Universidade Federal de Goiás
Instituto de Informática

Gilmar Ferreira Arantes

Uma Estratégia para a Avaliação e Evolução de Teste Funcional de Software

Goiânia
2012

Gilmar Ferreira Arantes

Uma Estratégia para a Avaliação e Evolução de Teste Funcional de Software

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Computação.

Área de concentração: Sistemas de Informação.

Orientador: Prof. Plínio de Sá Leitão Júnior

Goiânia
2012

Dados Internacionais de Catalogação na Publicação (CIP)
GPT/BC/UFG

A662e

Arantes, Gilmar Ferreira.
Uma estratégia para a avaliação e evolução de teste funcional de software [manuscrito] / Gilmar Ferreira Arantes. – 2012.
160 f. : il.

Orientador: Prof. Dr. Plínio de Sá Leitão Júnior.
Dissertação (Mestrado) – Universidade Federal de Goiás, Instituto de Informática, 2012.
Bibliografia.
Inclui lista de figuras e tabelas.
Apêndices.

1. Teste de software – Avaliação. 2. Teste funcional. I. Título.

CDU: 004.415.53-047.43

Gilmar Ferreira Arantes

Uma Estratégia para a Avaliação e Evolução de Teste Funcional de Software

Dissertação defendida no Programa de Pós-Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Ciência da Computação, aprovada em 02 de agosto de 2012, pela Banca Examinadora constituída pelos professores:

Prof. Dr.Plínio de Sá Leitão Júnior
Instituto de Informática - UFG
Presidente da Banca

Prof. Dr. Cássio Leonardo Rodrigues
Instituto de Informática - UFG

Profa. Dra. Silvia Regina Vergilio
DINF - UFPR

Gilmar Ferreira Arantes

Graduou-se em Análise de Sistemas pela Universidade Salgado de Oliveira, Campus Goiânia. Especializou-se em Gestão de Tecnologia da Informação pela Alfa - Faculdades Alves Faria e em Análise e Projeto de Sistemas pela Universidade Federal de Goiás. Durante a pós-graduação foi bolsista da **Fundação de Amparo à Pesquisa do Estado de Goiás - FAPEG**. Atualmente atua como Gestor de Tecnologia da Informação junto à Segplan Secretaria Estadual de Gestão e Planejamento do Estado de Goiás, participando de atividades de desenvolvimento e implantação de sistemas de informação.

Dedico este trabalho:

Primeiramente aos meus pais que na sua humildade e sabedoria souberam ensinar-me a paciência e a persistência necessárias para enfrentar os grandes desafios desta vida.

À minha esposa Kelly Cristina Pereira Volpato Arantes e aos meus filhos Larissa, João Marcos e Maria Eduarda, que pacientemente se privaram da minha presença em vários momentos durante esta pesquisa.

Agradecimentos

Manifesto um agradecimento especial ao meu orientador Plínio de Sá Leitão Júnior, que com inteligência e muita paciência soube conduzir eficientemente as atividades de orientação possibilitando o alcance dos objetivos definidos. Soube, acima de tudo motivar-me em relação ao objetivo final deste trabalho.

Agradeço ao professor e colega de trabalho Willian Divino Ferreira (Mestre Willian) que me deu apoio para ingressar neste programa de pós-graduação.

Agradeço à professora Maria Suely de Aguiar, que acreditou em mim, recomendando-me a este programa de pós-graduação.

Agradeço ainda os alunos deste programa de pós-graduação com quem tive a oportunidade de estudar por algum tempo, dentre os quais, Roneesley Moura Teles, Max Gontijo de Oliveira, Francisco Calaça Xavier, Leonardo Teixeira Queiroz e Adriana Rocha Vidal, pelo suporte que direta ou indiretamente deram-me durante este período.

Agradeço aos colegas de trabalho que se desdobraram para suprir a minha ausência durante o período, principalmente ao José Robério Neves Garcez Rocha e ao Telmo Queiroz Silva.

Agradeço ainda os servidores da Secretaria de Gestão e Planejamento do Estado de Goias (SEGPLAN) que participaram do deferimento do processo em que quatro horas de minhas atividades nesta Secretaria foram liberadas para a destinação exclusiva à conclusão das atividades inerentes a esta pesquisa, nos últimos quatro meses deste ano de 2012.

Agradeço à Fundação de Amparo à Pesquisa do Estado de Goiás (FAPEG), pelo suporte financeiro a esta pesquisa.

Coragem, coragem, se o que você quer é aquilo que pensa e faz. Coragem, coragem, eu sei que você pode mais...

Raul Seixas,
Por quem os sinos dobram.

Resumo

Arantes, Gilmar Ferreira. **Uma Estratégia para a Avaliação e Evolução de Teste Funcional de Software**. Goiânia, 2012. 166p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

Teste de Software faz parte das atividades de garantia da qualidade do software. Destina-se a revelar a presença de defeitos, que podem ser inseridos em vários estágios do desenvolvimento do software. Várias técnicas são usadas na atividade de teste, com destaque para as funcionais, que derivam requisitos de teste a partir da especificação do software. A pesquisa enfrenta o problema de como evoluir as estratégias de testes funcionais reduzindo o custo, em relação à quantidade de casos de teste necessários, sem comprometer o número de defeitos revelados. Uma revisão sistemática foi planejada e executada, com base em questões formuladas de modo a responder ao problema da pesquisa. Esta revisão apoiou a definição de um novo critério de teste funcional, o Teste Funcional Sistemático com Aplicação da Tabela de Decisão (TFS-TD), que é uma extensão do Teste Funcional Sistemático (TFS) e que prevê a aplicação conjunta dos critérios: Particionamento em Classes de Equivalência, Análise do Valor Limite e Tabela decisão. O TFS-TD define uma estratégia baseada em um conjunto de diretrizes e possui um processo para aplicar esta estratégia de forma sistemática. Três estudos empíricos foram aplicados com resultados promissores em relação ao TFS: todos eles reduzem, pelo menos, pela metade o conjunto adequado, sem impacto na quantidade de defeitos revelados.

Palavras–chave

Teste de Software, teste funcional, técnicas de teste, critérios de teste.

Abstract

Arantes, Gilmar Ferreira. **A Strategy for the Evaluation and Evolution of Functional Software Testing**. Goiânia, 2012. 165p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

Software Testing is part of software quality assurance activities. It aims to uncover the presence of defects, that can be inserted in various stages of software development. Several techniques are used in the testing activity, highlighting the functional ones, which derive test requirements from the software specification. The research faces the problem of how to evolve the functional testing strategies with low costs, relative to the amount of test cases needed, without compromising the number of uncovered defects. A systematic review was planned and executed, based on formulated questions so as to answer the research problem. Such review supported the definition of a new criterion for functional testing, the Systematic Functional Test with Decision Table Application (TFS-DT), which is an extension of Systematic Software Testing (TFS) and provides joint application of criteria: Partitioning Equivalence Classes, Boundary Value Analysis and Decision Table. The TFS-DT defines a strategy based on a set of requirements and has a process in order to apply the strategy in a systematic manner. Three empirical studies were applied with promising results compared to TFS: all of them reduces at least half the adequated set without impact on the number of uncovered defects.

Keywords

Software testing, funcional testing, test techniques, test criteria.

Conteúdo

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Software está inserido em muitas atividades do nosso dia-a-dia: educação, saúde, entretenimento, negócios, etc. O funcionamento correto do software é uma necessidade real, caso contrário pode haver desde perdas de menor escala até danos mais importantes, tais como, prejuízo financeiro e risco de vida. A produção de software possui a atuação humana e, portanto, é influenciada pelas imperfeições dessa atuação. A Engenharia de Software introduz várias atividades de garantia de qualidade no processo de desenvolvimento de software para minimizar esse problema.

Qualidade é uma característica importante na produção de software, sendo requerida no processo e no produto relacionado. Qualidade consiste de um conjunto de requisitos e de um produto ou serviço que esteja em conformidade com estes requisitos e, por esta razão, atenda completamente às necessidades dos clientes (ISO/IEC, 2001). De acordo com Pressman (2005), qualidade de software é a conformidade a: (i) requisitos funcionais e não funcionais explicitamente declarados; (ii) padrões de desenvolvimento que tenham sido claramente documentados; e (iii) características implicitamente esperadas de todo software a ser desenvolvido.

Teste de Software é uma das áreas da Engenharia de Software em que se busca a garantia da qualidade do software, contribuindo continuamente para a melhoria dos processos e produtos. Deve ser aplicado em todas as etapas do ciclo de vida do software, da concepção à implantação, e em suas manutenções posteriores.

Teste de Software está inserido no contexto dos processos denominados Verificação e Validação. De acordo com Ammann e Offutt (2008), **Verificação** é o processo de determinar se os produtos (artefatos) de uma dada fase do processo de desenvolvimento atendem aos requisitos estabelecidos na fase anterior e **Validação** é o processo de avaliação do software no final do seu desenvolvimento para garantir a conformidade com a sua finalidade, isto é, o software estar de acordo com o desejo do cliente.

O principal objetivo do teste é revelar a presença de defeitos no software, para que possam ser corrigidos antes que causem algum dano, o que aumenta a confiabilidade do software. Idealmente, a atividade de teste deve ser conduzida de maneira sistemática, aplicando-se técnicas que balanceiem a redução de custo e o aumento das chances para revelar a presença de defeitos, caso existam. Tais técnicas definem elementos requeridos, que representam requisitos que devem ser cobertos durante o teste. As técnicas mais populares para o teste de software são:

- **Técnica Estrutural**, também conhecida como *teste caixa-branca*, em que os elementos requeridos são derivados da estrutura do software.
- **Técnica Funcional**, também conhecida como *teste caixa-preta*, em que os elementos requeridos são derivados da especificação funcional do software.

Cada uma dessas técnicas possui um conjunto de *critérios de teste*, que podem ser usados na geração, seleção e avaliação de um conjunto de casos de testes. Alguns exemplos são: (i) o critério estrutural *Todos-comandos*, que requer que cada comando do programa seja executado pelo menos uma vez durante o teste; (ii) o critério funcional *Análise do valor limite*, que divide o domínio de entrada do software em partições e requer que os limites de cada partição sejam testados pelo menos uma vez durante o teste.

1.1 Motivação e Objetivos

A motivação maior deste trabalho é contribuir com as pesquisas na área de teste de software desenvolvidas pelo Instituto de Informática da Universidade Federal de Goiás, particularmente em relação ao emprego do teste funcional de software. A decisão pelo teste funcional é justificada pela indisponibilidade da implementação do software em muitos casos. Especificamente, o autor possui interesse em agregar valor ao teste de Programas Aplicativos Fiscais, os quais não possuem código fonte acessível.

