador dos *branches* R e os casos de teste p e c

Saída: A distância de Manhattan entre os dois casos de teste

```
    todosOsRequisitos ← R.getBranches()
    requisitosP ← NovoArray[todosOsRequisitos.tamanho]
 3: requisitosC ← NovoArray[todosOsRequisitos.tamanho]
    for i = 0; i < todosOsRequisitos.tamanho; i++ do
      if R.satisfaz(p, todosOsRequisitos[i]) then
 6:     requisitosP[i] ← 1
      else
        requisitosP[i] ← 0
 9:   end if
      if R.satisfaz(c, todosOsRequisitos[i]) then
        requisitosC[i] ← 1
12:   else
        requisitosC[i] ← 0
      end if
15: end for
    distancia ← 0
    for i = 0; i < todosOsRequisitos.tamanho; i++ do
18:   distancia = distancia + |requisitosP[i] − requisitosC[i]|
    end for
    return distancia
```

A segunda distância é utilizada pela técnica proposta por Zhou [Zho10] e tem por nome distância de Manhattan. Ela é calculada utilizando dois *arrays* de tamanho igual a quantidade de *branches* no modelo do sistema. Como a função avalia a distância entre dois casos de teste, para cada um deles é associado um dos *arrays* e cada posição deste contém o número 1 se o caso de teste contiver o *branch* simbolizado pela posição do *array* e 0 caso contrário. É possível verificar como esta distância é calculada através do Algoritmo 4.7.

Com relação ao cálculo das distâncias entre os conjuntos dos casos de teste já priorizados e os candidatos, ainda se deve especificar o conceito de "o mais distante". É possível maximizar a distância mínima, média ou máxima entre os pares dos casos de teste. Jiang et al. [JZCT09] conduziu um estudo empírico comparando estas três combinações entre si e concluiu que maximizar a distância mínima entre os casos de teste do conjunto já priorizado e do conjunto candidato é mais indicada para a detecção de defeitos. Então no experimento aqui relatado, as técnicas aleatório-adaptativas maximizam as distâncias mínimas e selecionam o próximo caso de teste a ser selecionado de acordo com o Algoritmo 4.8. De acordo com este algoritmo, na linha 2 é possível verificar que, quando nenhum caso de teste foi posto em ordem ainda, o primeiro é escolhido de maneira aleatória dentre os candidatos. Quando a mesma função é chamada novamente, a matriz das distâncias entre os casos de teste priorizados até o momento e os casos de teste do conjunto de casos de teste candidatos é preenchida, e na linha 7, a função distância é utilizada, sendo neste trabalho a função da distância de Jaccard ou a de Manhattan, como já visto anteriormente.

Finalmente, todos os algoritmos são utilizados pela função principal que contém o comando iterativo que seleciona o próximo caso de teste a ser posto em ordem até que todos sejam considerados, ela pode ser representada pelo Algoritmo 4.9

Algoritmo 4.8 Cálculo do próximo caso de teste a ser posto em ordem

Entrada: Os casos de teste já priorizados TS_p e os de casos de teste candidatos $Cand$

Saída: O próximo caso de teste a ser posto em ordem P_t

```
    if TS.tamanho = 0 then
        return Cand.getCasoDeTesteAleatorio()
3:  end if
    distancias ← NovaMatriz[TS_p.tamanho][Cand.tamanho]
    for i = 0; i < TS_p.tamanho do
6:      for j = 0; j < Cand.tamanho do
            distancias[i][j] ← funçãoDistância(TS_p[i], Cand[j])
        end for
9:  end for
    indice ← indiceMaximaDistanciaMinima(distancias)
    return Cand.remove(indice)
```

Algoritmo 4.9 Algoritmo geral de priorização aleatório-adaptativa

Entrada: O conjunto de casos de teste a ser priorizado TS e o rastreador de branches R

Saída: O conjunto de casos de teste priorizado P_{TS}

```
    P_TS ← { }
    while TS.tamanho > 0 do
3:      conjCandidato ← calculaCandidatos(R, TS)
        proximoCasoDeTeste ← selecionaProximo(P_TS, conjCandidato)
        TS.remove(proximoCasoDeTeste)
6:      P_TS.adiciona(proximoCasoDeTeste)
    end while
    return P_TS
```

Análise Assintótica

A abordagem aleatório-adaptativa assume que os casos de teste já estão disponíveis, então o processo de geração de casos de teste não é compreendido por ela. Dessa forma é possível efetuar a análise assintótica da geração do conjunto de casos de teste candidatos e da maximização das distâncias mínimas, que sugere o próximo caso de teste a ser posto em ordem.

- O processo de geração do conjunto de casos de teste candidatos é feito repetidas vezes e depende da quantidade de casos de teste são necessários para a formação deste conjunto. Como já visto no Algoritmo 4.5, há uma limitação do tamanho desse conjunto em dez elementos e eles devem ser adicionados enquanto a cobertura de *branches* aumentar e enquanto o conjunto de casos de teste ainda por priorizar contenha elementos. Assim, o melhor caso é quando o conjunto de casos de teste a priorizar contém apenas um elemento e apenas ele estará no conjunto de casos de teste candidatos e o pior caso é quando o conjunto candidato é definido com os dez elementos;
- A maximização das distâncias mínimas consiste na montagem de uma matriz com as distâncias entre os casos de teste candidatos e os já priorizados e depois percorrê-la analisando os valores previamente calculados, como visto no Algoritmo 4.8. Como a quantidade de casos de teste candidatos é fixa, esta etapa tem desempenho $O(n)$ em que n é a quantidade de casos de teste a ser priorizados.

4.2 Métricas Selecionadas

Da mesma forma que as técnicas, as métricas também foram reunidas através da realização da revisão sistemática. A seguir as duas métricas se encontram detalhadas.

4.2.1 *Average Percentage of Fault Detection* - APFD

Rothermel et al. [RUCH01] definiram uma métrica que tem por objetivo aferir quão rápido um conjunto de casos de teste priorizado revela defeitos, baseado nas posições que os primeiros casos de teste que revelam defeito são colocados na suite de testes. Tal métrica

é chamada APFD (*Average of the Percentage of Fault Detected*) e é expressa da seguinte forma:

$$APFD = 1 - \frac{TF_1 + TF_2 + \cdots + TF_m}{n \cdot m} + \frac{1}{2 \cdot n}$$

Em que:

- TF_i é a posição do caso de teste que revela o i-ésimo defeito;
- m é a quantidade de defeitos que podem ser revelados;
- n é a quantiadade de casos de teste que compõem o conjunto de casos de teste.

Como exemplo ilustrativo presente em [ERKM04], temos um conjunto com cinco caso de teste, A, B, C, D e E, e dez defeitos podem ser reveladas por este conjunto. A Figura 4.4 reflete a relação entre casos de teste e os defeitos que eles revelam.

Teste	Faltas reveladas pelo teste									
	1	2	3	4	5	6	7	8	9	10
A	x				x					
B	x				x	x	x			
C	x	x	x	x	x	x	x			
D					x					
E								x	x	x

Figura 4.4: Casos de teste e os defeitos que eles revelam

Agora considerando o conjunto de casos de teste não priorizado A-B-C-D-E e o conjunto priorizado C-E-B-A-D, seus valores de APFD calculados podem ser vistos, respectivamente, na Figura 4.5 e Figura 4.6.

A taxa é representada como a área sombreada vista nas Figuras 4.5 e 4.6. Como se trata de uma probabilidade, o valor de APFD x está no intervalo $0 \leq x \leq 1$, significando que quanto mais próximo de 1, mais rápido a suite consegue revelar faltas e quanto mais próximo de 1, melhor é o desempenho.

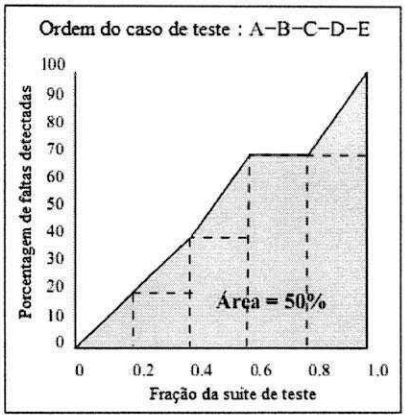


Figura 4.5: APFD do conjunto de casos de teste não priorizado

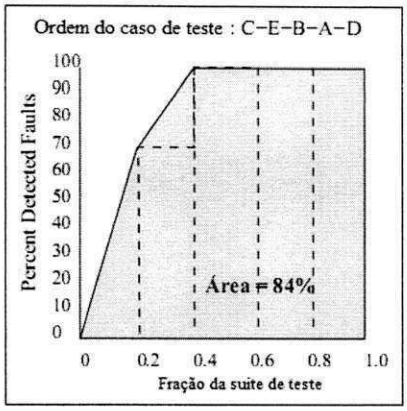


Figura 4.6: APFD do conjunto de casos de teste priorizado

4.2.2 *F-Measure*

A *F-measure* foi utilizada por Zhou [Zho10] dentre os resultados fornecidos pela revisão sistemática. Ela considera os mesmos fatores para calcular seu resultado, que são as posições dos casos de teste que revelam os defeitos.

Ela é calculada pela expressão $F = TF_1 - 1$, em que TF_1 é a posição do caso de teste que revela o primeiro defeito. Em outras palavras, *F-measure* significa a quantidade de casos de testes a serem executados até que o primeiro defeito seja revelado. Sua escala de valores vai de zero, quando o primeiro caso de teste já revela o primeiro defeito, até o tamanho do conjunto de casos de teste, quando nenhum defeito pode ser revelado por ele e quanto mais baixo o seu valor, melhor é o desempenho.

Apesar de serem bastante relacionadas, com APFD parecendo incluir o resultado sugerido por *F-measure*, não é dificil imaginar uma situação em que o mesmo conjunto de casos de teste obtenha baixo resultado com relação APFD e alto com relação a *F-measure* ou vice-versa. Na Figura 4.7, analisando as duas sequências, a primeira delas apresenta maior APFD, visto que os defeitos são revelados mais rapidamente. Já a segunda apresenta maior valor de *F-measure*, pois nela já o primeiro caso de teste revela o primeiro defeito.

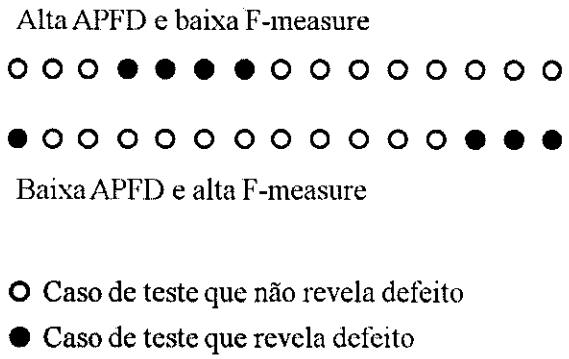


Figura 4.7: Resultados diferentes entre APFD e *F-measure*

4.3 Considerações Finais do Capítulo

Neste capítulo, **FixedWeights**, **STOOP** e **ART** com as duas funções de distância, e as duas métricas de avaliação de desempenho **APFD** e ***F-measure***, selecionadas pela revisão sistemática e que compuseram o experimento, foram detalhadas. Algoritmos em pseudo-código foram utilizados para mostrar como as mesmas foram sugeridas por seus respectivos autores e como foram implementadas para a execução do experimento. Duas delas, FixedWeights e STOOP contém seus próprios algoritmos de geração de casos de teste, mas ambos são modificações da busca em profundidade.

A análise assintótica de cada técnica foi feita de forma separada para cada procedimento com a finalidade de visualizar os pontos onde mais se consome tempo na execução das técnicas. A técnica STOOP é a que apresenta maior consumo de tempo, principalmente pela maior quantidade de etapas a serem desempenhadas, como visto na Figura 4.2.

As métricas APFD e *F-measure* analisam os casos de teste que revelam defeitos, com a finalidade de aferir o quão rápido tais defeitos são revelados pelo conjunto de testes sugerido pelas técnicas após o processo de priorização de casos de teste, durante a comparação das técnicas no experimento detalhado no próximo capítulo.

Capítulo 5

Avaliação Experimental

Observar fenômenos é uma forma bastante poderosa de geração de conhecimento, e uma ferramenta importante para este intento é a realização de experimentos. Existem trabalhos que realizam experimentos envolvendo técnicas de priorização geral de casos de teste, mas nenhum deles direcionado ao Teste Baseado em Modelos - TBM. Deste modo, a presente pesquisa se propõe a realizar um estudo experimental atacando justamente esta escassez.

Para que a observação dos dados no experimento reflitam bem a situação da literatura é necessário que as técnicas selecionadas e as métricas de análise sejam representativas. Para tanto, uma revisão sistemática foi realizada e relatada no Capítulo 3 com a finalidade de agregar resultados obtidos neste contexto, e, finalmente, sobre eles realizar observações por meio de um estudo experimental.

O presente capítulo versa sobre este estudo realizado no âmbito da presente pesquisa. Como já detalhado na Seção 2.5, a condução de um experimento prevê algumas etapas. Este capítulo está estruturado da seguinte forma: na Seção 5.1 é apresentada a definição do experimento, na Seção 5.2 todo o planejamento do experimento é feito e a instrumentação do experimento é contemplada na Seção 5.3. Os resultados obtidos dos dados brutos e dos testes de hipótese são apresentados na Seção 5.4 e na Seção 5.5, tais resultados são discutidos, relacionando com o comportamento dos tratamentos e na Seção 5.6 todas as ameaças à validade encontradas são elencadas juntamente com as medidas tomadas para reduzí-las.