O problema atribuído à pesquisa é **como aplicar o teste funcional visando a sua redução de custo, sem perdas relevantes com respeito à qualidade do teste em relação à quantidade de defeitos revelados**. Interroga-se a forma como os vários critérios e estratégias funcionais têm sido empregados e, especificamente, questiona-se como reduzir o custo da aplicação do Teste Funcional Sistemático (TFS), mantendo-se a qualidade do teste segundo a análise de mutantes. A solução desse problema perpassa por responder outras questões, tais como: (i)

quais os critérios e técnicas funcionais mais utilizados?, (ii) como tais critérios e técnicas são avaliados?, (iii) quais os cenários de aplicação desses critérios e técnicas? e (iv) quais critérios e técnicas têm sido aplicados em conjunto?

O objetivo principal é propor uma solução ao problema da redução do custo sem perda de qualidade, atribuído à pesquisa. Dois objetivos secundários são identificados: (i) **avaliar um conjunto de dados de teste**, que seja adequado a um ou mais critérios ou técnica de teste funcional, buscando reduzir a sua cardinalidade, sem prejuízo com respeito à detecção de defeitos; (ii) **evoluir um conjunto de dados de teste**, visando a identificar um subconjunto com medidas similares de qualidade. O atendimento a estes objetivos secundários contribuem com a solução do problema atribuído à pesquisa, tendo em vista que o primeiro diz respeito ao custo e o segundo à qualidade do conjunto de teste em análise.

1.2 Metodologia

Alguns aspectos metodológicos do trabalho são:

- estudar os critérios e técnicas funcionais propostos na literatura;
- planejar e conduzir uma revisão sistemática focada nas questões de pesquisa;
- utilizar as respostas obtidas na revisão sistemática para propor uma solução ao problema atribuído à pesquisa;
- avaliar a solução proposta, comparando-a, através de estudos empíricos, com resultados da literatura, em relação ao custo e à qualidade, onde custo diz respeito à quantidade de casos de teste utilizada e qualidade diz respeito à quantidade de defeitos detectados. Um indicador de qualidade da solução proposta é que a redução da quantidade de casos de teste deve ser sempre maior que a potencial quantidade de defeitos não revelados, dado que tamanho da redução não é conhecido antes da aplicação da solução proposta;
- empregar a solução proposta à especificação de teste de programa aplicativo fiscal, conforme interesse particular do autor em agregar valor ao teste de Programas Aplicativos Fiscais;
- analisar resultados;
- propor desdobramentos futuros à pesquisa.

1.3 Organização da Dissertação

De acordo com a motivação e metodologia definidas e visando a alcançar os objetivos, esta dissertação foi organizada da seguinte forma:

- o Capítulo 2 apresenta um estudo detalhado de nove critérios e técnicas de teste funcional, além do *Teste Aleatório*, onde são apresentadas definições, aplicabilidade, pontos fortes e fracos e exemplos de utilização;
- o Capítulo 3 apresenta o protocolo do planejamento de uma revisão sistemática sobre teste funcional, onde se busca levantar evidências sobre questões inerentes à sua aplicação;
- o Capítulo 4 descreve as análises efetuadas sobre os estudos primários selecionados pela revisão sistemática, as respostas obtidas para as questões de pesquisa e uma análise qualitativa dos estudos analisados relativamente à força das evidências presentes nestes estudos. Questões que constituem ameaças à validade desta revisão sistemática também são identificadas neste capítulo;
- o Capítulo 5 apresenta o Teste Funcional Sistemático com Aplicação de Tabela de Decisão (TFS-TD), um novo critério de teste funcional, cuja concepção foi motivada pelos resultados obtidos pela revisão sistemática. Três estudos empíricos são apresentados para a validação da efetividade deste novo critério;
- o Capítulo 6 apresenta as considerações finais, destacando as contribuições advindas deste trabalho à atividade de teste e os possíveis desdobramentos em novos trabalhos futuros.
- o Apêndice A apresenta um glossário com os termos utilizados ao longo do texto desta dissertação;
- o Apêndice B apresenta as informações relevantes extraídas de cada um dos estudos primários analisados, que serviram de suporte para as respostas às questões de pesquisa da revisão sistemática.
- o Apêndice C apresenta as etapas da condução da revisão sistemática.

CAPÍTULO 2

Técnica de Teste Funcional

Uma breve introdução sobre conceitos inerentes ao teste de software, tais como técnicas e critérios de teste, foi apresentada no capítulo anterior. Este capítulo abordará a técnica de teste funcional, que é baseada nos requisitos do software, não requerendo conhecimento da estrutura interna do software, tratando o objeto em teste como um mecanismo que recebe uma entrada x e produz uma saída y.

A técnica funcional possui um conjunto de critérios que são empregados durante a geração e/ou seleção de casos de teste, que ocorre a partir da análise da especificação de requisitos. O emprego do teste funcional possui pontos fortes, destacando-se:

- por ser baseado na especificação e, portanto independente da implementação, o conjunto de testes adequado às funcionalidades permanece inalterado, mesmo que haja alteração da implementação;
- os critérios de teste funcional derivam subconjuntos representativos de todo o domínio de entrada das variáveis em teste;
- a derivação dos casos de teste pode acontecer paralelamente à implementação, reduzindo o tempo do projeto, conforme destaca Jorgensen (2002);
- alguns dos critérios componentes do teste funcional (Grafo de Causa e Efeito, Tabela de Decisão, Teste Baseado em Transição de Estados e Teste Baseado em Casos de Uso) são ótimas ferramentas para auxiliar na especificação dos requisitos do software.

Nas próximas seções alguns dos principais critérios e técnicas de teste funcional e o Teste Aleatório serão analisados detalhadamente. Serão abordados: Particionamento em Classe de Equivalência, Análise do Valor Limite, Teste Funcional Sistemático (TFS), Teste Funcional Sistemático Estendido (TFSE), Tabela de Decisão, Grafo de Causa e Efeito, Teste em Pares, Teste Baseado em Transição de Estados, Teste Baseado em Casos de Uso e Teste Aleatório. Definições, aplicabilidade,

potenciais vantagens e desvantagens inerentes à utilização são alguns dos aspectos explorados.

2.1 Particionamento em Classes de Equivalência

Particionamento em Classes de Equivalência é um critério de teste funcional que faz parte do grupo de critérios e técnicas que definem uma estratégia de teste denominada "teste por particionamento de domínio". Este critério utiliza os conceitos de partição e de equivalência de acordo com a definição matemática atribuída aos mesmos:

- em Marques (2011), **Partição** é definida da seguinte forma: seja A um conjunto não vazio. Define-se como partição de A, e representa-se por *part(A)*, qualquer subconjunto do conjunto das partes de A (representado simbolicamente por *P(A)*), que satisfaz simultaneamente, às seguintes condições:
  1. nenhum dos elementos de *part(A)* é o conjunto vazio;
  2. a interseção de quaisquer dois elementos de *part(A)* é o conjunto vazio;
  3. a união de todos os elementos de *part(A)* é igual ao conjunto A.
- conforme Sodré e Neto (2004), uma relação de *equivalência* sobre o conjunto A é uma relação R que possui as propriedades: *reflexiva*, *simétrica* e *transitiva*. Exemplo: Seja R a relação definida no conjunto dos números reais por $(x,y) \in R$ se, e somente se, $|x|=|y|$. Para todo número real x temos que xRx, pois $|x|=|x|$, garantindo que R é reflexiva. Se xRy então $|x|=|y|$ e segue que yRx pois $|y|=|x|$, provando que R é uma relação simétrica. Se aRb e bRc, então $|a|=|b|$ e $|b|=|c|$, então $|a|=|c|$, ou seja aRc, logo R é transitiva. Concluímos que R é uma relação de equivalência.

De acordo com a definição de partição, a execução do teste por particionamento requer um conjunto de partições disjuntas, para a derivação de casos de teste. No entanto, conforme observa Reid (1997), a maioria das técnicas de teste falha em relação a este critério de homogeneidade, e produzem partições com intercessões entre si. Desta forma, é importante estabelecer a seguinte distinção:

1. o teste baseado em partições disjuntas é o teste por particionamento e
2. o teste baseado em partições com intercessões entre si é o teste de subdomínio.

Este texto descreve o teste por particionamento e aborda os conceitos de partição e classe de equivalência como sinônimos, sendo utilizados de maneira alternada nos capítulos e seções em que são referenciados, denotando o mesmo significado.

A aplicação deste critério de teste consiste em dividir o conjunto das entradas de um problema em subconjuntos disjuntos, de forma que qualquer elemento tenha a capacidade de ser representativo de todo o subconjunto. Desta forma, acredita-se que o resultado de um teste com qualquer destes elementos seja equivalente a todo o subconjunto. Sendo assim, se um elemento revela a presença de um defeito, acredita-se que todos os demais também a revelarão. Por outro lado, se um elemento não é capaz de revelar tal presença, então acredita-se também que nenhum outro elemento terá esta capacidade. A Figura 2.1, obtida de Guimarães (2011), apresenta um exemplo de particionamento por equivalência, onde o domínio válido de uma variável está entre 4 e 10.

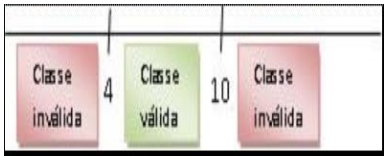

Figura 2.1: *Particionamento em classes de equivalência*

A motivação para a utilização deste critério de teste funcional é resultante da necessidade de se definir uma quantidade de casos de testes que seja representativa em termos de economia e abrangente em termos de cobertura das funcionalidades testadas. Myers e Sandler (2004) cita duas motivações principais para a utilização deste critério de teste funcional:

- Dado que a realização do teste exaustivo (testar todas as entradas e saídas do domínio) não é factível (KANER, 1997), então é necessário selecionar um pequeno subconjunto que tenha a capacidade de representar todas as entradas possíveis, se possível um subconjunto que tenha alta probabilidade de revelar a maioria dos potenciais defeitos existentes. Ainda, segundo Myers e Sandler (2004) este subconjunto selecionado para os testes deve ter as seguintes propriedades: (a) possuir uma quantidade mínima possível de casos de teste e (b) a cobertura do teste deve ser tão elevada quanto possível.
- evitar redundância, dada a disjunção das partiçoes definidas.

O processo de utilização é composto por apenas dois passos: (i) identificação das classes de equivalência e (ii) geração dos casos de teste. A identificação das classes de equivalência é baseada em alguma condição externa, constante da especificação

do requisito em teste. Definem-se classes de equivalências válidas, contendo aqueles elementos que atendem à condição externa e inválidas contendo aqueles elementos que não atendem à condição. Segundo Myers e Sandler (2004) dada esta condição externa, a identificação das classes de equivalência é um processo heurístico. Em adição, apresenta quatro dicas para facilitar a identificação destas classes:

1. se a condição de entrada especifica uma **faixa de valores**, identificam-se três classes de equivalência, sendo uma válida (valores dentro da faixa definida) e duas inválidas (valores abaixo e acima da faixa definida);
2. se a condição de entrada especifica uma **quantidade determinada de valor**, identifica-se uma classe de equivalência válida (quantidade exata) e duas inválidas (nenhum valor e quantidade acima da estabelecida);
3. se a condição de entrada especifica um **conjunto de valores de entradas** e existe razão para acreditar que o programa processa cada uma diferentemente (tipos de veículos: "Caminhão, ônibus, táxi, carro de passeio, motocicleta"). Identifica-se uma classe de equivalência válida que represente cada um dos elementos e uma classe de equivalência inválida (*"Trailler"*), por exemplo;
4. se a condição de entrada especifica uma situação de "**deve ser**", como por exemplo, o primeiro caractere de uma determinada variável deve ser uma letra. Identifica-se uma classe de equivalência válida (é uma letra) e uma inválida (não é uma letra).

Para reforçar o entendimento a respeito deste critério de teste funcional, apresenta-se um exemplo do seu emprego na geração de casos de teste para o problema do triângulo, conforme descrito em Myers e Sandler (2004). Este problema tem a seguinte especificação:

- O programa recebe como entrada três inteiros (a, b, c), que representam o tamanho dos lados de um triângulo;
- qualquer dos lados deve ser menor que a soma dos outros dois;
- são quatro as possíveis saídas produzidas pelo programa, com base nos valores de entrada:
    1. equilátero - quando todos os lados do triângulo são iguais;
    2. isósceles - quando dois lados do triângulo são iguais;
    3. escaleno - quando nenhum dos lados do triângulo são iguais;
    4. não é um triângulo - quando qualquer dos lados não for menor que a soma dos outros dois.

Podemos identificar oito classes de equivalência, descritas na Tabela 2.1, adaptada de Jorgensen (2001), para testar este problema, onde a faixa de valores válidos para os lados do triângulo foi definida entre 1 e 200. A primeira Coluna desta tabela apresenta o identificador da classe de equivalência, onde I = inválida e V = válida; o número entre parênteses representa uma numeração sequencial atribuída às classes. A segunda Coluna identifica os valores possíveis para cada classe e a terceira Coluna descreve a classificação (válida/inválida) de cada uma das classes.

Tabela 2.1: *Classes de equivalência definidas para o problema do triângulo*

Classe de Equivalência	Conteúdo	Classificação
V(1)	{a,b,c : 1..200 | a = b = c}	Válida
V(2)	{a,b,c : 1..200 | a = b, a ≠ c}	Válida
V(3)	{a,b,c : 1..200 | a = c, a ≠ b}	Válida
V(4)	{a,b,c : 1..200 | b = c, a ≠ b}	Válida
V(5)	{a,b,c : 1..200 | a ≠ b, a ≠ c, b ≠ c}	Válida
I(6)	{a,b,c : 1..200 | a ≥ b+c}	Inválida
I(7)	{a,b,c : 1..200 | b ≥ a+c}	Inválida
I(8)	{a,b,c : 1..200 | c ≥ a+b}	Inválida

Com estas classes de equivalência definidas é possível gerar o conjunto de casos de testes, constantes da Tabela 2.2 adaptada de Jorgensen (2001).

Tabela 2.2: *Exemplo de casos de testes para o problema do triângulo*

Caso de Teste	a	b	c	Resultado Esperado
CT01	5	5	5	Equilátero
CT02	2	3	3	Isósceles
CT03	3	4	5	Escaleno
CT04	4	1	2	Não é um triângulo
CT05	-1	5	5	“a” está fora da faixa
CT06	5	-1	5	“b” está fora da faixa
CT07	5	5	-1	“c” está fora da faixa
CT08	201	5	5	“a” está fora da faixa
CT09	5	201	5	“b” está fora da faixa
CT10	5	5	201	“c” está fora da faixa

Alguns pontos fortes deste critério são:

- representa um teste completo do domínio de entrada - o conjunto completo das entradas da variável em teste é representado pela união dos subconjuntos (partições) identificados;
- evita testes redundantes devido à disjunção dos subconjuntos;
- reduz o número de casos de testes - seleciona apenas um caso de teste para cada partição, como regra geral;

- é aplicável em vários níveis de teste, conforme Copeland (2003), dentre os quais: testes de unidade, teste de integração, teste de sistema e teste de aceitação.

Em relação aos pontos fracos, podem ser relacionados os seguintes:

- sendo destinado ao teste de variáveis individuais, este critério não é adequado ao teste variáveis que não são independentes umas das outras, variáveis cujo valor de entrada dependa do valor resultante do processamenteo de outras variáveis;
- ausência de um processo formal para a definição das classes de equivalência, deixando esta tarefa por conta de experiência do testador, o que pode redundar em erros.

2.2 Análise do Valor Limite

Análise do Valor Limite é um critério de teste funcional complementar ao critério Particionamento em Classes de Equivalência, diferenciando-se na forma de derivar casos de testes para cada classe de equivalência. O foco é o teste dos limites de cada partição, baseando-se na suposição de que muitos erros ocorrem nestes limites ou nas suas imediações (imediatamente acima e imediatamente abaixo).

Segundo Myers e Sandler (2004), ao invés de selecionar qualquer elemento da classe de equivalência como representativo para toda a classe, *Análise do Valor Limite* exige que um ou mais elementos sejam selecionados, tal que cada limite da classe de equivalência seja submetido ao teste.

A geração de casos de testes utilizando o critério de teste funcional Análise do Valor Limite, de acordo com Jorgensen (2002) leva em consideração dois fatores: a criticalidade dos requisitos e o modelo de tolerância à falha.

Para o teste de variáveis individuais em sistemas onde a tolerância a falhas não é crítica, utiliza-se o procedimento básico para a derivação de casos de testes, alcançando-se o número de cinco casos de testes para cada classe de equivalência. Neste cenário, para uma quantidade n de variáveis, a quantidade de casos de testes é dada pela fórmula: $4n + 1$, conforme descrito abaixo e ilustrado na Figura 2.2, obtida de Jorgensen (2002).

1. um caso de teste para o valor correspondente ao limite inferior da partição, representando por: (min);

2. um caso de teste para o valor imediatamente superior ao limite inferior da partição, representando por: (*min+*);
3. um caso de teste para testar um valor nominal, isto é, um valor que encontra-se nas imediações do centro da partição, representando por: (*nom*);
4. um caso de teste para testar o valor imediatamente abaixo do limite superior da partição, representando por: (*max-*) e
5. um caso de teste para testar o valor correspondente ao limite superior da partição, representado por (*max*).

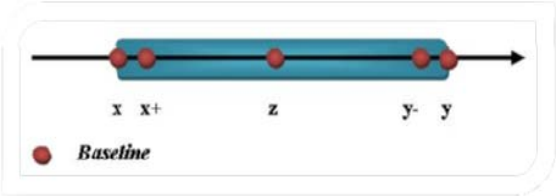

Figura 2.2: *Exemplo de utilização do critério Análise do Valor Limite (Procedimento Padrão)*

Para o teste de sistemas onde o tratamento de erros é crítico, como por exemplo, sistemas de tempo real, embarcados em aeronaves, reatores nucleares, etc., utiliza-se o procedimento amplificado ou robusto para a derivação dos casos de testes, ampliando a quantidade de cinco para sete. Estes dois novos casos de testes estão imediatamente abaixo *min-* e imediatamente acima *max+* dos limites de cada partição, conforme ilustra a Figura 2.3, obtida de Jorgensen (2002). Assim, a quantidade de casos de testes derivados para cada partição definida para testar sistemas desta natureza é dada pela seguinte fórmula: *6n + 1*.

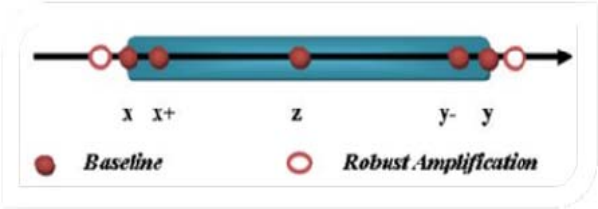

Figura 2.3: *Exemplo de utilização do critério Análise do Valor Limite (Procedimento Amplificado ou Robusto)*

.

O processo para a utilização do critério de teste *Análise do Valor Limite* é composto pelos mesmos passos definidos para a utilização do critério *Particionamento em Classes de Equivalência*, acrescentando um terceiro passo que é a identificação dos limites (inferior e superior) de cada classe de equivalência.

Para o reforço do entendimento, um exemplo da utilização do critério *Análise do Valor Limite* na geração de casos de teste para o problema do triangulo é apresentado a seguir. O conjunto completo de casos de teste para as três variáveis é dado por *6n +1*. Dado que $n = 3$, a quantidade total de casos de teste para este teste é *19*. A Tabela 2.3 apresenta os casos de teste drivados para o problema do triângulo, com a utilização do critério *Análise do Valor Limite*.

Tabela 2.3: *Exemplo de casos de testes derivados pelo critério de teste Análise do valor limite, para o problema do triângulo*

Caso de Teste	a	b	c	Resultado Esperado	Caso de Teste	a	b	c	Resultado Esperado
CT01	100	100	0	"c" está fora da faixa	CT11	100	199	100	Triângulo isósceles
CT02	100	100	1	Triângulo isósceles	CT12	100	200	100	Não é um triângulo
CT03	100	100	2	Triângulo isósceles	CT13	100	201	100	"b" está fora da faixa
CT04	100	100	100	Triângulo equilátero	CT14	0	100	100	"a" está fora da faixa
CT05	100	100	199	Triângulo isósceles	CT15	1	100	100	Triângulo isósceles
CT06	100	100	200	Não é um triangulo	CT16	2	100	100	Triângulo isósceles
CT07	100	100	201	"c" está fora da faixa	CT17	199	100	100	Triângulo isósceles
CT08	100	0	100	"b" está fora da faixa	CT18	200	100	100	Não é um triângulo
CT09	100	1	100	Triângulo isósceles	CT19	201	100	100	"a" está fora da faixa
CT10	100	2	100	Triângulo isósceles					

As vantagens de se utilizar este critério de teste são as mesmas identificadas para o critério *Particionamento em Classes de Equivalência*. Em Myers e Sandler (2004) observa-se que, se utilizado corretamente, este é um dos mais poderosos critérios de teste, pois há a correta identificação das classes de equivalência e seus respectivos limites (inferior e superior).