5.1 Definição

Motivado pela falta de conhecimento na literatura com relação à detecção de defeitos na priorização geral de casos de teste no contexto do TBM, o experimento tem como objetivo avaliar a capacidade de revelar defeitos das técnicas existentes. Sempre que possível será feita uma comparação com os resultados já obtidos na literatura em outros contextos com a finalidade de conferir mais generalização nos resultados aqui sugeridos. Apoiado nisto, é possível definir uma hipótese geral norteadora da pesquisa:

"As técnicas de priorização geral de casos de teste apresentam diferentes capacidades de detecção de defeitos."

Wohlin et al.[WRH+00] sugere um formato para a definição do experimento (abordado no Capítulo 2) que é baseado em conceitos-chave para o experimento:

- O **objeto de estudo** são as técnicas de priorização geral de casos de teste, que foram selecionadas por meio da revisão sistemática relatada no Capítulo 3;
- O **propósito** do experimento é comparar as técnicas com relação ao foco de qualidade;
- O **foco de qualidade** é a sugestão de conjuntos de casos de teste com maior capacidade de revelar defeitos das técnicas, medidas através de métricas que serão detalhadas posteriormente;
- A **perspectiva** considerada é a do testador, ou seja, o responsável pela execução das técnicas de priorização;
- E o **contexto** que permeia a pesquisa é o do Teste Baseado em Modelos.

Sumarizando tais conceitos em uma única expressão, o experimento consiste em:

Analisar técnicas de priorização de casos de teste, **com o propósito** de compará-las com respeito a sua capacidade de sugerir conjuntos de casos de teste com maior poder de revelar defeitos, **do ponto de vista do** testador **no contexto da** priorização geral de casos de teste no TBM.

Tal expressão em conjunto com a hipótese geral previamente postulada compõem o ponto inicial para o detalhamento do experimento, também chamado de planejamento, feito na seção seguinte.

5.2 Planejamento

No planejamento do experimento os detalhes são definidos, tais como o **contexto** do experimento (não devendo ser confundido com o contexto da pesquisa como visto na seção anterior), as **variáveis** e **hipóteses formais** bem como o **projeto do experimento**.

5.2.1 Seleção de Contexto

O experimento aqui relatado foi realizado em ambiente de laboratório, por necessidade de controle e, por isso, o experimento tem caráter *off-line* [WRH+00]. Além disso, todas as técnicas envolvidas requerem apenas o modelo da aplicação para poderem ser executadas, o que permitiu a ausência de pessoal para a configuração dos artefatos para o experimento, eliminando assim a influência do fator "nível de experiência", que poderia implicar em mais ameaças à validade.

Com relação à natureza das situações, as aplicações cujos modelos comportamentais foram construídos para serem submetidos ao experimento são reais, o que torna ainda mais aceitável o contexto do experimento. Ainda que o experimento envolva aplicações reais, não se pode generalizar os resultados obtidos, uma vez que o conjunto universo de aplicações é muito extenso. Logo o experimento trata de um contexto específico.

5.2.2 Variáveis, Fatores e Tratamentos

A seleção de variáveis foi diretamente orientada pelas hipóteses, que expressamente relacionam os objetos de estudo com os aspectos dos mesmos que devem ser aferidos. Fazendo uso da definição de variáveis exposta na Seção 2.5 do Capítulo 2, as variáveis selecionadas foram:

- As **variáveis dependentes** do experimento são as medidas de detecção de defeitos **APFD** e ***F-Measure***, calculadas ao final da execução do experimento. APFD é calculada pela expressão $APFD = 1 - \frac{TF_1 + TF_2 + \cdots + TF_m}{n \cdot m} + \frac{1}{2 \cdot n}$, em que:
  - TF_i é a posição do caso de teste que revela o i-ésimo defeito;
  - m é a quantidade de defeitos que o conjunto de testes pode revelar;

– n é a quantidade de casos de teste que compõem o conjunto de testes.

O valor de APFD é máximo quando, considerando os m defeitos possíveis, os primeiros m casos de teste os revelam.

F-measure é expressa pela quantidade de casos de teste executados até que o primeiro defeito seja revelado. Seu valor é máximo quando o primeiro caso de teste da sequência revela um defeito qualquer;

- A **variável independente** e também **fator** do experimento é **técnicas de priorização de casos de teste**, que assume alguns valores que serão discutidos logo adiante.

O foco do experimento é analisar o impacto das técnicas na capacidade de revelar defeitos de um conjunto priorizado de casos de teste. Para isto, é preciso selecionar os **tratamentos do fator**, de modo a representar bem o Estado da Arte. Após a realização da revisão sistemática, detalhada no Capítulo 3, as seguintes técnicas (tratamentos) foram selecionadas:

- Técnica Ótima (**optimal**): esta é bastante utilizada em experimentos por servir de limite superior para técnicas sendo avaliadas, uma vez que apresenta o melhor resultado que poderia ser obtido. Mas, por contar com informações não disponíveis fora de um ambiente controlado, ela é inviável na prática, pois conta com conhecimento capaz de maximizar a métrica de análise que, no caso do presente experimento, o posicionamento dos defeitos nos modelos das aplicações. Para maximizar APFD, basta para cada defeito posicionar um caso de teste no início do conjunto priorizado e maximizar *F-measure* é consequência natural, pois sempre o primeiro caso de teste revelará algum defeito;

- Técnica Aleatória (**random**): também muito utilizada em experimentos [JZCT09, EMR02] por ser uma forma fácil e rápida de efetuar uma seleção e é frequentemente posta como limite inferior de desempenho, uma vez que não é seguida nenhuma heurística que possa guiar a escolha. Para realizar a priorização aleatória, basta tomar o próximo caso de teste aleatoriamente até que todos sejam considerados;

- Técnica Aleatório-Adaptativa com Distância de Jaccard (**ART_jac**): técnica que segue o paradigma aleatório-adaptativo que faz uso do conceito de distância entre casos de

teste. Para se calcular a distância entre os casos de teste em questão usa-se a distância utilizada por Jiang et al. [JZCT09] denominada distância de Jaccard. Os detalhes do cálculo desta distância foram apresentados na Seção 4.1.3 (Capítulo 4);

- Técnica Aleatório-Adaptativa com Distância de Manhattan (**ART_man**): segue o mesmo paradigma que a técnica anterior, mas com a alteração da função de medida de distância. Para se calcular a distância entre os casos de teste em questão usa-se a distância utilizada por Zhou [Zho10] denominada distância de Manhattan. Os detalhes de como proceder para calcular esta distância foram apresentados na Seção 4.1.3 (Capítulo 4);

- Fixed Weights (**fw**): técnica definida por Sapna e Mohanty [SM09] que define pesos para os elementos do diagrama de atividades que recebe como entrada e sugere a ordenação de acordo com esses pesos. Detalhes de funcionamento e implementação foram apresentados na Seção 4.1.1;

- STOOP (**stoop**): técnica apresentada por [KSSM09] que se baseia em elementos estruturais do modelo de entrada para efetuar a priorização dos casos de teste. Uma explicação mais detalhada da técnica foi apresentada na Seção 4.1.2.

Depois de expostas todas as técnicas que compõem o experimento, elas serão doravante tratadas por seus mnemônicos entre parênteses e sempre em negrito, com a finalidade de simplificar a leitura.

5.2.3 Hipóteses Formais

Com base nas variáveis do experimento e na hipótese geral, é possível derivar as hipóteses formais que serão testadas no experimento. Como são duas variáveis dependentes, as hipóteses devem comtemplá-las separadamente. Para a APFD são elas:

$$H_0^1 : APFD_i = APFD_j, \forall i, j \in \{optimal, random, ART\_jac, ART\_man, fw, stoop\}$$
$$H_1^1 : APFD_i \neq APFD_j, \exists i, j \in \{optimal, random, ART\_jac, ART\_man, fw, stoop\}$$

A hipótese nula H_0^1 representa a situação em que todas as técnicas tem comportamento estatisticamente semelhante, no nível de confiança considerado e a alternativa (H_1^1) é o caso

em que existe diferença estatística entre pelo menos um par de técnicas. A mesma situação se aplica para a métrica *F-measure*:

$H_0^2 : F_i = F_j, \forall i, j \in \{optimal, random, ART\_jac, ART\_man, fw, stoop\}$
$H_1^2 : F_i \neq F_j, \exists i, j \in \{optimal, random, ART\_jac, ART\_man, fw, stoop\}$

Diversos testes estatísticos são utilizados para testar as hipóteses postuladas apenas dependendo do projeto do experimento e se as premissas para os testes são confirmadas pelos dados.

5.2.4 Objetos do Experimento

Os objetos deste experimento são os modelos das aplicações que servem de entrada para as técnicas. Esse modelos foram construídos usando a linguagem de modelagem *Unified Modelling Language* - UML, mais especificamente através do diagrama de atividades e utilizando o procedimento descrito na Seção 5.3.1, com base nos documentos de especificação de duas aplicações, LTS-BT e PDFsam. Estas aplicações serão descritas a seguir.

LTS-BT

LTS-BT (*Labelled Transition System-Based Testing*) é uma ferramenta utilizada para dar suporte ao processo de TBM, sendo capaz de realizar a geração e seleção de casos de teste baseados em modelo [CANM08].

A ferramenta LTS-BT é especificada através de um documento de casos de uso. Neste documento é representado o comportamento da aplicação, expresso através de todos os passos que um usuário pode executar, das condições a serem satisfeitas e os dos resultados esperados para cada passo. Partindo deste documento, o diagrama de atividades que modela o seu comportamento foi derivado e este é o artefato a ser utilizado no experimento.

O algoritmo de geração de casos de teste fornecido pela técnica *FixedWeighs*, que está detalhado na Seção 4.1.1, gerou 53 casos de teste, partindo do diagrama de atividades que foi obtido manualmente, com base nas informações contidas no documento de casos de uso. Para que seja possível a análise dos defeitos deste modelo, faz-se necessário o registro dos mesmos. Através da execução dos casos de teste foram observadas 14 falhas, que revelavam quatro defeitos.

PDFsam

A outra aplicação que foi modelada para fazer parte do experimento foi a *PDF Split and Merge* - PDFsam[1]. É uma aplicação que manipula documentos PDF, executando operações como mescla, separação, edição e geração de novos documentos PDF. É uma aplicação de código-fonte livre, distribuida sob a licença GPL versão 2 e possui uma boa quantidade de documentação. Foram utilizados dois artefatos como base para a modelagem da interação do usuário com a aplicação, também através de diagrama de atividades. O primeiro deles foi um documento de requisitos para a versão 2.1 da aplicação e o segundo artefato foi a aplicação em si, com o cuidado de ser a mesma versão contemplada pelo documento de requisitos. Tais artefatos podem facilmente ser obtidos no site do projeto[1].

O algoritmo de geração de casos de teste contido na técnica *FixedWeighs* gerou 87 casos de teste partindo do diagrama de atividades (resultado da modelagem da aplicação). Está disponível também um documento que registra as alterações que foram feitas entre as versões e nele consta algumas correções de defeitos. Deste documento, cinco defeitos foram derivados e, de todos os casos de teste gerados, 32 deles revelam esses defeitos.

Conhecendo as aplicações consideradas e que foram fornecidas como entrada para as tecnicas que compõem do experimento, é necessário definir a forma que as técnicas são executadas e como os modelos comentados nesta seção são utilizados no experimento. Tais detalhes estão contidos no projeto experimental.

5.2.5 Projeto Experimental

Uma vez que todos os elementos foram explicitados separadamente, é preciso relacioná-los de maneira a possibilitar a produção de resultados, e isso será feito nesta seção e os resultados são condensados sob a forma do projeto do experimento.

Analisando as variáveis envolvidas, apenas uma delas foi selecionada e classificada como fator, que é "técnicas de priorização de casos de teste", e, segundo a classificação apresentada por Wohlin et al. [WRH+00], como várias técnicas serão analisadas no experimento, tal projeto experimental é de **um fator com mais de dois tratamentos**, podendo ser ilustrado pela Figura 5.1.

[1]Site do projeto: http://www.pdfsam.org

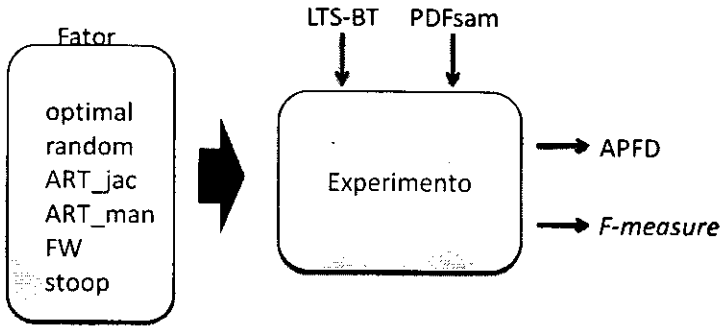


Figura 5.1: Ilustração do projeto experimental.