Algumas limitações da aplicação do critério são:

1. teste de variáveis booleanas, pois não possuem limites (sim/não);
2. teste de variáveis que não são independentes umas das outras;
3. dependendo da abordagem utilizada, pode apresentar alto grau de redundância na derivação dos casos de testes.

2.3 Teste Funcional Sistemático

O Teste Funcional Sistemático (TFS), conforme descrito na Seção B.8, na página 133, é um critério de teste funcional que define um conjunto de diretrizes para derivar dados de teste, inspirando-se na utilização conjunta dos critérios funcionais *Particionamento em Classes de Equivalência* e *Análise do Valor Limite*.

As diretrizes do TFS sugerem a derivação dois dados de teste para cada classe de equivalência identificada e dados de teste para explorar as fronteiras de cada uma

destas classes, bem como os arredores das mesmas (valores imediatamente inferiores e superiores). Este processo pode gerar uma quantidade elevada de dados de testes, podendo haver redundância entre eles. Considere o exemplo do teste de uma variável numérica x cujo domínio são valores inteiros variando de 1 a 1000. As seguintes classes de equivalência são identificadas para o teste desta variável:

I(1): {x, $\forall$x < 1} (classe inválida);
V(2): {x, $\forall$x | $1 \leq$ x ≤ 1000} (classe válida);
I(3): {x, $\forall$x > 1000} (classe inválida);
I(4): {x, $\forall$x não inteiro} (classe inválida contendo valores de tipos de dados diferentes de inteiro).

Seguindo as diretrizes do TFS, o seguintes dados de teste poderão ser derivados para exercitar cada uma destas classes de equivalência:

DT_1: -1 (exercita I(1));
DT_2: 0 (exercita I(1));
DT_3: 1 (exercita V(2));
DT_4: x_1 | $1 < x_1 < 1000$ (exercita V(2));
DT_5: x_2 | $1 < x_2 < 1000$ e $x_2 \neq x_1$ (exercita V(2));
DT_6: 1000 (exercita V(2));
DT_7: 1001 (exercita I(3));
DT_8: 1002 (exercita I(3));
DT_9: a (exercita I(4));
DT_{10}: 2.0 (exercita I(4)).

Neste pequeno exemplo, 4 classes de equivalência e 10 dados de teste foram definidos para uma única variável. Em um outro exemplo, essa quantidade pode ser superior, pois o TFS sugere a derivação de pelo menos dois dados de teste para cada classe de equivalência. Se considerarmos situações onde existam interações entre variáveis, a quantidade de dados de teste será ainda maior. Uma questão pertinente é: o TFS pode ser empregado de forma mais econômica, conservando-se a qualidade do conjunto adequado?

O TFS é baseado no conceito de equivalência forte. Por exemplo, considerando duas variáveis de entrada, os dados de teste devem derivar valores que exercitem a combinação de classes válidas-inválidas e inválidas-inválidas, elevando assim a quantidade de dados de teste. A utilização desta abordagem é recomendada para o teste de sistemas críticos, onde a ocorrência de defeitos pode gerar sérias

consequências seja em termos de segurança, finanças e até mesmo representar perigo a vidas humanas.

2.4 Teste Funcional Sistemático Estendido

Teste Funcional Sistemático Estendido, doravante referenciado como TFSE, é um critério de teste funcional proposto por Vidal (2011). Consiste de uma extensão do TFS para para contemplar os tipos de dados Data e Hora. Formaliza as diretrizes apresentadas por Linkman et al. (2003) para a geração de casos de testes, através de um conjunto de algoritmos, descritos abaixo, tendo definido um algoritmo para cada tipo de dado específico, facilitando tanto o entendimento quanto a aplicação do TFS e do próprio TFSE.

A validadação do TFSE é efetuada através de dois estudos de caso, que contemplam a geração de casos de testes para dois sistemas: um sistema *Web* voltado para apoiar a de Gestão Estratégica Simeon (2010) e outro para a geração de casos de teste para alguns requisitos do roteiro de testes do PAF-ECF Confaz (2010). Em ambos os estudos de caso foram destacados a maior potencialidade para a detecção de defeitos a partir da aplicação do critério proposto.

O conjunto de algoritmos constituintes do TFSE, juntamente com a identificação do tipo de dado contemplado por cada um deles, estão listados abaixo:

Algoritmo 4.1: Tipo de dado numérico.
Algoritmo 4.2: Tipo de dado booleano.
Algoritmo 4.3: Quantidade de elementos de entrada e saída do software.
Algoritmo 4.4: Tipo de dado Matriz.
Algoritmo 4.5: Tipo de dado texto ou string.
Algoritmo 4.6: Tipo Data.
Algoritmo 4.7: Tipo Hora.
Algoritmo 4.8: Tipo de dado estruturado heterogêneo.
Algoritmo 4.9: Todos os tipos de dados.

Uma descrição mais minuciosa do TFSE encontra-se na Seção B.27, na Página 154 do Apêndice B. Exemplos da geração de casos de teste empregando o TFSE podem ser observados na Seção 5.3, na Página 99.

2.5 Tabela de Decisão

Os critérios de teste funcional analisados nas seções anteriores focam atenção no teste de variáveis individuais, independentes e baseiam-se no particionamento do domínio do conjunto dos possíveis valores de entrada para estas variáveis. No entanto, existem situações em que as variáveis não são independentes. Há funcionalidades que são executadas tendo como parâmetros de entrada o resultado da combinação de valores de outras variáveis. Esta combinação de valores é expressa na forma de relacionamentos condicionais entre estas variáveis. O critério de teste funcional descrito nesta seção, *Tabela de Decisão*, é uma ferramenta poderosa para testar funcionalidades que possuem variáveis com estas características.
Uma tabela de decisão é composta dos seguintes elementos:

- **Regras** - são derivadas das possíveis combinações das condições;
- **Condições** - que devem ser atendidas para a execução de alguma ação;
- **Ações** - que devem ser executadas em virtude das possíveis combinações das condições.

A quantidade de regras constantes da tabela de decisão é dependente da quantidade de condições que devem ser avaliadas. Como cada condição deve ter pelo menos duas avalições (uma como verdadeiro e outra como falso) e estas avaliações são combinadas entre si, o total de regras é 2^n, onde n representa o número de condições.

A Tabela 2.4, conforme modelo adaptado de Copeland (2003), apresenta um exemplo da forma geral de uma tabela de decisão, onde é possível observar a distribuição dos elementos (regras, condições e ações). Condições são descritas na parte superior da primeira coluna. A parte inferior desta primeira coluna é destinada à descrição das ações. As regras constam da primeira linha, a partir da segunda coluna. As demais células são destinadas aos valores das entradas, que podem ser *verdadeiro ou falso*, sendo representado por “V ou F”, “0 ou 1”, “S ou N”. Estes valores de entrada identificam o atendimento ou não de uma condição e a respectiva ação a ser tomada. Na Figura 2.4 as regras são representadas por R_i, com $i = 1 ...$ 8. O atendimento às condições está representado por “V”, o não atendimento por “F”. Um “X” identifica a ação que deve ser executada como resultado da avaliação da regra.

Tabela de decisão consiste de uma representação tabular das possíveis combinações de condições lógicas que devem ser avaliadas para a tomada de determinadas decisões que são expressas na forma de ações, executadas pelo software. Para Copeland (2003) Tabela de Decisão é mais que uma técnica de teste, na verdade é uma

Tabela 2.4: *Exemplo de tabela de decisão*

condições	R1	R2	R3	R4	R5	R6	R7	R8
condição 1	V	V	V	V	F	F	F	F
condição 2	V	V	F	F	V	V	F	F
condição 3	V	F	V	F	V	F	V	F
ações								
ação 1	X							
ação 2					X			
ação 3								X

técnica de projeto, pois é uma ferramenta valiosa para a avaliação das combinações sobre os valores de entrada, para capturar certos tipos de requisitos do sistema e documentar sua estrutura interna. É usada para registrar regras de negócio complexas que o sistema deve implementar e servem ainda como um guia para derivar casos de testes.

O processo para a utilização da Tabela de Decisão como critério de teste é composto pelos seguintes passos:

1. análise e divisão da especificação de requisitos em unidades lógicas;
2. identificação das condições de entrada (causas) e as ações que o software deve executar em resposta a estas condições (efeitos);
3. desenvolvimento de um grafo de causa e efeito, ligando as causas aos seus respectivos efeitos;
4. transformação do grafo de causa e efeito em uma tabela de decisão;
5. conversão das regras da tabela de decisão em casos de testes. Cada coluna da tabela representa um caso de teste, de forma que o número de casos de testes é igual ao número de regras da tabela.

A principal vantagem obtida com a utilização da Tabela de Decisão é o fato de ela constituir-se numa poderosa ferramenta para auxiliar no levantamento de requisitos, sobretudo naqueles cenários em que existem relacionamentos lógicos entre variáveis. O uso de Tabela de Decisão para auxiliar na derivação casos de testes para estas variáveis, auxilia também na identificação da potencial ausência de algum requisito.

Tabela de Decisão também possui as suas limitações, dentre as quais pode-se destacar que: uma tabela de decisão pode conter uma combinação de condições que não existe na realidade. Por exemplo, suponha que se esteja modelando o cálculo de desconto concedido por uma seguradora de veículos, cujo valor depende da idade e do estado civil do segurado. Numa tabela para este cenário, existirá alguma regra em que a pessoa poderá ter mais de um estado civil, e o pior ainda, poderá ter todos os possíveis (casado, solteiro, desquitado, viúvo). Este tipo de limitação deve ser avaliada com a experiência do testador e com o conhecimento das regras de negócio.

Um exemplo da utilização da Tabela de decisão, juntamente com o critério Grafo de Causa e Efeito, na derivação de casos de teste, será apresentado na próxima seção.

2.6 Grafo de Causa e Efeito

O critério de teste funcional *Grafo de Causa e Efeito* é complementar ao critério *Tabela de Decisão*. A utilização em conjunto, destes dois critérios aumenta o nível de entendimento das condições em análise e melhora o processo de derivação de casos de testes. Embora seja possível a utilização individual da *Tabela de Decisão*, o contrário não é verdadeiro, pelo menos a partir da perspectiva da derivação dos casos de testes. Pode-se perfeitamente utilizar um grafo, sem transformá-lo numa *Tabela de Decisão*, na atividade de validação de requisitos, mas para a derivação dos casos de testes, este passo é imprescindível. Na literatura especializada é possível encontrar autores referenciando um ou outro critério independentemente, por exemplo: Copeland (2003) e Jorgensen (2002) abordam somente a *Tabela de Decisão*. Myers e Sandler (2004) aborda somente o *Grafo de Causa e Efeito*, mas descreve o passo de transformação do grafo na tabela de decisão, como parte constituinte do processo de utilização do critério. Este passo de transformação do grafo de causa e efeito na tabela de decisão é objeto de estudos de muitos pesquisadores, que buscam desenvolver algoritmos mais eficientes para automatização desta tarefa, como por exemplo Srivastava et al. (2009) e Sharma e Chandra (2010).

A Figura 2.4, adaptada de Myers e Sandler (2004) mostra a representação gráfica deste critério de teste funcional. Nesta figura é possível observar os elementos constitutivos do grafo de causa e efeito, como por exemplo, os **nós** que representam as condições, as **arestas** que representam as avaliações destas condições, conjuntamente com seus **rótulos** representando os operadores lógicos **"identidade", "e", "ou", "não"**. O operador **identidade** é ilustrado na região da figura identificada pelo número 1. O operador **não** na região identificada pelo número 2. O operador **ou** é ilustrado na região identificada pelo número 3 e o operador **e** é ilustrado na região identificada pelo número 4.

Quanto ao processo para utilização deste critério de teste funcional, Myers e Sandler (2004) o define em 6 passos, que são os mesmos definidos no processo de utilização da tabela de decisão:

1. a especificação é dividida em unidades lógicas;
2. as causas e os efeitos são identificados na especificação;

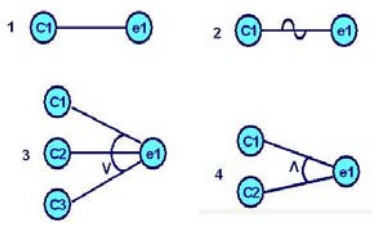

Figura 2.4: *Exemplo do critério de teste funcional Grafo de Causa e Efeito*

3. o conteúdo semântico da especificação é analisado e transformado em um grafo booleano, ligando causas e efeitos;
4. o grafo é anotado com restrições descrevendo combinações de causas e/ou efeitos que são impossíveis, devido à síntaxe ou a restrições do ambiente;
5. o grafo é convertido em uma tabela de decisão;;
6. as colunas na tabela de decisão são convertidas em casos de teste. É gerado um caso de teste para cada coluna da tabela.

Os seis passos constantes do processo, acima descritos, serão seguidos no exemplo, extraído de Hunt (2007) e descrito a seguir para auxiliar na consolidação do entendimento em relação à *Tabela de Decisão* e *Grafo de Causa e Efeito*. Este exemplo contempla o cálculo do valor do prêmio anual de um seguro de automóvel. No passo 1, a análise dos requisitos, foram identificados os requisitos que dizem respeito a este cálculo, que são os seguintes:

- para mulheres com idade inferior a 65 anos, o valor é de R$ 500,00;
- para homens com idade inferior a 25 anos, o valor é de R$ 3.000,00;
- para homens com idade entre 25 e 64 anos, o valor é de R$ 1.000,00;
- para homem ou mulher com idade superior a 65 anos, o valor é de R$ 1.500,00.

No passo 2, um conjunto de cinco causas e quatro efeitos são identificados a partir deste conjunto de requisitos. As causas são numeradas sequencialmente de 1 a 5 e os requisitos de 100 a 103:

1. sexo é masculino;
2. sexo é feminino;
3. idade < 25;
4. $25 \leq$ idade < 65;
5. idade ≥ 65.

100. valor do Prêmio = R$ 1.000,00;
101. valor do Prêmio = R$ 3.000,00;

102. valor do Prêmio = R$ 1.500,00;
103. valor do Prêmio = R$ 500,00;

No passo 3, o grafo de causa e efeito é construído mapeando as causas aos seus respectivos efeitos, conforme pode ser verificado na Figura 2.5. Esta figura está divida em quatro partes. Cada uma destas partes tem a seguinte interpretação:

1. a primeira contém o mapeamento das causas 1 e 4 ao efeito 100, significando que: sexo é masculino e a idade é maior ou igual a 25 e menor que 65, gerando o efeito 100: valor do Prêmio = R$ 1.000,00;
2. a segunda contém o mapeamento das causas 1 e 3 ao efeito 101, significando que: sexo é masculino e idade é menor que 25, gerando o efeito 101: valor do Prêmio = R$ 3.000,00;
3. a terceira contém o mapeamento das causas 1, 2 e 5 ao efeito 102, significando que: o sexo é masculino e a idade é maoir ou igual a 65 ou o sexo é feminino e idade é maior ou igual 65, gerando o efeito 102: valor do Prêmio = R$ 1.500,00;
4. a quarta parte contém o mapeamento das causas 2, 3 e 4 ao efeito 103, significando que: o sexo é feminino e a idade é menor que 25 ou sexo é feminino e a idade é maior ou igual a 25 e menor que 65.

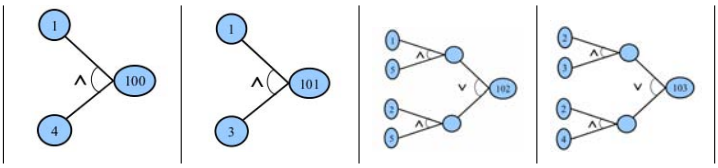

Figura 2.5: *Grafo de Causa e Efeito - Seguro de Veículos*

No passo 4 somente uma anotação denotando a impossibilidade de a pessoa ser do sexo masculino e feminino ao mesmo tempo foi adicionada ao grafo, conforme pode ser observado na Figura 2.6 em que foi adicionada a restrição "o", significando "um e somente um", do inglês *one and only one*. Existem outras restrições, como por exemplo a pessoa possuir mais de uma faixa de idade. Mas, para efeito de simplicação somente uma restrição foi contemplada neste exemplo.

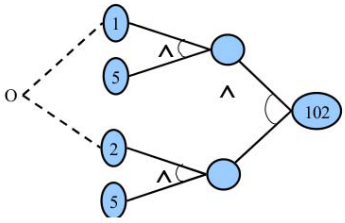

Figura 2.6: *Exemplo de anotação de restrição ao Grafo de Causa e Efeito*

No passo 5 o grafo de causa e efeito é transformado na tabela de decisão. O resultado desta transformação é apresentado na Tabela 2.5. Nesta tabela, uma linha em branco separa as causas dos efeitos. Um total de seis regras identificadas (R1 a R6) estão descritas na primeira linha a partir da segunda coluna. As demais células preenchidas denota o atendimento (1) ou não (0) das condições expressas pelas causas ou a execução ou não da ação expressa pelos efeitos.

Tabela 2.5: *Tabela de Decisão - Seguro de Veículos*

Condições/Regras	R1	R2	R3	R4	R5	R6
1	1	1	1	0	0	0
2	0	0	0	1	1	1
3	1	0	0	0	1	0
4	0	1	0	0	0	1
5	0	0	1	1	0	0
100	0	1	0	0	0	0
101	1	0	0	0	0	0
102	0	0	1	1	0	0
103	0	0	0	0	1	1

No último passo (6), os casos de teste são gerados a partir de cada uma das colunas da tabela de decisão. O conjunto de casos de teste adequado a este exemplo é apresentado na Tabela 2.6.

Tabela 2.6: *Casos de Teste - Seguro Veículos*

Caso de Teste	**Entradas (Causas)**		**Saída Esperada (Efeitos)**
	Sexo	Idade	Prêmio
1	Masculino	< 25	R$ 3000,00
2	Masculino	≥ 25 e < 65	R$ 1000,00
3	Masculino	≥ 65	R$ 1500,00
4	Feminino	≥ 65	R$ 1500,00
5	Feminino	< 25	R$ 500,00
6	Feminino	≥ 25 e < 65	R$ 500,00

Grafo de Causa e Efeito em conjunto com a *Tabela de Decisão* representa um passo evolutivo no processo de teste, contudo não apresenta uma solução definitiva, pois conforme observa Copeland (2003), o testador possui uma caixa de ferramentas para executar suas tarefas, cada ferramenta é um dos critérios de teste disponíveis. Até este momento a caixa de ferramenta já conta com seis critérios, no entanto, ainda existem tipos de defeitos que não são possíveis de serem detectados com a utilização deste conjunto de critérios analisados até o momento. Nas próximas seções novos critérios capazes de detectar novos tipos de defeitos serão abordados, defeitos estes oriundos da interação entre variáveis, por exemplo no teste combinatório, das transições de estados dos objetos em teste e defeitos detectados durante o teste de uma transação completa, por exemplo, o teste baseado em casos de uso.

2.7 Teste em Pares - *Pairwise Testing*

Os critérios de teste funcional abordados nas seções anteriores, *Particionamento em Classes de Equivalência* e *Análise do Valor Limite*, Teste Funcional Sistemático e Teste Funcional Sistemático Estendido são empregados no teste de variáveis individuais e independentes. Os critérios *Tabela de Decisão* e *Grafo de Causa e Efeito* são utilizados nos testes de variáveis que possuam algum tipo de dependência lógica entre elas. O critério de teste funcional abordado nesta seção é o *Teste em Pares* do inglês *Pairwise Testing* (CZERWONKA, 2011), que é a abordagem mais popular do tipo de teste denominado teste combinatório, conforme Bach e Schroeder (2004). Teste Combinatório é aplicado a um cenário em que as variáveis possuem algum grau de interação, ou seja, é aplicado em cenários onde existam várias variáveis e cada uma possua mais de uma opção de valores possíveis. O teste deve ser efetuado sobre a combinação destes vários valores.

Teste combinatório testa a combinação de várias váriaveis, sendo capaz de revelar a presença de defeitos manifestados por estas combinações. No entanto o teste combinatório possui a limitação da explosão combinatória dos cenários de teste, por exemplo, um sistema qualquer contendo cinco váriáveis e cada qual com cinco opções de valores possíveis, gerará um total de 3.125 combinações que devem ser testadas (5^5).