Este projeto experimental é considerado padrão [Jai91, WRH+00]. Isto oferece algumas facilidades, e a maior delas é já estar disponível as formas de análise de dados, os testes de hipótese a serem utilizados e o tipo de resultados esperados através deles. Com relação ao tipos de teste de hipótese, Jain [Jai91] indica a Análise de Variância - ANOVA, caso os dados sejam normais e tenham variâncias semelhantes, ou o teste de *Kruskal-Wallis* como alternativa não paramétrica.

Como são duas variáveis dependentes, dois pares de hipóteses foram postuladas e serão testadas. Como as técnicas têm influência do caráter aleatório, a quantidade de execuções necessárias para obter signficância estatística para testá-las foi determinada com base em uma amostra de 31 replicações; esta é considerada suficientemente grande [Jai91]. Para calcular a quantidade de execuções de cada técnica com cada modelo de aplicação foram extraidos os dados de APFD e de *F-measure*. Para todas as amostras técnica/modelo, intervalos de confiança deste cálculo com nível de confiança igual a 95%, por consequência, significância $\alpha = 0,05$, foi calculada a quantidade pela Equação 5.1 [Jai91].

$$repli = \left(\frac{100 \cdot Z_{\frac{\alpha}{2}} \cdot s}{r \cdot \bar{x}} \right)^2 \tag{5.1}$$

Em que:

$\mathbf{Z}_{\frac{\alpha}{2}}$ é o valor crítico da distribuição normal para o nível de significância α;

s é o desvio-padrão amostral;

r é a precisão do intervalo de confiança, medida em % da média amostral;

$\bar{\mathbf{x}}$ é a média amostral.

Como os tipos de valores das duas variáveis são diferentes, as precisões dos intervalos de confiança foram selecionadas para cada intervalo separadamente. Como o valor de APFD está entre zero e um, a precisão foi de 2% da média, o que, por exemplo, em uma média de APFD de 0,5 significa um intervalo de confiança [0,49; 0,51]. Para *F-measure* uma menor precisão de 15%, pois como os valores são naturais entre 1 e o tamanho do conjunto de casos de teste a ser priorizado, intervalos de confiança pequenos não representariam diferenças significativas, então, por exemplo uma média 4 de *F-measure* resulta no intervalo de confiança [3,4; 4,6].

É possível verificar todas as quantidades de execuções calculadas na Tabela 5.1. Os valores 0,00, tanto na Tabela 5.1a quanto na Tabela 5.1b, significam amostras constantes, o que implica que qualquer amostra satisfará a precisão do intervalo de confiança estipulado. Já para o restante dos valores, são o primeiro número natural maior que o calculado pela Equação 5.1, pois não existem frações de execução. Como a quantidade máxima é 227, cada par técnica/modelo será executado 227 vezes, mantendo assim as amostras balanceadas.

A função que calcula estas quantidades foi implementada utilizando a linguagem R que é utilizada pelo aplicativo de análise estatística homônimo[2].

Tabela 5.1: Quantidade de execuções calculadas.

(a) Para APFD

—	LTS-BT	PDFsam
optimal	0,00	0,00
random	44	17
ART_jac	25	5
ART_man	127	29
fw	1	1
stoop	0,00	0,00

(b) Para *F-measure*

—	LTS-BT	PDFsam
optimal	0,00	0,00
random	901	179
ART_jac	101	161
ART_man	104	227
fw	11	30
stoop	0,00	0,00

[2]Disponível em: http://www.r-project.org/

5.3 Instrumentação e Ferramentas

Esta ainda é uma etapa do planejamento do experimento, mas por conter várias informações, será aqui tratada separadamente. A instrumentação contém a definição de artefatos que auxiliam a realização do experimento. Tais artefatos são: os modelos das aplicações que serão entrada para as técnicas de priorização, algoritmos auxiliares, as próprias técnicas e o ambiente de execução do experimento, adicionado com o código que coleta os dados para análise.

5.3.1 Modelagem das Aplicações

Para ilustrar o processo de modelagem de aplicações utilizando o diagrama de atividades, um exemplo é visto logo em seguida.

Fluxo Principal

Passo	Condição do Sistema	Resposta do Sistema
Preencher campo com o nome do usuário		A caixa de texto mostra o nome do usuário preenchido
Clique no botão "OK"	O usuário está cadastrado no sistema	Uma janela com todas as informações do usuário é exibida

Fluxo Alternativo

Passo	Condição do Sistema	Resposta do Sistema
Clique no botão "OK"	O usuário não está cadastrado no sistema	Um alerta de erro é exibido com a frase "Usuário não cadastrado"

Figura 5.2: Exemplo de descrição de requisitos de uma aplicação hipotética

Inicialmente os requisitos do sistema, como pode ser visto um exemplo na Figura 5.2, são definidos. Apenas um trecho de uma aplicação hipotética é utilizado nesta ilustração. Tais requisitos estão expressos em uma notação tabular, em que os passos a serem realizados pelo usuário durante o uso da aplicação, as condições para a execução de um passo e o resultado esperado de um passo são separados simplificando a leitura e a compreensão do comportamento da aplicação.

Agora considerando a descrição da Figura 5.2, como a coluna **Passo** representa uma ação a ser desempenhada pelo usuário no sistema, é possível tomar cada passo como uma ação re-

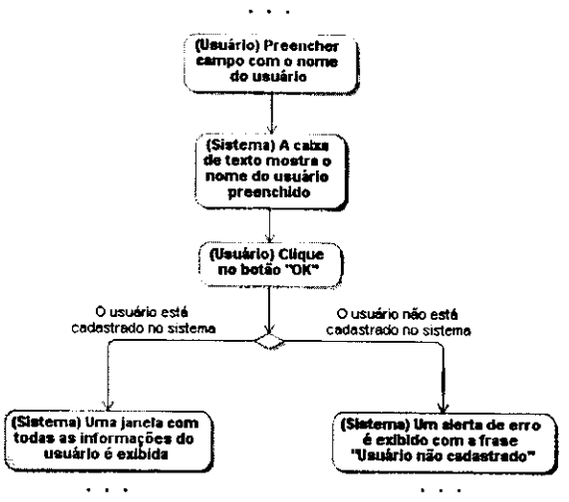


Figura 5.3: Exemplo de trecho de diagrama de atividades construído com origem na descrição da Figura 5.2

alizada pelo usuário. A coluna **Condição do Sistema** representa uma condição satisfeita pelo sistema para a continuidade do fluxo, então ela foi tomada como sendo um novo caminho saindo de um nó de decisão. A coluna **Resposta do Sistema** representa o que o sistema deve fazer caso receba um passo correspondente e que as condições associadas sejam satisfeitas, então é possível relacioná-la com uma ação realizada pelo sistema e por fim, a sequência de passos é encadeada utilizando fluxos de controle. As raias foram utilizadas para definir qual objeto executa a ação, mas ao invés de utilizar o recurso que envolve linhas horizontais ou verticais para delimitar o espaço das ações para um determinado objeto, o nome do objeto é posto entre parênteses no início do texto da ação, como pode ser visto, por exemplo, na Figura 5.3.

Esta sequência de tarefas para partir de uma notação de tabular que expressa os passos do usuário e respostas do sistema e gerar um diagrama de atividades não foi encontrada na literatura e, assim, pode ser considerada uma contribuição do presente trabalho.

Com tais diretrizes, é possível ver na Figura 5.3 o trecho de diagrama de atividades que representa a descrição dos requisitos considerada para o exemplo. LTS-BT foi modelada utilizando esta descrição, através do documento de casos de uso da aplicação. Já a aplicação PDFsam foi modelada tendo como base o documento de requisitos disponível no site do projeto que, apesar de não ter o formato tabular como no caso de LTS-BT, contém todas as funcionalidades da aplicação e todos os passos que podem ser realizados pelo usuário, dessa

forma o procedimento de modelagem foi semelhante.

A técnica **FixedWeights** requer que o comportamento do sistema seja representado por **apenas** um diagrama de atividades, pois não é dotada de uma etapa de mescla dos diagramas que representariam cada caso de uso da aplicação, o que será incluído em uma próxima versão da técnica, segundo Sapna e Mohanty [SM09]. Já **STOOP** requer um conjunto de diagramas de sequência que representem o comportamento do sistema considerado [KSSM09], e antes que a técnica efetue a priorização, existe um passo de mescla deste conjunto de diagramas, como pode ser visto na Seção 4.1.2.

Ambas as técnicas possuem algoritmos de geração de casos de teste próprios sugeridos pelos respectivos autores, que são modificações no critério de parada de uma busca em profundidade (*Depth-First Search* - DFS). As técnicas aleatório-adaptativas, **ART_jac** e **ART_man**, **random** e **optimal**, requerem apenas um conjunto de casos de teste como entrada, não restringindo a maneira que foram gerados.

Dada esta diferença nas entradas, as aplicações LTS-BT e PDFsam foram modeladas através de diagramas de atividades, e por ser mais abrangente e gerar mais casos de teste que os outros, o conjunto de casos de teste derivado pelo algoritmo de geração de casos de teste de **FixedWeights** foi utilizado para alimentar todas as técnicas, mas no caso particular de **STOOP**, tais casos de teste foram utilizados para originar um conjunto de diagramas de sequência que representam o sistema, satisfazendo o seu requisito de entrada, de acordo com ao Figura 5.4.

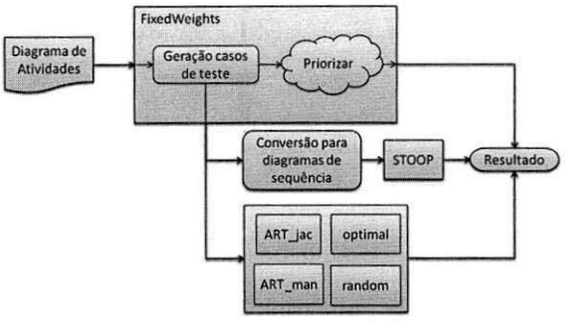


Figura 5.4: Fluxo de preparação das técnicas na execução do experimento, englobando geração de casos de teste e entradas das técnicas envolvidas.

Como um diagrama de sequência exibe a troca de mensagens entre objetos descrevendo a interação entre eles e o diagrama de atividades diferencia as atividades realizadas pelo usuá-

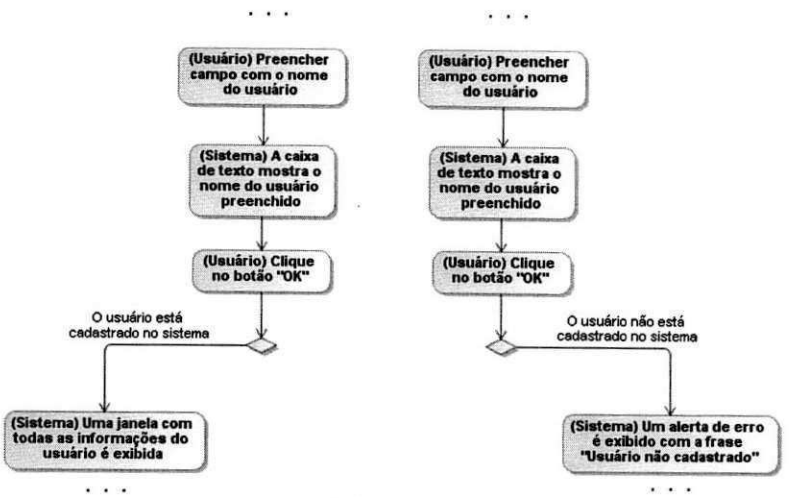


Figura 5.5: Exemplo de trechos de casos de teste gerados com origem no diagrama de atividades da Figura 5.3

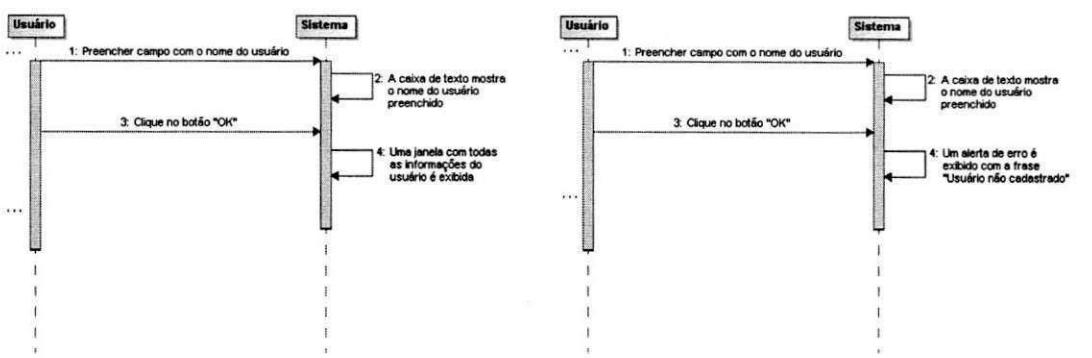


Figura 5.6: Exemplo de diagramas de sequência convertidos com base nos casos de teste da Figura 5.5

rio e pelo sistema, é possível associar as atividades de um determinado objeto a mensagens enviadas por este objeto. Partindo deste princípio, o algoritmo de conversão itera sobre as atividades do caso de teste e cada uma se transforma em uma mensagem, cuja origem é o objeto que realiza a atividade e o destino é, caso a origem seja o usuário, o sistema e vice-versa. Partindo do diagrama de atividades da Figura 5.3 é possível com o algoritmo de geração de casos de teste de **FixedWeights** gerar os dois casos de teste vistos na Figura 5.5.