O *Teste em Pares* se apresenta como uma alternativa a este problema, no entanto esta modalidade de teste só é capaz de revelar a presença de defeitos manifestados em virtude da combinação dos valores de variáveis em pares. Isto não representa necessariamente uma limitação do critério de teste, sobretudo levando em consideração que a manifestação da presença de defeitos não se dá em virtude de complexas interações entre variáveis, mas principalmente pela interação de variáveis em pares, conforme relata Bach e Schroeder (2004), citando vários estudos que comprovam esta afirmação.

O *Teste em Pares* reduz significativamente a quantidade de teste para estes cenários. Copeland (2003) apresenta o exemplo de um cenário em que o correto funcionamento de um *web site* depende:

- do navegador *web* que pode ser um entre as oito opções: *internet explorer* 5.0, *internet explorer* 5.5, *internet explorer* 6.0, *Netscape* 6.0, *Netscape* 6.1, *Netscape* 7.0, *Mozilla* 1.1 e Opera 7.0;
- de *plugins* que pode ser um entre as seguintes opções: *RealPlayer*, *MediaPlayer* *Nenhum*;

- do sistema operacional cliente, que pode ser: *windows* 95, *windows* 98, *windows* ME, *windows* NT, *windows* XP, *windows* 2000;
- do servidor *web*, com três opções: *IIS*, *Apache* e *WebLogic*;
- do sistema operacional servidor, com três opções: *windows* NT, *windows* 2000 e Linux.

As combinações destes valores resultará num total de 1296 combinações, dado por (8 * 3 * 6 * 3 * 3). Testar todas estas possibilidades exige a criação de um caso de teste para cada uma delas, o que corresponderia a um teste exaustivo. Desta forma, neste cenário há a necessidade de se buscar um conjunto de casos de testes reduzidos que possua a capacidade de representar todo o conjunto de casos de testes possíveis. Para o teste deste exemplo, Copeland (2003) descreve a redução da quantidade de casos de teste para 64, representando uma redução de 95% em relação ao total de 1296 combinações. Esta redução é exemplificada utilizando a ferramenta *Orthogonal array*, descrita mais adiante.

Este critério de teste, em razão da sua natureza de lidar com quantidades de dados relativamente grandes, vai necessariamente requerer a presença de ferramentas para sua automatização, uma listagem das ferramentas atualmente disponíveis encontra-se em Czerwonka (2011). Dentre estas, as mais populares são: *OATS - Orthogonal Array Test System* Phadke (2000), *AETG - Automatic Efficient Test Generator* Telcordia (2012) e *AllPairs* Bach (2011). Apenas a primeira e a terceira serão abordadas nesta seção.

Para exemplificar um *Array Orthogonal* será utilizado um exemplo presente em Bach e Schroeder (2004) em que um hipotético sistema S possui três variáveis de entrada X, Y e Z. Assume-se que D é o conjunto de valores de dados que foram selecionados para cada uma destas variáveis, tal que D(X) = {1, 2}, D(Y) = {Q, R} e D(Z) = {5, 6}. A Figura 2.7 ilustra graficamente este sistema. Seriam necessários oito (2 * 2 * 2) casos de teste para testar este sistema. Com a utilização do teste em pares, esta quantidade é reduzida para quatro casos de teste, descritos na Tabela 2.7, onde é possível observar que todos os possíveis pares formados pela combinação dos valores desta três variáveis estão presentes. Este pequeno exemplo mostra a redução de 50% da em relação à quantidade de casos de teste sem perda de cobertura. Bach e Schroeder (2004) citam referências sobre estudos que apresentam grandes reduções em relação a quantidade de casos de teste, por exemplo, um conjunto contendo 2^{120} casos de teste foi reduzido para 10 casos de teste e outro em que a quantidade de 10^{29} foi reduzido para 28 casos de teste.

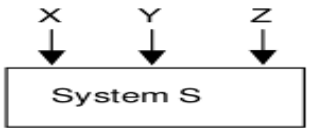

Figura 2.7: *Sistema S com três variáveis de entrada*

Tabela 2.7: *Casos de teste para o Sistema "S" com a utilização do teste em pares*

Teste id	Entrada X	Entrada Y	Entrada Z
TC_1	1	Q	5
TC_2	1	R	6
TC_3	2	Q	6
TC_4	2	R	5

O processo para a utilização do teste em pares com o auxílio do *orthogonal array*, consiste dos seguintes passos:

1. identifique as variáveis;
2. determine o número e as opções para cada variável;
3. aloque um *orthogonal array* que tenha uma coluna para cada variável e valores nas colunas que corresponda as opções de cada variável.
4. preencha o *array* com os valores das opções de cada variável;
5. construa os casos de testes. Sendo um para cada linha do *array*.

A outra ferramenta auxiliar na aplicação deste critério de teste é o *All Pair Algorithm*, como o próprio nome sugere, consiste de um algoritmo destinado à geração de todos os pares possíveis para um conjunto de dados de entrada, e que é implementado por Bach (2011), conforme citado anteriormente. O processo para a utilização do teste em pares auxiliado por esta ferramenta segue os seguintes passos:

1. repita os passos 1 e 2 definidos para a utilização com *orthogonal array*;
2. crie um arquivo de texto com as opções disposta de forma tabular;
3. execute o algoritmo informando o arquivo gerado no passo anterior, como entrada.

A principal vantagem da utilização deste critério de teste é a redução da quantidade de casos de testes. Como desvantagem podem ser consideradas algumas questões citadas por Bach e Schroeder (2004) descrevendo quando o teste em pares falha:

- quando não se seleciona os valores corretos de entrada para os testes;

- quando não se tem um oráculo suficientemente bom;
- quando as combinações altamente prováveis recebem pouca atenção;
- quando a forma de interação entre as variáveis não é conhecida.

2.8 Teste de Transição de Estados

Teste de Transições de Estados é uma técnica de teste funcional que auxilia no levantamento de requisitos. A técnica propriamente dita é simples e é destinado ao teste: (i) dos possíveis estados de um objeto; (ii) de suas possíveis transições; (iii) dos eventos internos que motivam as transições e (iv) das ações executadas para a efetiva transição de um estado para outro. A modelagem de estados, utiliza uma ferramenta gráfica denominada Diagrama de Estados, proposto por David Harel em Harel (1987) e que foi posteriormente foi incorporado à UML - Linguagem de modelagem unificada, OMG (2011), tornando-se um de seus diagramas constituintes. Em adição, é importante observar que a implementação desta técnica de modelagem de transição de estados, também foi objeto de estudos da famosa "gang" dos quatro no clássico livro sobre padrões de projeto, onde definem o padrão *state*, como um dos vinte e três padrões de projeto introduzidos em Gamma et al. (1995).

Esta técnica de modelagem trabalha com os seguintes conceitos:

1. **Estado** - conjunto de valores dos dados do sistema em um determinado momento. Podem ser os seguintes:
   (a) **Estado inicial** - estado do sistema ou componente em que o primeiro evento é aceito;
   (b) **Estado origem / Estado destino** - uma transição leva o sistema do estado de origem para o estado de destino, os quais podem ser iguais;
   (c) **Estado atual** - estado corrente em que se encontra a execução do sistema;
   (d) **Estado final** - estado do sistema no qual eventos não são mais aceitos. O sistema pode ter nenhum ou muitos estados finais.
2. **Transição** - conduz o sistema de um estado para outro devido à ocorrência de um evento;
3. **Evento** - entrada ou período de tempo;
4. **Ação** - resultado ou saída produzida em resposta ao evento ocorrido.

A Figura 2.8, obtida de Copeland (2003), apresenta um modelo de diagrama de transicão de estados referente a uma reserva de passagem aérea. Nesta figura é possível observar a presença dos elementos constituintes da modelagem de estados,

como por exemplo os diversos estados, inicial, intermediários e final; as transições entre estados, os eventos que motivam as transições e o fluxo de estados em virtude das transições, etc.

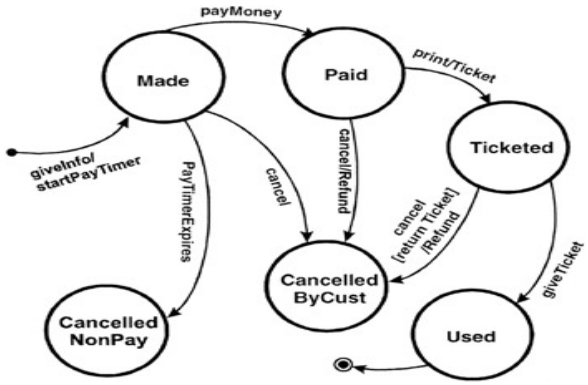

Figura 2.8: *Exemplo de diagrama de transição de estados.*

O processo para a utilização desta técnica de teste funcional é composto pelos seguintes passos:

1. identificar os potenciais estados que o objeto em teste pode assumir durante seu ciclo de vida;
2. identificar as possíveis transições entre estes estados;
3. identificar os eventos externos que motivam as transições entre estados;
4. identificar as ações que são executadas pelo objeto durante a transicão de um estado para outro;
5. desenhar o diagrama de transição de estados;
6. transformar o diagrama em uma tabela de transição de estados;
7. construir os casos de testes a partir da tabela de transição de estados, criando um caso de teste para cada linha desta tabela.

Um exemplo da sua aplicação consta da Tabela 2.8 obtida de Copeland (2003), descrevendo de forma tabular as transições de estado constantes da Figura 2.8. Vale ressaltar que foram transcritas somente as linhas que representam estados válidos, por simples questão de senso prático, uma vez que a tabela apresentada no exemplo é muito extensa dispondo de linhas para todos as potenciais transições de estados, mesmo aquelas identificadas como impossíveis de ocorrer.

Um diagrama de transiçao de estados oferece muito mais facilidade para se visualizar os estados e as transições entre estes, inclusive os eventos e a ações disparadas por estes eventos. Contudo a tabela de transição de estados, também é bastante rica em informações e constitui-se numa poderosa ferramenta para a

Tabela 2.8: *Exemplo de uma tabela de transição de estados*

Estado atual	evento	ação	próximo estado
inicial	obtemInfo	iniciaTempoPagamento	reservada
reservada	efetuaPagamento		Paga
reservada	cancelar		Cancelada-cliente
reservada	TempoPagamentoExpirado		Cancelada-NãoPaga
Paga	imprimir	emitirBilhete	BilheteEmitido
Paga	cancelar	Reembolsar	Cancelada-cliente
BilheteEmitido	obtemBilhete		usada
BilheteEmitido	cancelar	reembolsar	cancelada-cliente

modelagem destes estados. Na Tabela 2.8 é possível verificar algumas situações que devem ser descritas na forma de requisitos, e consequentemente serem testados:

- uma reserva é criada e assume como seu primeiro estado *reservada*;
- a partir do estado *reservada* é possível que a reserva transite para os estados *paga*, *cancelada pelo cliente* e *cancelada por falta de pagamento*;
- a partir do estado *paga* é possível que a reserva transite para os estados *BilheteEmitido* e *cancelada pelo cliente*;
- a partir do estado *BilheteEmitido* é possível que a reserva transite para os estados *usada* e *cancelada pelo cliente*.
- Não existe nenhuma transição de estados para uma reserva nos estados *cancelada pelo cliente* e *cancelada por falta de pagamento*;
- o cliente pode cancelar a reserva a qualquer momento, desde que ela ainda não tenha sido *usada*;
- o cancelamento solicitado pelo cliente sempre gerará um reembolso.

Este pequeno exemplo mostra a riqueza de informações que esta ferramenta carrega e quão poderosa se torna como técnica de teste.

2.9 Teste Baseado em Casos de Uso

A técnica de teste descrita nesta seção é o *Teste Baseado em Casos de Uso*, que é destinado ao teste de transações, ou seja, testar a execução de uma funcionalidade do início ao fim. De acordo com Copeland (2003), difere das técnicas abordadas anteriormente, onde o foco era variáveis, individuais ou mesmo em conjunto, mas sem um escopo definido como é o caso de uma transação.

Caso de uso é uma técnica para a documentação de requisitos funcionais, proposta por Jacobson et al. (1992), onde é definido como "um cenário que descreve o uso de um sistema, por um ator para alcançar um objetivo específico". Um ator, na

perspectiva de um caso de uso, é um agente externo (uma pessoa ou outro sistema) que executa alguma funcionalidade do sistema em um determinado contexto. Um cenário é uma sequência de passos e interações entre o ator e o sistema. A Figura 2.9, obtida de Barros (2011), ilustra um diagrama de caso de uso para um sistema simulador de ambiente. Nesta figura é possível observar que casos de usos são iniciados por um ator e possui relacionamentos entre si. Estes relacionamentos podem ser de:

- **inclusão** - quando um caso de uso "A" inclui (denotado pelo estereótipo <*include*>) um caso de uso "B", significa que sempre que "A" for executado, "B" também será;
- **extensão** - quando um caso de uso "A" tem um relacionamento do tipo extensão (denotado pelo estereótipo <*extends*>) com outro caso de uso "B", implica que ao executar o caso de uso "A" não necessariamente "B" será executado.

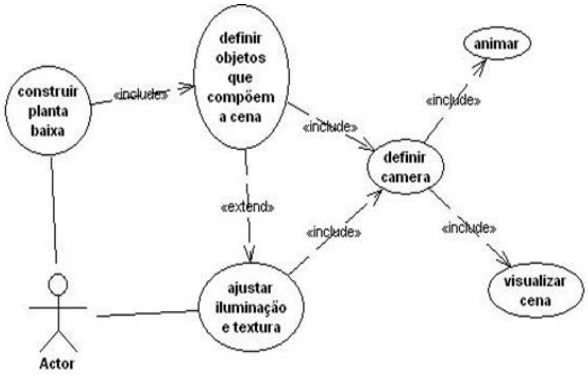

Figura 2.9: *Exemplo de diagrama de caso de uso*

A descrição de um caso de uso utiliza uma linguagem de negócio e não uma linguagem técnica e é desenvolvida na forma de fluxos de interações entre o ator e o sistema. Estes fluxos são divididos em "Fluxo Principal" e "Fluxos Alternativos", sendo que o fluxo principal representa a sequência de passos que cobrem o que normalmente acontece quando o caso de uso é executado. Os fluxos alternativos representam um comportamento opcional ou excepcional em relação ao comportamento normal presente no fluxo principal. Pode-se pensar nos fluxos alternativos como desvios a partir do fluxo principal. Desta descrição do caso de uso devem constar ainda as pré-condições que devem ser atendidas para a execução do caso de uso e as pós-condições que são os resultados e o estado final do sistema após a execução bem sucedida do caso de uso.

A Figura 2.10 obtida de Rational (2010) apresenta uma estrutura típica de fluxos de eventos de um caso de uso:

Figura 2.10: *Exemplo de fluxos de um caso de uso*

A Tabela 2.9, obtida de Rational (2010) apresenta os possíveis cenários para execução do caso de uso, cujos fluxos estão ilustrados na Figura 2.10.

Tabela 2.9: *Exemplo de cenários de um caso de uso*

Cenários	Fluxos			
Cenário 1	Fluxo Básico			
Cenário 2	Fluxo Básico	Fluxo Alternativo 1		
Cenário 3	Fluxo Básico	Fluxo Alternativo 1	Fluxo Alternativo 2	
Cenário 4	Fluxo Básico	Fluxo Alternativo 3		
Cenário 5	Fluxo Básico	Fluxo Alternativo 3	Fluxo Alternativo 1	
Cenário 6	Fluxo Básico	Fluxo Alternativo 3	Fluxo Alternativo 1	Fluxo Alternativo 2
Cenário 7	Fluxo Básico	Fluxo Alternativo 4		
Cenário 8	Fluxo Básico	Fluxo Alternativo 3	Fluxo Alternativo 4	

O processo para derivação de casos de teste a partir dos casos de uso deve seguir pelos menos os três passos:

1. Para cada caso de uso, gere um conjunto completo de cenários;
2. Para cada cenário, identifique pelo menos um caso de teste e as condições que o tornarão executável;
3. Para cada caso de teste, identifique os valores dos dados com os quais testar.

Particularmente, a derivação de casos de testes a partir dos casos de uso pode apresentar os seguintes problemas:

1. o projeto de um caso de uso não tem a intenção de revelar defeitos, então é possível que utilizando casos de usos como fonte para os testes não se consiga revelar muitos defeitos, o que pode tornar esta técnica pobre a partir desta perspectiva;

2. caso de uso promove a impressão que os testes podem ser executados por qualquer um (ator), independente das habilidades especificas para a atividade de teste;
3. caso de uso promove a idéia de que o objetivo do teste é mostrar que o sistema funciona e não de revelar defeitos.

Em relação à cobertura alcançada por este tipo de teste, dois critérios são utilizados: (i) *todos os cenários*, onde cada cenário deve ser testado por pelo menos um caso de teste e (ii) *todas as transições*, onde todas as transições devem ser testadas pelo menos uma vez.

As vantagens de se utilizar a modelagem do sistema utilizando casos de uso, dentre outras, apresentadas por Copeland (2003), são as seguintes:

1. captura os requisitos funcionais do sistema a partir da perspectiva dos usuários; não a partir de uma perspectiva técnica e independente de qual paradigma de desenvolvimento empregado na construção do sistema;
2. pode ser utilizado para o efetivo envolvimento dos usuários, tanto no levantamento de requisitos, quanto na definição de processos.
3. fornece uma base para a identificação dos principais componentes internos do sistema, estruturas de dados, banco de dados e relacionamentos;
4. serve como base para o desenvolvimento de casos de testes no nível de teste de aceitação.

Caso de uso, como uma técnica de documentação de requisitos, está sujeito aos mesmos problemas inerentes a qualquer outro método de documentação (incluindo as tradicionais especificações de requisitos), alguns problemas que o testador deve estar ciente, é que um caso de uso pode (i) não estar completo, (ii) não estar suficientemente detalhado, (iii) ser impreciso, (iv) não ter sido revisto, (v) não ter sido atualizado quando da mudança em algum requisito e (vi) ser ambíguo.

2.10 Teste Aleatório

Teste Aleatório *(Random Testing)* normalmente não é considerado um critério de teste funcional. Consiste de uma estratégia tanto para geração, quanto para a seleção de dados de teste utilizando como fonte todo o domínio de entrada do programa em teste. Relativamente à seleção, todos os elementos deste domínio possuem a mesma probabilidade de serem selecionados.

O Teste Aleatório é o oposto do teste por particionamento, pois nesta estratégia de teste não há subdivisão de domínio, este é considerado na sua totalidade, não há a visão de comportamento igualitário dos elementos. Por este motivo, frequentemente comparado ao teste por particionamento. Myers (1978) considera que o teste aleatório é a mais pobre dentre as metodologias para a geração de casos de teste. Duran e Ntafos (1984) observa que intuitivamente o teste por particionamento seja superior ao teste aleatório no quesito detecção de defeitos, no entanto devido ao maior esforço dispensado por este, estes resultados não são tão relevantes, a maior capacidade de detecção de defeitos não é compensadora em relação ao grande esforço e custo dispensados. Hamlet e Taylor (1990) não acharam estes resultados muito intuitivos e replicaram o trabalho de Duran e Ntafos (1984), chegando praticamente aos mesmos resultados. Weyuker e Jeng (1991) analisa estas duas técnicas de teste a partir de uma perspectiva teórica e os resultados apontam na mesma direção.

Se o teste por particionamento é superior ao teste aleatório, então por que utilizar o teste aleatório e em que circunstâncias? Ciupa et al. (2008) observa que "A geração aleatória de dados de entrada é atraente por ser largamente aplicável e possuir custo baixo, tanto em termos de esforço de implementação quanto em tempo de execução". Hamlet (2006) apresenta situações nas quais somente o teste aleatório é aplicável, como por exemplo: (i) domínio não subdividíveis, isto é, sem a definição explicita de delimitações e (ii) estados persistentes, onde não exista a definição de um estado padrão.

Recentemente têm sido publicadas muitas pesquisas relativas ao teste aleatório, sobretudo em relação a uma nova variação do teste aleatório, o teste aleatório adaptativo, alguns exemplos destas publicações são: (CHEN et al., 2005), (CHEN et al., 2007), (CHEN et al., 2010) e (MAYER; SCHNECKENBURGER, 2006). Chen et al. (2010) observa do que o teste aleatório adaptativo é uma evolução em relação ao teste aleatório. Considerando que aspectos tais como, custo, eficácia e eficiência de muitas técnicas e critérios de teste são avaliados em comparação como teste aleatório, qualquer melhoria porporcionada a este teste, terá impacto significativo em outras técnicas e critérios de teste.