É possível ver um exemplo desta conversão partindo dos casos de teste da Figura 5.5 na Figura 5.6, onde, respectivamente, o diagrama de sequência da esquerda corresponde ao diagrama de atividades da esquerda e o mesmo é válido para os da direita em cada figura.

Os diagramas de atividades frutos da modelagem de LTS-BT e de PDFsam foram analisados por outros pesquisadores, com a finalidade de analisar se a modelagem foi feita de maneira satisfatória, ou seja, se todas as funcionalidades e cenários descritos pelos documentos de requisitos foram contemplados nos respectivos diagramas de atividades.

Para a modelagem, foi utilizada a aplicação MagicDraw UML 16 [3], que provê a capacidade de edição nos mais diversos diagramas UML, inclusive o diagrama de atividades, que foi o modelo utilizado para o presente experimento. A interface gráfica do MagicDraw facilita a tarefa de edição dos diagramas e, para a manipulação programática dos diagramas gerados, a ferramenta é capaz de exportar os diagramas em um arquivo que contém uma representação textual dos mesmos baseada na liguagem *eXchange Markup Language* - XML. Tais arquivos são lidos por LTS-BT e é feita a tradução para um objeto Java correspondente. Uma resumo de como os diagramas de atividades são processados para o experimento pode ser visto na Figura 5.7.

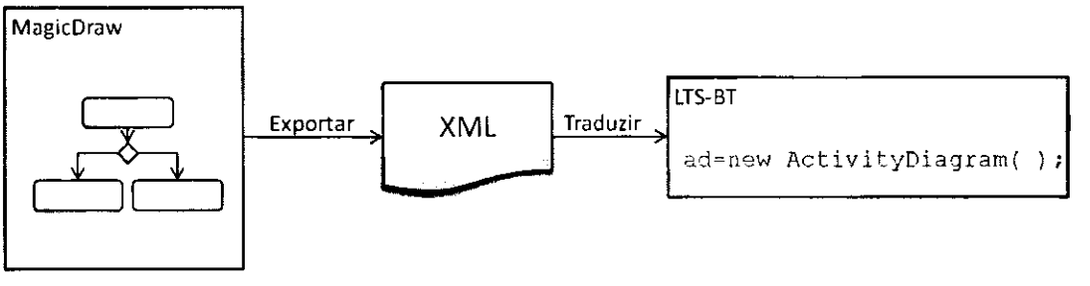


Figura 5.7: Fluxo de tarefas dos diagramas de atividades modelados em MagicDraw e importados em LTS-BT

A ferramenta LTS-BT foi o suporte operacional do presente experimento e a próxima seção trata deste aspecto.

5.3.2 Implementação das Técnicas

As técnicas pesquisadas e selecionadas, conforme visto na Seção 5.2.2, foram implementadas na ferramenta LTS-BT, de maneira a fazer parte de sua arquitetura. Ela é uma ferramenta que tem por finalidade dar suporte ao processo de TBM, provendo algoritmos de geração e seleção de casos de teste [CANM08].

[3]Disponível em: https://www.magicdraw.com/

A arquitetura base de LTS-BT utiliza o padrão *facade* para prover as funcionalidades oferecidas pela lógica da aplicação para a interface gráfica. A classe de fachada da aplicação, como pode ser visto na Figura 5.8, se comunica com os diversos elementos responsáveis pelas mais diversas funções, dentre elas algumas se destacam para o intento deste experimento como a geração de casos de teste, a tradução dos modelos de entrada e os próprios diagramas, de atividades e de sequência. Além destes elementos, LTS-BT também contém técnicas de seleção de casos de teste [CNM07], de redução de conjuntos de casos de de teste [BCM+10] e ainda de seleção de casos de teste de regressão [dON11]. Além do uso normal das técnicas em um processo de desenvolvimento de *software*, cumprindo suas tarefas de TBM, LTS-BT possui um módulo de experimento, onde métricas são implementadas a depender do objetivo dos experimentos conduzidos.

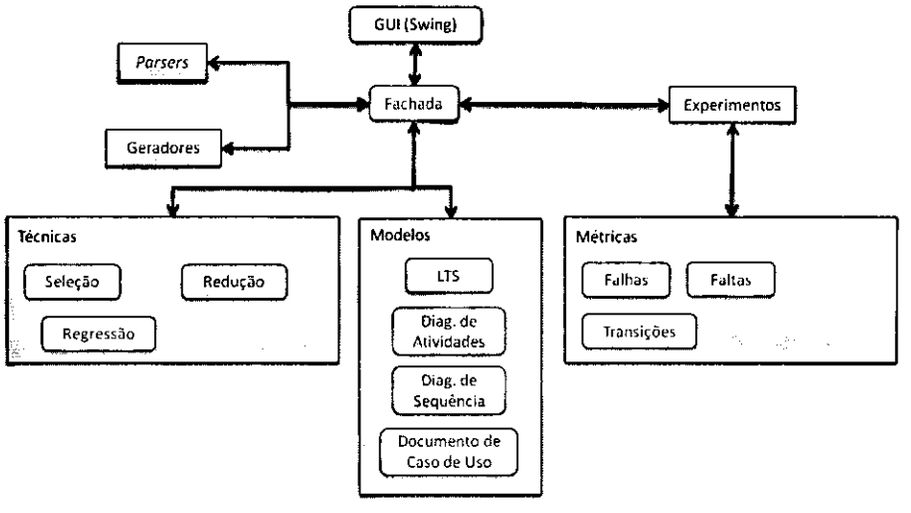


Figura 5.8: Esquema da arquitetura de LTS-BT antes da instrumentação para o experimento

Com a finalidade de conduzir o experimento definido neste capítulo, as técnicas e métricas componentes do experimento foram implementadas e os módulos alterados pela suas adições estão marcados em preto na Figura 5.9. Inicialmente, as técnicas selecionadas para compor o experimento foram codificadas e juntas compõem o novo elemento "Priorização" no módulo "Técnicas" de LTS-BT, conforme a figura supracitada. Para avaliar o desempenho das técnicas de priorização durante o estudo experimental, as métricas APFD e *F-measure* foram desenvolvidas e adicionadas ao módulo "Métricas". E para a execução própria do experimento, uma classe Java foi desenvolvida para instanciar, configurar as técnicas, controlar a sua execução e coletar os dados referentes as métricas, e a mesma foi associada ao módulo

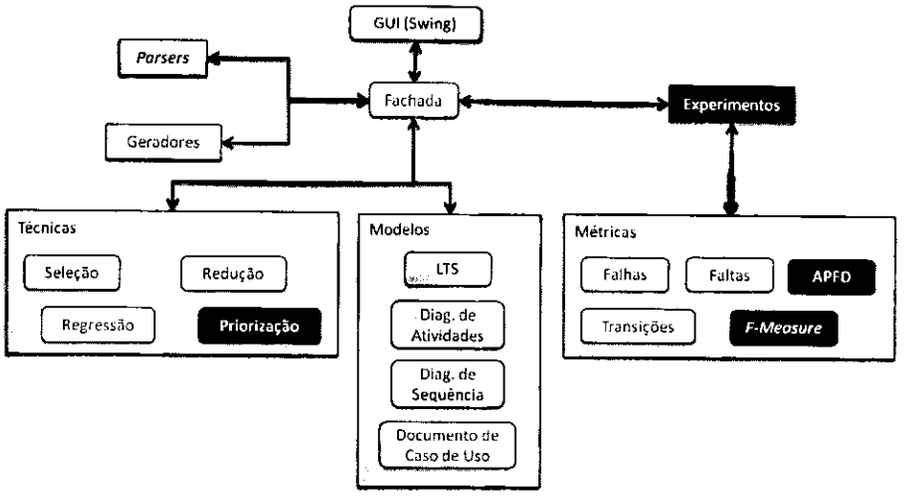


Figura 5.9: Esquema da arquitetura de LTS-BT depois da instrumentação para o experimento

"Experimentos", conforme pode ser visto na Figura 5.9.

Como cada técnica tem necessidades diferentes para seu funcionamento, algumas decisões foram tomadas para uniformizar as entradas, influenciando o mínimo possível na semântica das aplicações. A técnica

Todos os elementos necessários para a realização do experimento foram implementados utilizando a linguagem de programação Java e utilizando o ambiente integrado de desenvolvimento Eclipse[4], aliado a um repositório de controle de versões CVS.

5.3.3 Validação e Verificação

Para os modelos das aplicações, a validação foi feita analisando se todos os cenários de uso explicitados no documento de requisitos, tanto de PDFsam quanto de LTS-BT, estão representados nos respectivos modelos. Para esta análise, três pesquisadores fizeram a leitura dos modelos e verificaram se todas as funcionalidades estavam bem representadas nos modelos.

Já para das técnicas e métricas, a validação foi feita de acordo com as instruções e exemplos dados pelos autores dos trabalhos de origem, analisando se os mesmos resultados sugeridos nos artigos são obtidos pela implementação. E para a verificação, mais modelos "toy" foram criados para alimentar as técnicas e observar se as saídas estão em conformidade com os resultados previstos pelo autor das técnicas e métricas. Tais checagens foram feitas usando

[4]Site do projeto http://www.eclipse.org

a biblioteca JUnit[5], que automatiza o teste de unidade, provendo métodos de comparação de resultados e exibição de veredito de teste.

Neste ponto, os elementos componentes do experimento estão prontos para serem executados e originar os dados para a realização da análise estatística.

5.4 Análise e Interpretação

Nesta seção, os dados obtidos com a execução do experimento são exibidos através de dispositivos já expostos na Seção 2.5.4. Todos os testes realizados são comentados, juntamente com os seus resultados, e são tecidas as primeiras considerações sobre os resultados adquiridos. Todos os testes aqui realizados tem nível de significância de $0,05$, o que segundo Jain [Jai91] é um nível de significância aceitável para comparações. Mais comentários e discussões sobre justificativas para os resultados sugeridos são tecidos na Seção 5.5.

Com a quantidade de execuções calculada resultando em 227, o mesmo projeto foi realizado com esta nova quantidade de execuções, e para os dados obtidos nesta, a seguinte abordagem de análise de dados foi utilizada:

1. Estatística descritiva: primeiro contato com os dados. Dele se consegue verificar quais os tratamentos apresentam desempenho comparável e com isso, avaliar quais deles necessitam ser comparados através do teste estatístico;

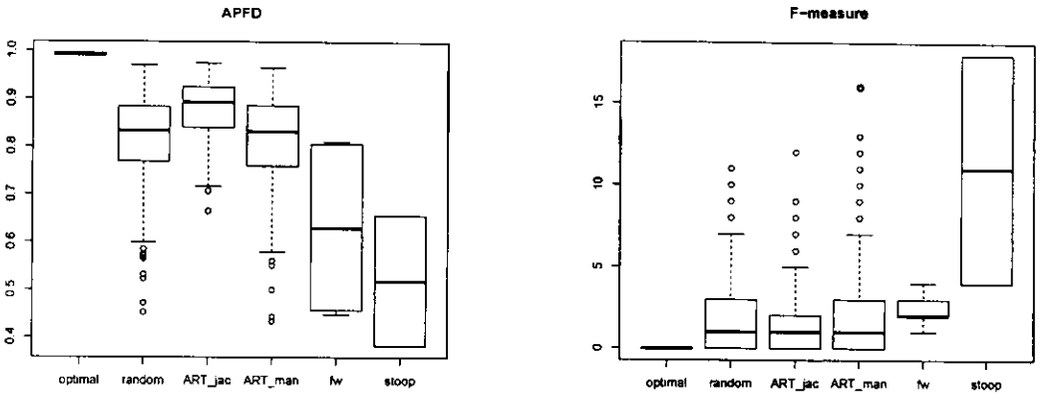
2. Verificação de premissas dos dados: analisar se os dados colhidos das amostras provém de populações que seguem distribuição normal. Em caso positivo, verificar a igualdade de variâncias [QK02];

3. Conduzir o teste apropriado aos dados: com base nas verificações das suposições acerca dos dados, selecionar e executar o teste de hipótese indicado;

4. Conduzir um pós-teste: caso seja necessário, executar um pós-teste com a finalidade de sugerir resultados mais precisos.

Os dados resultantes dos dois modelos de aplicação considerados foram agrupados com a finalidade de analisar o desempenho geral das técnicas. Seguindo a abordagem descrita anteriormente, a Figura 5.10 condensa os dados obtidos da execução do experimento.

[5]Site do projeto http://http://www.junit.org/



(a) APFD (b) *F-measure*

Figura 5.10: Sumário dos dados obtidos com a execução do experimento através de gráficos *boxplot*.

A Figura 5.10a traz o desempenho das técnicas para a métrica APFD. Quanto mais o *box-plot* se desloca para cima, melhor é o desempenho, o que é ilustrado pelo gráfico relativo a **optimal**. É possível verificar que já existe uma tendência de que as técnicas mais baseadas na aleatoriedade, que são **random**, **ART_jac** e **ART_man**, se comportem melhor, de maneira geral, que as baseadas puramente em elementos de cobertura, que são **fw** e **stoop**. Mas também é possível verificar que as baseadas na estratégia aleatória produzem vários resultados destoantes em relação à média, sendo caracterizados por pequenas circunferências (conhecicas como *outliers*) nos gráficos *box-plot*, o que acentua a necessidade de analisar um conjunto determinado e calculado de repetições.

Já a Figura 5.10b expõe o desempenho das técnicas com relação à *F-measure* e quanto mais deslocado para baixo está o gráfico, melhor é o desempenho, como representado por **optimal**. Para esta métrica, existe um equilibrio maior entre as técnicas e o mesmo se pode dizer sobre valores destoantes produzidos pelas técnicas fundamentadas na escolha aleatória.

Analisando a Figura 5.10a e a Figura 5.10b, é possível verificar que existe sobreposição dos *box-plots* das técnicas, exceto **optimal**, então o teste de hipótese deve ser aplicado a todos os tratamentos [Jai91]. A inclusão de **optimal** no teste de hipótese causaria a refutação automática das hipóteses nulas, pois ela é visualmente superior a todas as outras, dessa forma ela será removida dos testes de hipótese e estes serão conduzidos com as amostras referentes

as cinco técnicas restantes.

5.4.1 Tratamento da Métrica APFD

Uma vez que todos os tratamentos para as métricas necessitam ser testados, com a exceção de **optimal**, é necessário verificar se algumas premissas são satisfeitas pelos dados obtidos com a finalidade de definir qual teste se aplica a eles. As hipóteses a serem testadas para a métrica APFD são:

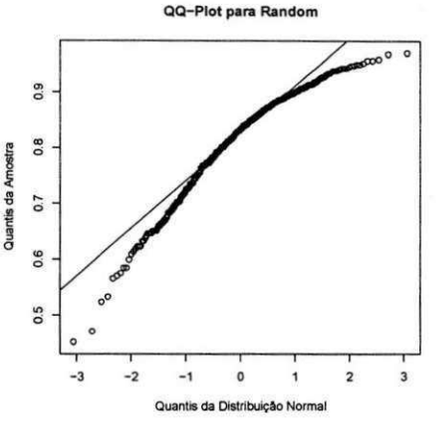
$$H_0 : APFD_i = APFD_j, \forall i, j \in \{optimal, random, ART\_jac, ART\_man, fw, stoop\}$$
$$H_1 : APFD_i \neq APFD_j, \exists i, j \in \{optimal, random, ART\_jac, ART\_man, fw, stoop\}$$

A primeira suposição é a de normalidade, ou seja, verificar se as amostras advém de populações normais. Para isso, basta verificar na Figura 5.11 os gráficos quantil-quantil com a distribuição normal, que consiste em um teste visual de normalidade. As pequenas circunferências representam os quantis dos dados testados e a linha contínua representa os quantis da distribuição normal. Então, quanto mais próximo da linha os dados se encontrarem, melhor é o grau de normalidade.

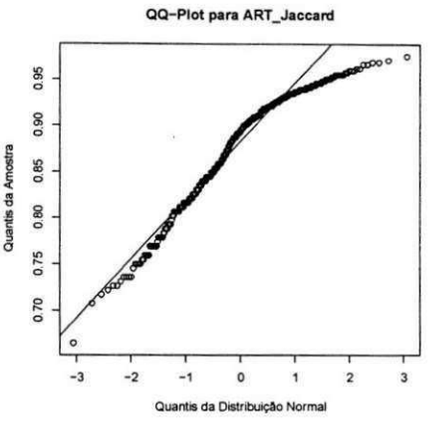
Como é possível verificar em todas as figuras componentes da Figura 5.11, nenhuma das amostras se situa próximo da linha que representa os quantis da distribuição normal, o que significa que nenhuma das amostras advém de população que segue distribuição normal, que implica em que o teste de hipótese utilizado não pode ser paramétrico. O teste não paramétrico indicado para a comparação de mais de dois tratamentos é o teste de Kruskal-Wallis [QK02, Jai91, WRH+00].

A Figura 5.12 contém na primeira e segunda linhas, os comandos necessários para a junção das amostras a serem testadas em uma estrutura de dados apropriada, excluindo a que provém de **optimal**, para a execução propriamente dita do teste. No resultado do teste, é mostrado o valor da estatística de teste, os graus de liberdade e, o mais importante, o p-valor do teste. Comparando o p-valor, que é igual a $2,2 \cdot 10^{-16}$, com a significância fixada de 0,05, a hipótese nula deve ser refutada.

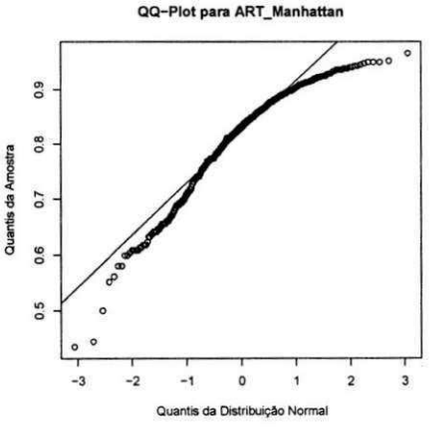
A refutação da hipótese nula de igualdade H_0 implica em dizer que existe alguma diferença entre os tratamentos, mas essa diferença pode ser em apenas um par de técnicas ou todas elas entre si. Para determinar as diferenças, faz-se necessário um pós-teste que efetue



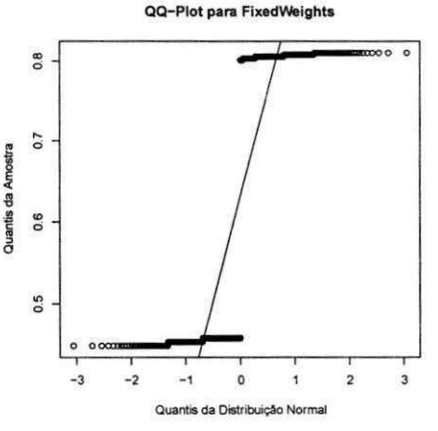
(a) Random



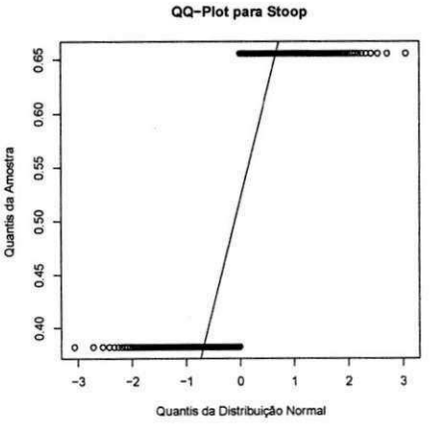
(b) ART_Jac



(c) ART_Man



(d) Fixed Weights



(e) Stoop

Figura 5.11: Gráficos quantil-quantil com a distribuição normal para os valores de APFD das amostras.

```
> stackedAPFD <- stack(APFD, select= -Optimal)
> kruskal.test(values~ind, data = stackedAPFD)

        Kruskal-Wallis rank sum test

data:  values by ind
Kruskal-Wallis chi-squared = 1301.882, df = 4, p-value < 2.2e-16
```

Figura 5.12: Resultado do teste de Kruskal-Wallis para APFD

comparações dois a dois entre os tratamentos e a alternativa não paramétrica utilizada foi um conjunto de testes de Mann-Whitney com a correção da significância de cada teste pelo método de Bonferroni para que a significância de toda a família de testes seja igual a 0,05.

Para fazer a comparação dois a dois das cinco técnicas, faz-se necessário executar dez testes de Mann-Whitney, cada um deles com nível de significância, corrigido pela método de Bonferroni, de 0,005, que é igual a significância desejada de 0,05 dividido pela quantidade de testes a serem realizados, que são 10. A hipótese nula de cada um destes testes é que não existe diferença entre as duas técnicas envolvidas e a alternativa é de que existe alguma diferença. Como são apenas duas técnicas, é possível saber qual apresenta melhor desempenho.

A Tabela 5.2 mostra todos os resultados das comparações. Cada linha representa uma das dez comparações feitas e apresenta também cinco colunas, que representam:

- Comparação: as técnicas que estão sendo comparadas na linha da tabela;
- Lim. Inf.: limite inferior do intervalo de confiança da diferença entre as técnicas comparadas;
- Diferença: diferença média entre as técnicas;
- Lim. Sup.: limite superior do intervalo de confiança da diferença entre as técnicas comparadas;
- P-valor: o p-valor do teste realizado.

Analisando a coluna do p-valor, é possível verificar que apenas uma linha apresenta um valor maior que a significância de 0,005, significando que a hipótese nula de igualdade entre as técnicas não pode ser refutada, o que implica que não há diferença estatística entre elas,

Tabela 5.2: Comparações entre técnicas no pós-teste para APFD aplicando a correção de Bonferroni nos níveis de significância, explicitando os intervalos de confiança de cada diferença.

Comparação	Lim. Inf.	Diferença	Lim. Sup.	P-valor
random - ART_jac	$-0,0685$	$-0,0551$	-0.0414	$2.2 \cdot 10^{-16}$
random - ART_man	-0.0141	$0,0013$	0.018	$0,7892$
random - fw	$0,1035$	$0,1416$	0.236	$2.2 \cdot 10^{-16}$
random - stoop	0.2491	$0,2721$	0.3113	$2.2 \cdot 10^{-16}$
ART_jac - ART_man	$0,0424$	$0,0566$	0.0707	$2,2 \cdot 10^{-16}$
ART_jac - fw	0.1356	$0,1912$	$0,3444$	$2,2 \cdot 10^{-16}$
ART_jac - stoop	0.2858	$0,3158$	$0,4197$	$2,2 \cdot 10^{-16}$
ART_man - fw	0.1058	$0,138$	$0,2217$	$2,2 \cdot 10^{-16}$
ART_man - stoop	0.2499	$0,2688$	$0,2971$	$2,2 \cdot 10^{-16}$
fw - stoop	$0,0754$	$0,11$	$0,1479$	$2,2 \cdot 10^{-16}$

que são **random** e **ART_man**. Todos os outros pares têm seus p-valor menores que o nível de significância, incorrendo em diferença entre as técnicas envolvidas. Com o resultado da tabela é possível elaborar uma sequência de desempenho decrescente entre as técnicas com relação a APFD:

optimal > ART_jac > random = ART_man > fw > stoop

A investigação com relação a APFD culmina com o resultado de que, com relação à detecção de defeitos, com confiança estatística de 95%, a técnica aleatório adaptativa que usa a função da distância de Jaccard tem melhor comportamento dentre todas as técnicas investigadas neste estudo. Logo em seguida, a técnica puramente aleatória e a aleatório adaptativa que usa a distância de Manhattan, que não tiveram diferença estatística na significância considerada. E por fim, as técnicas *Fixed Weights* e STOOP que obtiveram desempenho muito abaixo das anteriores.

De maneira geral, as técnicas que seguem a estratégia aleatória, seja puramente como **random** ou parcialmente como **ART_jac** e **ART_man**, obtiveram resultados muito bons comparadas às técnicas que são baseadas apenas em elementos de cobertura consideradas no estudo experimental. Tal conclusão será melhor discutida na Seção 5.5, relacionando

o desempenho das técnicas com características dos modelos das aplicações consideradas, suposições feitas pelas técnicas e ainda seus respectivos desempenhos assintóticos.

5.4.2 Tratamento da Métrica *F-measure*

A análise comparativa das técnicas para a capacidade de revelar a primeira falha, medida pela métrica *F-measure*, foi feita testando o par de hipóteses:

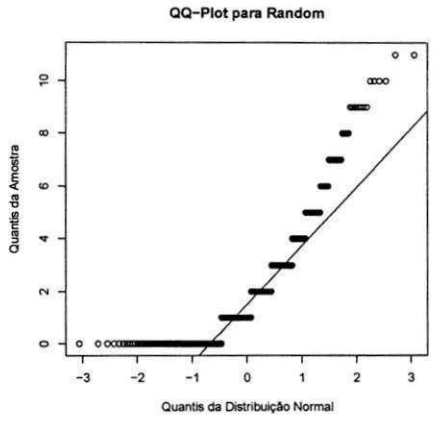
$H_0 : F_i = F_j, \forall i, j \in \{optimal, random, ART\_jac, ART\_man, fw, stoop\}$
$H_1 : F_i \neq F_j, \exists i, j \in \{optimal, random, ART\_jac, ART\_man, fw, stoop\}$

De maneira análoga ao tratamento feito para APFD, para *F-measure* os dados foram colhidos na etapa de operação do experimento e, como visto na Figura 5.10, exceto **optimal**, todas as outras técnicas necessitam ser comparadas através de testes de hipótese. Mas para definir qual teste utilizar, ainda é necessário efetuar a verificação das premissas nos dados colhidos.

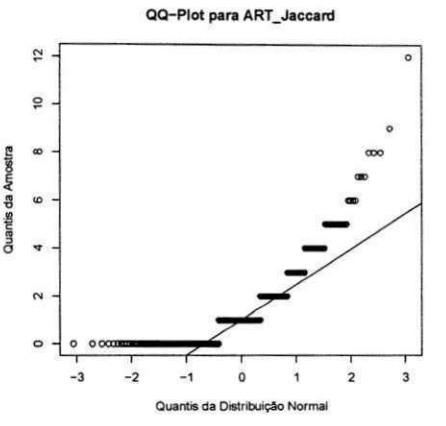
A primeira suposição é a de que os dados advém de populações que seguem distribuição normal, e, na Figura 5.13, é possível ver o teste visual de normalidade feito através de gráficos quantil-quantil de cada amostra com a distribuição normal teórica. Como, em todos os gráficos, os pequenos círculos que simbolizam os quartis dos dados sendo testados não coincidem com a reta que simboliza os quartis da distribuição normal, os dados não pertencem a populações normais, o que implica ainda que testes paramétricos não podem ser utilizados. Para este contexto, o teste utilizado foi o teste de Kruskal-Wallis [QK02, Jai91, WRH+00], da mesma forma que na análise feita na Seção 5.4.1.

O teste não-paramétrico de Kruskal-Wallis foi realizado no mesmo nível de significância de 5%, como todos os testes feitos ao longo da pesquisa e a Figura 5.14 mostra os comandos necessários para a execução do teste nas duas primeiras linhas, e o resultado do mesmo. Avaliando o p-valor do teste, por ser bem menor que a significância adotada, a hipótese nula foi rejeitada, o que implica que pelo menos um par de técnicas são diferentes com relação a *F-measure*.

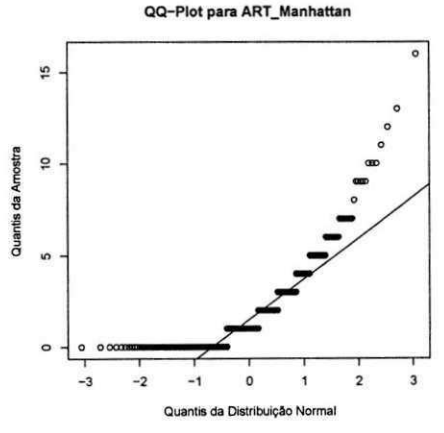
Para mensurar quais são as diferenças entre todas as técnicas, da mesma forma que o tratamento de APFD, na Seção 5.4.1, é necessário conduzir um pós-teste. Para o caso em que



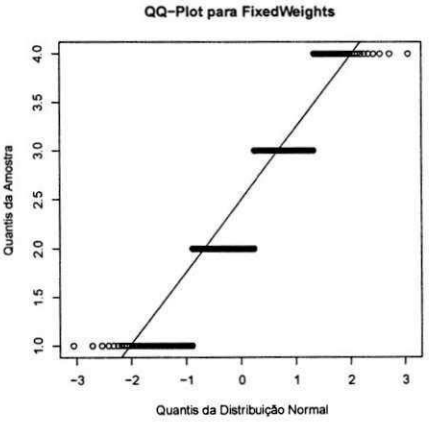
(a) Random



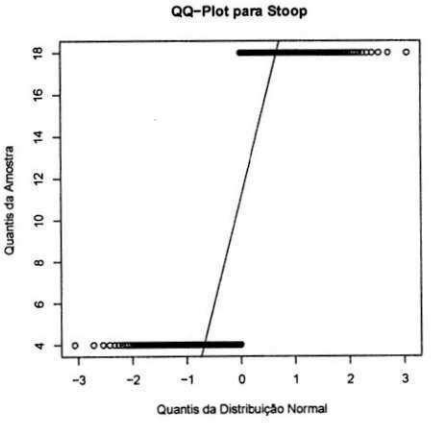
(b) ART_Jac



(c) ART_Man



(d) Fixed Weights



(e) Stoop

Figura 5.13: Gráficos quantil-quantil com a distribuição normal para os valores de *F-measure* das amostras.

```
> stackedF <- stack(FMeasure, select= -Optimal)
> kruskal.test(values~ind, data = stackedF)

        Kruskal-Wallis rank sum test

data:  values by ind
Kruskal-Wallis chi-squared = 946.0714, df = 4, p-value < 2.2e-16
```

Figura 5.14: Resultado do teste de Kruskal-Wallis para *F-measure*

foi realizado um teste não paramétrico para a primeira comparação entre mais de dois tratamentos, o teste de Mann-Whitney deve ser realizado entre cada um dos pares de tratamentos para que se defina onde estão as diferenças, não detectáveis pelo teste de Kruskal-Wallis realizado anteriormente.

Da mesma forma que na seção anterior para a métrica APFD, cada par de técnicas foi comparado através de testes de Mann-Whitney, cada um destes com nível de significância 0,005, corrigido pelo método de Bonferroni para que o pós-teste inteiro tenha significância de 0,05 ou 5%.

Para *F-measure*, dois pares de técnicas apresentaram desempenho estatisticamente semelhante, pois analisando a última coluna da Tabela 5.3 que mostra o p-valor de cada teste, a comparação entre **random** e **ART_man** e entre **ART_jac** e **ART_man** apresentam p-valor maior que 0,005, o que implica que as hipóteses nulas de igualdade de cada um dos testes não podem ser refutadas. O p-valor dos outros testes foram menores que a significância corrigida pelo método de Bonferroni, o que significa que são estatisticamente diferentes com relação a *F-measure* e com os valores fornecidos pelo teste sumarizado pela Tabela 5.3, é possível fornecer uma sequência crescente das técnicas com relação a esta métrica. É ela:

optimal < random = ART_man = ART_jac < fw < stoop

É necessário frisar que a igualdade **random** = **ART_man** = **ART_jac** não é transitiva, ou seja, **random** = **ART_man** e **ART_man** = **ART_jac** não quer dizer **random** = **ART_jac**. É possível ver na Tabela 5.3 no primeiro teste que o p-valor desta comparação é menor que a significância fixada, portanto existe diferença estatística entre **random** e **ART_jac**.

Tal resultado, que põe as técnicas baseadas na estratégia aleatória em vantagem com relação a métrica *F-measure* se deve principalmente à quantidade de casos de teste que exercitam os defeitos existentes no modelo da aplicação. A cada escolha feita, se nela já se encontra

Tabela 5.3: Comparações entre técnicas no pós-teste para *F-measure* aplicando a correção de Bonferroni nos níveis de significância, explicitando os intervalos de confiança de cada diferença.

Comparação	Lim. Inf.	Diferença	Lim. Sup.	P-valor
random - ART_jac	$2,3198 \cdot 10^{-5}$	$8,6625 \cdot 10^{-5}$	$9,9 \cdot 10^{-1}$	0,0019
random - ART_man	$-7,8355 \cdot 10^{-5}$	$3,4014 \cdot 10^{-5}$	$4,2698 \cdot 10^{-5}$	0,3389
random - fw	-1	$-0,99996$	$-0,99999$	$3,109 \cdot 10^{-12}$
random - stoop	-13	$-6,4099$	-4	$2,2 \cdot 10^{-16}$
ART_jac - ART_man	$-7,0398 \cdot 10^{-6}$	$-3,4504 \cdot 10^{-5}$	$3,5188 \cdot 10^{-5}$	0.0416
ART_jac - fw	$-1,00004$	$-1,00003$	$-0,99999$	$2,2 \cdot 10^{-16}$
ART_jac - stoop	-14	$-5,9826$	-4	$2,2 \cdot 10^{-16}$
ART_man - fw	-1	$-0,99995$	$-0,99997$	$2,2 \cdot 10^{-16}$
ART_man - stoop	-13	$-4,00006$	$-4,00001$	$2,2 \cdot 10^{-16}$
fw - stoop	-15	$-8,6850$	-3	$2,2 \cdot 10^{-16}$

um caso de teste que revela algum defeito, a métrica é calculada, e se não revela, a próxima escolha será feita com mais chances de acerto.

É possível efetuar análises mais acuradas das técnicas com relação às métricas utilizadas no experimento quando se leva em consideração mais características dos modelos das aplicações. Por exemplo, podemos analisar a quantidade de nós de decisão no modelo, existência e quantidade de laços, tamanho dos casos de teste gerados a partir deles e a relação entre os defeitos e os casos de teste que os revelam. Tais análises são feitas na seção seguinte.

5.5 Discussão dos Resultados

Nesta seção, os resultados dos testes estatísticos são retomados com a finalidade de perceber aspectos que suscitem generalização. Foram analisados elementos particulares de cada técnica (como suposições feitas por cada uma delas ou especificidades dos algoritmos envolvidos), características dos modelos de aplicações utilizados (por exemplo a quantidade de laços, nós de decisão e a maneira que os defeitos se situam nos modelos) e, por fim, resultados obtidos por outros autores em estudos conduzidos em outros contextos (como, por exemplo, a priorização de casos de teste baseado em código-fonte).

Tabela 5.4: Informações úteis sobre os modelos utilizados no experimento.

Modelo	LTS-BT	PDFsam
Num. Nós Escolha	26	11
Num. Laços	0	5
Num. Nós Junção	7	6
Casos de Teste	53	87
Tam. Menor	10	17
Tam. Maior	34	43
Num. Defeitos	4	5
Num. Rev. Defeitos	14	32

O experimento conduzido neste trabalho revelou que as técnicas que se baseiam em escolhas aleatórias tem melhor desempenho. Tal resultado credita-se à taxa de casos de teste que revelam defeitos. Analisando-se os dados da Tabela 5.4, nota-se que, no modelo da aplicação LTS-BT com 53 casos de teste, 14 deles revelam os quatro defeitos presentes, o que implica em uma taxa de 26,41% e em PDFsam, dos 87 casos de teste, 32 revelam os cinco defeitos presentes, o que resulta em uma taxa de 36,78%. Com tais taxas, existe uma boa probabilidade de aleatoriamente encontrar um caso de teste que revele algum defeito.

5.5.1 APFD

A métrica APFD tem por objetivo avaliar o quão rápido os defeitos presentes em uma versão de uma aplicação sob teste são revelados, sempre assumindo que, se um caso de teste falha em decorrência de um defeito, já é bastante para que esta defeito seja revelado [EMR00]. Dessa forma, **optimal**, que é inviável na prática, maximiza tal comportamento e, ao sugerir uma nova técnica, deseja-se obter desempenho o mais próximo possível deste comportamento.

Com o resultado do teste de hipótese foi dito que **ART_jac** tem melhor desempenho para APFD, seguida por **random** e **ART_man** com comportamento estatisticamente semelhante. Isto pode ser justificado, além do já exposto de maneira geral a respeito da taxa de casos de teste que revelam defeitos, pois estes casos de teste estão bem agrupados (alguns revelaram o mesmo defeito), o que confirma o resultado de Chen et al. [CKMT10] de que existe um chamado padrão de defeitos.

As técnicas baseadas na estratégia aleatório adaptativa são guiadas pela distância entre os casos de teste, como uma forma de estimar um padrão de falhas. **ART_jac** usa a função de distância de Jaccard, que assume que dois casos de teste são próximos quando contém muitos passos em comum e poucos diferentes e **ART_man** a função de distância de Manhattan que assume que casos de teste próximos são os que cobrem muitos ramos de decisão em comum. Por essa estimação de padrão de defeitos, aliada a alta taxa de casos de teste que os revelam, o desempenho destas técnicas é altamente satisfatório.

A possibilidade de sugerir bons resultados das técnicas de teste baseadas na estratégia aleatório adaptativa é estudada por Chen et al., inicialmente em [CLM04] e em seguida em [CKMT10], e aplicada especificamente na priorização de casos de teste por Jiang et al. [JZCT09] e Zhou [Zho10]. Este último, inclusive, sugere a execução de um estudo comparativo entre as funções de distância de Jaccard e Manhattan, no contexto do teste orientado ao código-fonte de aplicações como trabalhos futuros. Na presente pesquisa, tal comparação foi feita no contexto do teste baseado em modelo.

Os resultados obtidos neste trabalho, aliados aos sugeridos pelos trabalhos pesquisados, dão um indício mais forte de que o bom desempenho da abordagem aleatório adaptativa é veraz, tanto para o teste baseado em código-fonte quanto para o baseado em modelos. Isto mostra indícios de que mais investigações necessitam ser feitas incorporando tais abordagens.

Os dois últimos lugares foram assumidos pelas técnicas baseadas em cobertura, que são **fw** e **stoop**. Sua baixa taxa de detecção de defeitos deveu-se principalmente ao não alinhamento da estrutura dos modelos e a localização dos defeitos com as suposições feitas pelas técnicas. Em seguida, cada uma delas é discutida.

O autor de **fw** assume que os casos de teste com menos nós de escolha e com menos passos (considerado por ele como os fluxos principais) são mais importantes [SM09]. Tais casos de teste são ordenados e os que são mais importantes para a técnica não foram os que revelaram os defeitos, pois estes são revelados majoritariamente por casos de teste com muitos passos e com muitos nós de escolha. Apenas um dos defeitos é revelado por casos de teste com poucos nós de escolha. Dessa forma a capacidade de revelar os defeitos fica prejudicada pelas suposições feitas pelo autor da técnica.

Já em **stoop**, existe uma suposição de que um caso de teste é importante quando seus

passos são executados também por outros casos de teste. Então. quanto mais um caso de teste contém passos comuns a outros, mais importante ele é, como pode ser visto na expressão de AWPL na Seção 4.1.2. Tal suposição faz com que casos de teste que contenham muita redundância entre si sejam postos nas posições iniciais do conjunto priorizado de casos de teste. Nos modelos considerados no experimento aqui conduzido, os casos de teste que contém muitos nós de escolha que revelam os defeitos são ao mesmo tempo os mais extensos, o que torna menor o valor de AWPL associados aos casos de teste que revelam os defeitos. Assim sendo, esta mesma característica dos modelos utilizados, aliado a pressupostos não atendidos, prejudicou o desempenho de **stoop**.

5.5.2 *F-measure*

A outra métrica considerada no experimento foi uma medida de capacidade de revelar mais rapidamente o primeiro defeito, a *F-measure*. Mesmo podendo ser enxergada como sendo uma simplificação de APFD, ela consegue revelar circunstâncias que APFD não consegue. Não é complicado imaginar um conjunto já priorizado de casos de teste em que o primeiro caso de teste revela um defeito e os outros casos de teste que revelam defeitos estão no final da sequência de casos de teste. Avaliando APFD, a sequência de casos de teste tem resultado mediano, mas com relação a *F-measure* ela tem resultado máximo, pois logo o primeiro caso de teste revela um defeito. Dessa forma, ambas as métricas são complementares.

Através do experimento, foi possível sequenciar as técnicas, ordenando-as pela métrica aqui tratada, e é possível perceber que se repete o melhor desempenho de **ART_jac**, **ART_man** e **random** comparando com **fw** e **stoop**. As mesmas considerações sobre a questão da alta taxa de casos de teste que revelam defeitos que impactaram no desempenho das técnicas com relação a APFD se mostraram presentes também para *F-measure*.

Comparando com APFD, ocorreu uma pequena diferenciação entre os resultados das técnicas baseadas na estratégia aleatória. Foi possível verificar que, com a alta taxa de casos de teste que revelam defeitos, **random** tinha mais chance de tomar casos de teste que revelam defeitos logo no início. Em **ART_jac** e **ART_man**, o caráter aleatório é utilizado para a seleção do conjunto candidato, de modo que para cada seleção, a amostra se resume a dez casos de teste do conjunto de candidatos, e dentre eles pode haver ou não casos de teste que revelam defeitos. Já em **random**, a cada seleção, o caso de teste é selecionado aleatoriamente

dentre todos, o que exclui a possibilidade de não ter chance de selecionar um caso de teste que revele defeito.

Com relação a **fw** e **stoop**, as suposições feitas sobre o que é um caso de teste importante foram as grandes responsáveis pelos seus baixos rendimentos, ainda mais em **stoop**, pois os casos de teste que revelam defeitos possuem muita redundância, mas são bem extensos, inclusive os mais extensos de todos, como visto na Tabela 5.4.

5.6 Avaliação da Validade

Após o experimento definido, planejado e finalmente executado, faz-se necessário avaliar a sua validade através das ameaças à validade. Quanto menos ameaças forem identificadas, mais confiança se tem nos resultados do mesmo. Tais ameaças, como já visto na Seção 2.5.2, podem ser classificadas em quatro tipos e, nesta seção, todas serão abordadas, bem como as precauções tomadas com cada ameaça.

A validade de conclusão diz respeito a relação dos tratamentos com os resultados e segundo Wohlin et al. [WRH+00], o indevido trato estatístico ameaça esta validade. Com relação a esta validade, o presente experimento não é ameaçado, pois todos os procedimentos para o tratamento dos dados foram tomados, como a determinação do tamanho da amostra considerada através do cálculo como visto em Jain [Jai91], verificação das suposições acerca dos dados colhidos na fase de operação do experimento e com base neles foi feita a escolha dos testes de hipótese indicados, segundo diretrizes sugeridas por Wohlin et al. [WRH+00]. Por fim o pós-teste executado corretamente com a correção do nível de significância, em consonância com [Abd07]. Todos os testes foram realizados a um nível de significância de 5%, o que significa que existiu uma chance de 5% de rejeitar as hipóteses nulas indevidamente, o que é bastante aceitável para comparações [Jai91].

Com relação à validade interna, alguns problemas são identificados com relação aos modelos das aplicações e o relacionamento com os defeitos identificados nas aplicações. Uma delas é de que os autores de duas técnicas avaliadas no experimento, **fw** e **stoop**, incluiram um processo próprio de geração de casos de teste nas atividades das respectivas técnicas e isto insere uma diferença nos procedimentos na execução experimento. Tal diferença foi mantida com a finalidade de respeitar inteiramente os passos definidos pelos autores das

técnicas, fazendo com que todas as suposições feitas pelos mesmos fossem respeitadas.

Além desta ameaça, quando as aplicações foram executadas e desta execução foram relatados os defeitos, associados aos relatórios de mudanças no caso de PDFsam, os passos que modelam as funcionalidades que apresentaram tais defeitos foram adotadas como contendo um defeito e tal adoção foi uma suposição feita. Para minimizar esta ameaça, tais associações foram revistas por mais dois investigadores.

Relativo à validade de construção, o grande mecanismo utilizado para garantir que todos os constructos do mundo da teoria foram bem mapeados no experimento foi a realização da revisão sistemática, relatada integralmente no Capítulo 3. Como resultado da mesma, foi possível verificar quais as técnicas desenvolvidas exclusivamente para o contexto abordado no presente trabalho: priorização geral de casos de teste baseado em modelo, e de que maneira as mesmas são avaliadas, ou seja, quais métricas são utilizadas para aferir o desempenho das mesmas. A importância destes resultados está justamente no fato de terem sido sugeridas por um procedimento detalhado, gerenciável e verificável de aquisição de informações na literatura e fornece uma evidência mais forte de que os tratamentos refletem bem o estado apresentado pela literatura.

E por fim, com relação à validade externa, uma ameaça sempre presente em pesquisas conduzidas nesta área é a capacidade reduzida de generalizar os resultados do experimento pela pequena quantidade de aplicações utilizadas para exercitar as técnicas [dON11]. No contexto do código-fonte, existem repositórios de aplicações que são usadas por um bom tempo e já adquiriram caráter de *benchmark*, a exemplo da *suite* de aplicativos Siemens, largamente utilizada por pesquisadores [ERKM04, EMR00, EMR02, RUCH01]. Mas, quando se refere a modelos de aplicações, podendo ser descritos em UML, ou algum formalismo como máquina de estados ou mesmo sistemas de transições rotuladas, existe a escassez de aplicações que podem ser utilizadas. Mais resultados, utilizando o projeto experimental aqui mostrado, podem ser vistos em [OCM10]. Neste trabalho foi utilizado no experimento uma aplicação diferente das que foram aqui apresentadas.

Para tentar reduzir esta ameaça, neste experimento foram utilizados dois modelos de aplicações, onde já é possível ver variações nas estruturas, como pode ser visto na Tabela 5.4. Mais um indício de generalização pode ser percebido analisando o desempenho das técnicas aleatório adaptativas e, em consonância com resultados obtidos por Chen et al. [CKMT10],

os resultados sugeridos por técnicas que aplicam a estratégia aleatório adaptativa são promissores, o que suscita mais pesquisas no tema.

5.7 Considerações Finais do Capítulo

Este capítulo foi o relato do estudo experimental realizado com a finalidade de aumentar o corpo teórico sobre a priorização geral de casos de teste baseado em modelo. Para avaliar tal conceito, a literatura foi analisada, conforme relatado no Capítulo 3, com o objetivo de levantar as técnicas já desenvolvidas e as métricas utilizadas para aferir o seu desempenho e tais elementos. Juntamente com modelos de aplicações, foram formatados em uma estrutura de experimento e o mesmo fora realizado nesta pesquisa de mestrado.

Como resultados foi possível sequenciar as técnicas consideradas com relação às duas métricas consideradas no experimento. Com relação a APFD a sequência foi:

optimal > ART_jac > random = ART_man > fw > stoop

E com relação a *F-measure* a sequência definida foi:

optimal < random = ART_man = ART_jac < fw < stoop

Além destas sequências, um outro resultado apresentado foi o de relacionar o desempenho delas com as demais características do experimento, especialmente com os modelos das aplicações utilizadas e as suposições feitas por cada uma das técnicas. Um resultado interessante foi que as técnicas que se baseiam no paradigma aleatório adaptativo apresentaram desempenho bastante interessante, o que confirma resultados em outros contextos obtidos por [CKMT10]. Isto dá mais indícios de uma possível generalização do resultado deste experimento e gera margem para mais pesquisas envolvendo essa abordagem.

Capítulo 6

Considerações Finais

Neste trabalho foi feita uma investigação acerca da Priorização Geral de Casos de Teste de Caixa Preta, no contexto do Teste Baseado em Modelo. Inicialmente, foi conduzida uma revisão sistemática no mesmo contexto com a finalidade de selecionar quais técnicas e métricas seriam analisadas através do experimento.

Concluída esta revisão, foram selecionadas cinco técnicas para compor o estudo experimental. Tais técnicas foram comparadas considerando a capacidade de detecção de defeitos através de duas métricas: APFD e *F-measure*.

Por fim, através da condução de um estudo experimental, no qual as cinco técnicas selecionadas através da revisão sistemática, foram comparadas com relação a sua capacidade de revelar defeitos através das métricas acima mencionadas.

Na seção seguinte, as conclusões sobre os resultados obtidos pela revisão sistemática e pelo experimento são apresentadas.

6.1 Conclusões

Na revisão sistemática, foram analisadas as bases de trabalhos mais importantes como a IEEE, ACM e a ScienceDirect, com a finalidade de responder as perguntas:

- Quais técnicas de priorização geral de casos de teste baseado em modelo foram propostas na literatura?
- Quais métricas são utilizadas na avaliação de desempenho destas técnicas?

Conforme apresentado anteriormente, três etapas de seleção foram definidas com base nas informações dos trabalhos, sendo a primeira baseada apenas no título, a segunda apenas no resumo e a terceira em todo o artigo. Ao final delas foi possível responder as perguntas definidas.

A realização desta revisão revelou o maior enfoque dado à priorização de casos de teste orientado ao teste derivado do código-fonte das aplicações e orientados ao teste de regressão, como é possível ver nas tabelas que resumem as etapas de seleção, com o motivo das remoções e as respectivas quantidades de trabalhos (Tabela 3.2 para a primeira etapa, Tabela 3.3 para a segunda e a Tabela 3.4 para a terceira etapa de seleção).

As técnicas dirigidas ao teste de regressão contam com informações úteis para a detecção de defeitos como histórico de defeitos revelados, severidade, tempo para reabilitação e as técnicas. Entretanto, no início do processo de teste tais informações não estão disponíveis e esta é a limitação que as técnicas de priorização geral tenta lidar, utilizando basicamente elementos de cobertura para detectar defeitos mais rapidamente.

As técnicas que compuseram o experimento relatado neste trabalho foram as selecionadas através da revisão sistemática, sendo que cinco delas (**ART_jac**, **ART_man**, **fw**, **stoop** e **random**) podem ser aplicadas em um processo de teste que siga a abordagem do Teste Baseado em Modelo (TBM), e **optimal** serve apenas como limite superior de desempenho para as técnicas no experimento.

E com relação às métricas do experimento, dentre as métricas sugeridas pela revisão sistemática, duas delas foram analisadas no experimento. Uma delas é APFD, que inspira grande parte das outras métricas selecionadas pelo processo de revisão, e a outra é *F-measure*. APFD avalia a taxa que os defeitos são revelados pelo conjunto de casos de teste avaliado e *F-measure* mede a quantidade de casos de teste são necessários que sejam executados até que seja detectado o primeiro defeito. Já as outras métricas encontradas levam em consideração características que não fizeram parte do contexto do experimento conduzido, como por exemplo, custos e severidade dos defeitos e, por este motivo, não foram consideradas.

Então com base nessas técnicas e métricas, o experimento foi definido e planejado. Como norteadores do experimento foram definidos dois pares de hipóteses (nulas e alternativas), um para cada uma das métricas envolvidas. As nulas previam igualdade entre as técnicas

com relação a métrica e as alternativas, as respectivas diferenças entre as técnicas.

Considerando um nível de confiança de 95%, as hipóteses de igualdade testadas foram refutadas, significando assim que há diferenças entre a capacidade de revelar defeitos das técnicas analisadas. Pós-testes foram realizados com a finalidade de investigar melhor as diferenças entre as técnicas, e através destes testes foi possível definir a ordem de eficiência entre as técnicas. Os detalhes dos testes de hipótese conduzidos são expostos no Capítulo 5.

Os resultados obtidos por meio deste experimento confirmam resultados sugeridos por outros autores como Chen et al.[CKMT10], que afirmam existirem os chamados padrões de defeitos e que técnicas baseadas na estratégia aleatório adaptativa tentam estimar tais padrões. Tais abordagens ou são mais ou são, no mínimo, igualmente eficientes na detecção do primeiro defeito em comparação com puramente aleatória.

Além disso, o experimento revelou que as suposições feitas por **fw** e **stoop** sobre o que é um caso de teste importante a ser posto no início de um conjunto priorizado podem não ser indicadas para a detecção mais rápida de defeitos, reflexo direto do desempenho mais baixo com relação às métricas analisadas no experimento. Detalhes dessa conclusão podem ser vistos na Seção 5.5, no Capítulo 5.

Um fato interessante sobre a pesquisa aqui conduzida é que Zhou [Zho10] sugeriu como trabalhos futuros um estudo entre a técnica por ele sugerida (**ART_man**) e a definida por Jiang et al. [JZCT09] (**ART_jac**), por se tratarem de um mesmo algoritmo com a diferença apenas na forma que a distância entre dois casos de teste é calculada. Como resultado, no contexto do TBM, **ART_jac** teve melhor desempenho com relação à APFD e com relação a *F-measure* as duas técnicas não tiveram diferença estatística no nível de confiança considerado, que foi de 95%.

Algumas dificuldades foram encontradas. Durante a revisão sistemática, o grande número de trabalhos a serem analisados apenas por uma pessoa acarretou um consumo de tempo bem elevado. Uma outra dificuldade enfrentada foi a modelagem das duas aplicações utilizadas no experimento através de diagramas de atividade, o que motivou a definição de passos a serem seguidos para esta modelagem. Este conjunto de passos pode ser visto na Seção 5.3.1.

6.2 Trabalhos Futuros

Os resultados da presente pesquisa sugerem alguns estudos mais aprofundados como trabalhos futuros, são eles:

- **Investigação da abordagem aleatório-adaptativa**: por ter se mostrado bastante eficiente na detecção de defeitos através da estimação de um dito padrão de defeitos, como mostrado neste trabalho, e ainda segundo Chen et al. [CKMT10], a abordagem aleatório-adaptativa precisa ser melhor investigada. Isto pode ser feito através da proposição de novas funções de cálculo de distância entre casos de teste, da utilização em outros contextos (teste de regressão, por exemplo), ou ainda do uso desta abordagem como integrante em uma outra técnica como critério de desempate em uma escolha, por exemplo;

- **Execução de mais estudos experimentais**: por ainda conter ameaças à validade, o presente experimento pode ser melhorado de modo a oferecer evidências ainda mais fortes. Aumentar a quantidade de aplicações a serem analisadas em um experimento é uma sugestão imediata de melhoria, pois implicaria em um aumento na capacidade de generalização dos resultados obtidos. Além disto, avaliar a influência da estrutura dos modelos (quantidade de transições, nós de escolha, laços entre outros) na capacidade de revelar defeitos, tornando aspectos estruturais dos modelos fatores do experimento e avaliando a influência destes, junto com as técnicas, na capacidade de detecção de defeitos.

Referências Bibliográficas

[Abd07] Hervé Abdi. *Bonferroni and Sidak corrections for multiple comparisons*. Sage, Thousand Oaks, CA, 2007.

[Bab90] E. Babbie. *Survey Research Methods*. Wadsworth, 2 edition, 1990.

[BCM+10] Antonia Bertolino, Emanuela Cartaxo, Patrícia Machado, Eda Marchetti, and João Felipe Silva Ouriques. Test suite reduction in good order: Comparing heuristics from a new viewpoint. In Alexandre Petrenko, Adenilso Simão, and José Carlos Maldonado, editors, *Proceedings of the 22nd IFIP International Conference on Testing Software and Systems: Short Papers*, pages 13–17. Centre de Recherche Informatique de Montréal, CRIM, Outubro 2010.

[BEG09] Fevzi Belli, Mubariz Eminov, and Nida Gökçe. Model-based test prioritizing - a comparative soft-computing approach and case studies. In Bärbel Mertsching, Marcus Hund, and Muhammad Zaheer Aziz, editors, *KI*, volume 5803 of *Lecture Notes in Computer Science*, pages 427–434. Springer, 2009.

[Bei90] Boris Beizer. *Software Testing Techniques*. International Thomson Computer Press, 2 edition, June 1990.

[Bei95] Boris Beizer. *Black-Box Testing: Techniques for Functional Testing of Software and Systems*. Verlag John Wiley & Sons, Inc, 1 edition, 1995.

[BSH86] V. R. Basili, R. Selby, and D. Hutchens. Experimentation in Software Engineering. *IEEE Transactions on Software Engineering*, 12(7):733–743, July 1986.

[CANM08] Emanuela G. Cartaxo, Wilkerson L. Andrade, Francisco G. Oliveira Neto, and Patrícia D. L. Machado. LTS-BT: a tool to generate and select functional test

cases for embedded systems. In *SAC '08: Proc. of the 2008 ACM Symposium on Applied Computing*, volume 2, pages 1540–1544, New York, NY, USA, 2008. ACM.

[CJ02] Rick D. Craig and Stefan P. Jaskiel. *Systematic Software Testing*. Artech House, 2002.

[CKMT10] Tsong Yueh Chen, Fei-Ching Kuo, Robert G. Merkel, and T. H. Tse. Adaptive random testing: The art of test case diversity. *Journal of Systems and Software*, 83(1):60–66, 2010.

[CLM04] T. Y. Chen, H. Leung, and I. K. Mak. Adaptive random testing. In *Advances in Computer Science - ASIAN 2004*, volume 3321/2005 of *Lecture Notes in Computer Science*, pages 320–329. Springer Berlin / Heidelberg, 2004.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Nova Yorque, 2ª edição edition, 2001.

[CNM07] Emanuela G. Cartaxo, Francisco G. Oliveira Neto, and Patrícia D. L. Machado. Test case selection using similarity function. In *MOTES 2007*, pages 381 – 386, 2007.

[DJK+99] S. R. Dalal, A. Jain, N. Karunanithi, J. M. Leaton, C. M. Lott, G. C. Patton, and B. M. Horowitz. Model-based testing in practice. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 285–294, New York, NY, USA, 1999. ACM.

[dL09] Lucas Albertins de Lima. Test case prioritization based on data reuse for black-box environments. Master's thesis, Universidade Federal de Pernambuco, 2009.

[dON11] Francisco Gomes de Oliveira Neto. Investigação e avaliação experimental de técnicas de re-teste seletivo para teste de regressão baseado em especificação. Master's thesis, Universidade Federal de Campina Grande, Janeiro 2011.

[EFW01] Ibrahim K. El-Far and James A. Whittaker. Model-based software testing. In *Encyclopedia on Software Engineering*. Wiley-Interscience, 2001.

[EMR00] Sebastian G. Elbaum, Alexey G. Malishevsky, and Gregg Rothermel. Prioritizing test cases for regression testing. In *ISSTA*, pages 102–112, 2000.

[EMR01] Sebastian G. Elbaum, Alexey G. Malishevsky, and Gregg Rothermel. Incorporating varying test costs and fault severities into test case prioritization. In *ICSE*, pages 329–338. IEEE Computer Society, 2001.

[EMR02] Sebastian G. Elbaum, Alexey G. Malishevsky, and Gregg Rothermel. Test case prioritization: A family of empirical studies. *IEEE Transactions in Software Engineering*, Fevereiro 2002.

[ERKM04] Sebastian G. Elbaum, Gregg Rothermel, Satya Kanduri, and Alexey G. Malishevsky. Selecting a cost-effective test case prioritization technique. *Software Quality Journal*, 12(3):185–210, 2004.

[ESR08] Emelie Engström, Mats Skoglund, and Per Runeson. Empirical evaluations of regression test selection techniques: a systematic review. In H. Dieter Rombach, Sebastian G. Elbaum, and Jürgen Münch, editors, *ESEM*, pages 22–31. ACM, 2008.

[Fei06] Dror G. Feitelson. Experimental computer science: The need for a cultural change, 2006.

[Gau03] H.G. Gauch. *Scientific method in practice*. Cambridge University Press, 2003.

[HGS93] Mary Jean Harrold, Rajiv Gupta, and Mary Lou Soffa. A methodology for controlling the size of a test suite. *ACM Trans. Softw. Eng. Methodol.*, 2(3):270–285, 1993.

[Jai91] R. K. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1991.

[JG08] Dennis Jeffrey and Neelam Gupta. Experiments with test case prioritization using relevant slices. *Journal of Systems and Software*, 81(2):196 – 221, 2008. Model-Based Software Testing.

[JH03] James A. Jones and Mary Jean Harrold. Test-suite reduction and prioritization for modified condition/decision coverage. *IEEE Trans. Software Eng.*, 29(3):195–209, 2003.

[JJ02] Claude Jard and Thierry Jéron. Tgv: theory, principles and algorithms. In H. Ehrig, B.J. Krämer, and A. Ertas, editors, *Integrated Design & and Process Technology*, volume IDPT 1. Society for Process & Design Sciences, 2002.

[Jor95] P. C. Jorgensen. *Software Testing - a Craftsman Approach.* CRC Press, 1995.

[JZCT09] Bo Jiang, Zhenyu Zhang, Wing Kwong Chan, and T. H. Tse. Adaptive random test case prioritization. In *ASE*, pages 233–244. IEEE Computer Society, 2009.

[Kan06] Gopal K. Kanji. *100 statistical tests.* Sage, Londres, 3ª edition, 2006.

[Kit04] Barbara Kitchenham. Procedures for performing systematic reviews. Technical report, Keele University, July 2004.

[KK09] Bogdan Korel and George Koutsogiannakis. Experimental comparison of code-based and model-based test prioritization. In *ICSTW '09: Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation Workshops*, pages 77–84, Washington, DC, USA, 2009. IEEE Computer Society.

[KKT08] Bogdan Korel, George Koutsogiannakis, and Luay Ho Tahat. Application of system models in regression test suite prioritization. In *ICSM*, pages 247–256. IEEE, 2008.

[KM09] R. Krishnamoorthi and S. A. Sahaaya Arul Mary. Factor oriented requirement coverage based system test case prioritization of new and regression test cases. *Information & Software Technology*, 51(4):799–808, 2009.

[KSSM09] Debasish Kundu, Monalisa Sarma, Debasis Samanta, and Rajib Mall. System testing for object-oriented systems with test case prioritization. *Softw. Test., Verif. Reliab.*, 19(4):297–333, 2009.

[LBC+10] Sihan Li, Naiwen Bian, Zhenyu Chen, Dongjiang You, and Yuchen He. A simulation study on some search algorithms for regression test case prioritization. *Quality Software, International Conference on*, 0:72–81, 2010.

[MR03] Douglas C. Montgomery and George C. Runger. *Applied Statistics and Probability for Engineers, 4th Edition, and JustAsk! Set.* John Wiley & Sons, 3ª edition, May 2003.

[Mye04] Glenford J. Myers. *The Art of Software Testing, Second Edition.* Wiley, 2 edition, 2004.

[OCM10] João Felipe Silva Ouriques, Emanuela Cartaxo, and Patrícia Duarte Lima Machado. Comparando técnicas de priorização de casos de teste no contexto de teste baseado em modelos. In *Proceedings of the IV Brazilian Workshop on Systematic and Automatic Software Testing (SAST 2010)*, 2010.

[OMG04] OMG. The uml 2.0 superstructure specification. Specification Version 2, Object Management Group, 2004.

[Pre01] Roger S. Pressman. *Software Engineering: A Practitioner's Approach.* McGraw-Hill, 5th edition, 2001.

[QK02] Gerry P. Quinn and Michael J. Keough. *Experimental Design and Data Analysis for Biologists.* Editora da Universidade de Cambridge, Abril 2002.

[QNXZ07] Bo Qu, Changhai Nie, Baowen Xu, and Xiaofang Zhang. Test case prioritization for black box testing. In *COMPSAC (1)*, pages 465–474. IEEE Computer Society, 2007.

[RUCH01] Gregg Rothermel, Roland H. Untch, Chengyun Chu, and Mary Jean Harrold. Prioritizing test cases for regression testing. *IEEE Trans. Software Eng.*, 27(10):929–948, 2001.

[SM07] Heiko Stallbaum and Andreas Metzger. Employing Requirements Metrics for Automating Early Risk Assessment. In *MeRePO7: Proceedings of the Workshop on Measuring Requirements for Project and Product Success*, pages 1–12, 2007.

[SM09] P.G. Sapna and H. Mohanty. Prioritization of scenarios based on uml activity diagrams. In *Computational Intelligence, Communication Systems and Networks, 2009. CICSYN '09. First International Conference on*, pages 271 –276, july 2009.

[SMP08a] Heiko Stallbaum, Andreas Metzger, and Klaus Pohl. An automated technique for risk-based test case generation and prioritization. In *AST '08: Proceedings of the 3rd international workshop on Automation of software test*, pages 67–70, New York, NY, USA, 2008. ACM.

[SMP08b] Heiko Stallbaum, Andreas Metzger, and Klaus Pohl. An automated technique for risk-based test case generation and prioritization. In Hong Zhu, W. Eric Wong, and Fevzi Belli, editors, *AST*, pages 67–70. ACM, 2008.

[Som06] Ian Sommerville. *Software Engineering (8th edition).* Addison Wesley, New York, 8th edition, 2006.

[SWO05] Hema Srikanth, Laurie Williams, and Jason Osborne. System test case prioritization of new and regression test cases. In *ISESE*, pages 64–73. IEEE, 2005.

[TAS06] Paolo Tonella, Paolo Avesani, and Angelo Susi. Using the case-based ranking methodology for test case prioritization. In *ICSM*, pages 123–133. IEEE Computer Society, 2006.

[UL07] Mark Utting and Bruno Legeard. *Practical Model-Based Testing: A Tools Approach.* Morgan Kauffman, first edition, 2007.

[WRH+00] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn Regnell, and Anders Wesslén. *Experimentation in software engineering: an introduction.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[Zho10] Zhi Quan Zhou. Using coverage information to guide test case selection in adaptive random testing. In *Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual*, pages 208 –213, july 2010.