2.11 Considerações Finais

Este capítulo abordou o teste funcional, em que os casos de teste são derivados a partir da especificação do software. Foi desenvolvida uma análise minuciosa de vários dos critérios e técnicas, considerados como bons representantes de todo o

conjunto dos critérios e técnicas funcionais, tendo em vista a reprentação do teste de variáveis individuais, dependências lógicas entre variáveis, teste combinatório, transição de estados e teste de transações. Foram analisados: *Particionamento em Classes de Equivalência*, *Análise do Valor Limite*, *Tabela de Decisão*, *Teste Baseado em Casos de Uso* dentre outros.

Sobre os critérios e técnicas funcionais, foram abordados aspectos teóricos, pontos fortes e fracos, e alguns exemplos de aplicação dos mesmos. Os exemplos apresentados empregaram os critérios ou técnicas isoladamente, com exceção das seções que abordaram Grafo de Causa e Efeito e Tabela de Decisão, em que o exemplo empregou os dois em conjunto.

Uma revisão sistemática será explorada nos Capítulos 3 e 4, visando a esclarecer outros elementos pertinentes ao teste funcional, por exemplo, os seus cenários de utilização, o emprego conjunto dos critérios e técnicas, aspectos de comparação entre os critérios/técnicas funcionais e se existem pesquisas que avaliam especificações de teste. Os resultados desta revisão sistemática subsidiarão a proposição da solução do problema associado à pesquisa deste trabalho, tendo em vista que a análise dos potenciais estudos primários fornecerá elementos que vão possibilitar a avaliação do custo associado à utilização dos principais critérios e técnicas de teste funcional e a capacidade de detecção de defeitos de cada um deles.

CAPÍTULO 3

Protocolo de Revisão Sistemática Sobre Teste Funcional

3.1 Planejamento

O planejamento do protocolo desta revisão sistemática foi elaborado conforme o modelo apresentado em Barbosa (2011), que utilizou o modelo definido por Biolchini et al. (2007). Nesta seção, são apresentados os principais pontos do plano elaborado.

3.1.1 Objetivos da Pesquisa

Sabe-se a importância do teste funcional na melhoria de qualidade de software, que é uma abordagem complementar a outras técnicas de teste. Nesse sentido, é pertinente conhecer como os critérios/técnicas de teste funcional são empregados, identificar pontos fortes e pontos fracos, e observar os cenários em que estão sendo aplicados.

3.1.2 Formulação da Questão de Pesquisa

Esta revisão sistemática foi motivada pela busca de respostas para as seguintes questões:

- **Questão de Pesquisa Primária**: Que comparações têm sido realizadas entre os critérios/técnicas de teste funcional?
- **Questão de Pesquisa Secundária 1**: Qual o cenário para a aplicação de cada critério/técnica de teste funcional?
- **Questão de Pesquisa Secundária 2**: Que critérios/técnicas de teste funcional têm sido aplicados para avaliar roteiros (especificações) de teste?

A composição das respostas às questões de pesquisa darão suporte à solução ao problema atribuído à pesquisa deste trabalho, definindo uma abordagem para a aplicação do teste funcional, uma potencial contribuição à melhoria de qualidade de software, de forma que:

- A Questão de Pesquisa Primária objetiva conhecer os pontos fortes e fracos dos critérios/técnicas de teste funcional, pela comparação entre si, observando-se várias dimensões, tais como custo de aplicação e habilidade para a detecção de defeitos. Essa questão é dita primária pois: (i) provê informação sobre a forma de aplicação e limitações; (ii) determina os fatores que influenciam a eficiência e eficácia; e (iii) fornece suporte para a proposição abordagens relativas ao teste funcional.
- A Questão de Pesquisa Secundária 1 busca caracterizar o tipo de software em que os critérios/técnicas funcionais são empregados. É importante pois estabelece a abrangência de aplicação dos critérios/técnicas funcionais, e pode determinar a predominância e a restrição de emprego em algumas áreas.
- A Questão de Pesquisa Secundária 2 explora um interesse específico deste trabalho: agregar valor ao Roteiro de Teste PAF-ECF, visando à melhoria de qualidade do software fiscal em teste. Define-se aqui, Roteiro de Teste, sinonímico de Especificação de Teste, como instruções voltadas à geração de casos de teste e à sua sequência de aplicação. São também incluídas instruções sobre o ambiente de teste (por exemplo, funcionamento em rede para acesso a dados remotos), layout de arquivos de configuração, operação de periféricos (por exemplo, alteração de data de impressora fiscal), etc. Para responder a essa questão de pesquisa, são procurados estudos que avaliam a qualidade de roteiros de teste.

3.1.3 Qualidade e Amplitude da Questão

Uma questão de pesquisa bem formulada é composta pelos seguintes elementos:

3.1.3.1 Palavras-chaves e sinônimos

Foram consideradas como palavras-chaves da língua inglesa as seguintes palavras:

1. **População**: “software test”, “software testing”, “defect detection”, “software validation”;

2. **Intervenção**: “functional testing”, “black-box testing”, “specification-based testing”, “requirements-based testing”;
3. **Resultado**: “characteristic”, “attributes”, “property”, “criteria”, “evaluation”, “application”.

3.1.3.2 Intervenção

Critérios e técnicas de teste funcional. O conjunto de critérios e técnicas de teste funcional é grande. Não será abordado na sua totalidade. Foi selecionado um subconjunto destes critérios e técnicas representativo de todo o conjunto. Este subconjunto é composto pelos critérios e técnicas listados a seguir e que estão descritos detalhadamente no Capítulo 2:

1. Particionamento por classes de equivalência;
2. Análise do valor limite;
3. Teste Funcional Sistemático;
4. Teste Funcional Sistemático Estendido;
5. Grafo de causa e efeito
6. Tabelas de decisão
7. Teste Aleatório (Random Testing). Não é teste funcional, no entanto é utilizada, com uma certa frequência, em comparações com os critérios e técnicas funcionais.

3.1.3.3 Controle

Foram definidos os seguintes artigos, constantes da Tabela 3.1 como controle da pesquisa. Todos foram retornados na busca efetuada nas fontes selecionadas, demonstrando desta forma a qualidade da *string* de busca definida.

Tabela 3.1: *Artigos de controle*

Item	Referência	Abordagem
AC1	Basili e Selby (1987)	Comparação de técnicas de teste
AC2	Nebut et al. (2006)	Testes baseados em casos de uso
AC3	Noikajana e Suwannasart (2008)	Tabela de decisão
AC4	Vij e Feng (2008)	Análise do Valor Limite

3.1.3.4 População

O grupo observado foi o de pesquisadores e desenvolvedores de software que trabalham no escopo das técnicas de teste funcional.

3.1.3.5 Resultados

- propriedades, características e comparações entre critérios/técnicas de teste funcional;
- contexto de aplicação de cada critério/técnica de teste funcional analisados;
- subsídios para avaliação de especificações de teste pela aplicação das propriedades/particularidades dos critérios/técnicas de teste funcional.

3.1.3.6 Aplicação

Servir de base ou apoiar pesquisas envolvendo:

- estabelecimento de relações entre os critérios/técnicas de teste funcional;
- fornecer subsídios para a avaliação de especificações (não formais) de testes, a partir da perspectiva de teste funcional.

3.1.4 Estratégia de Busca para Seleção de Estudos Primários

A estratégia de busca e seleção dos estudos primários foi definida de acordo com as fontes de estudos, palavras-chave, idioma e os tipos de estudos primários selecionados para a revisão:

3.1.4.1 Critério de seleção das fontes

Bases de dados eletrônicas indexadas e máquinas de busca eletrônica.

3.1.4.2 Métodos de busca de fontes

Manual e máquina de busca na web.

3.1.4.3 Listagem de fontes

As fontes serão bases de dados eletrônicas, disponíveis no portal CAPES, incluindo conferências, journals e relatórios técnicos indexados por:

- IEEExplore;

- ACM Digital Library
- Google Scholar

Estas fontes foram escolhidas devido à sua difusão no meio acadêmico, que atesta qualidade e confiabilidade, e à facilidade de acesso para recuperação de referências, incluindo o texto completo. Em adição, são pertinentes pois oferecem publicações na área e que podem contribuir significativamente para o resultado da pesquisa.

3.1.4.4 Tipo dos estudos primários

Listas de referência de estudos primários, periódicos, relatórios técnicos, trabalhos em andamento e *proceedings* de conferências.

3.1.4.5 Idioma dos estudos primários

Inglês, por ser a língua internacionalmente aceita para a redação de trabalhos científicos. Em adição, textos em português, embora se reconheça a sua importância, podem não estar adequadamente indexados, o que aumenta o esforço ou impede sua busca.

3.1.5 Execução de Busca Piloto

A partir das questões de pesquisa, dos seus respectivos atributos de qualidade e amplitude e estratégia de busca para seleção de estudos primários, definiu-se uma *string* de busca para cada base eletrônica indexada. Uma vez aplicada, esta *string* de busca, às bases selecionadas, há uma avaliação inicial dos estudos primários retornados e adequações nas etapas anteriores.

3.1.6 Critérios e Procedimento para Seleção dos Estudos

3.1.6.1 Critérios de inclusão

Os seguintes critérios de inclusão de trabalhos foram definidos:

1. $\mathbf{CI}_1$ - Artigos que abordam qualquer característica de algum dos critérios/técnicas de teste funcional;
2. $\mathbf{CI}_2$ - Artigos que tratam da comparação entre propriedades das técnicas e critérios de teste funcional;

3. **CI$_3$** - Artigos que tratam da comparação entre propriedades das técnicas e critérios de teste funcional, estrutural e técnica de teste aleatório;
4. **CI$_4$** - Artigos que abordam questões relativas a especificações de teste, tais como qualidade, formas de elaboração da especificação, geração automática, etc..

3.1.6.2 Critérios de exclusão

Os seguintes critérios de exclusão de trabalhos foram definidos:

1. **CE$_1$** - Artigos que apenas referenciam teste de software, sem que este seja o tema central;
2. **CE$_2$** - Artigos que abordam teste de software, mas cujo foco não seja nas técnicas de teste funcional ou técnicas de teste aleatório;
3. **CE$_3$** - Artigos que abordam técnicas de teste funcional, mas que não constam dos grupos de técnicas definidas previamente para a análise;
4. **CE$_4$** - Artigos que abordam técnicas de teste funcional, mas cujo foco não conste em nenhuma das categorias definidas previamente para análise;
5. **CE$_5$** - Artigos que descrevem sistemática de avaliação de critérios/técnicas de teste, *frameworks*, *benchmarks* para a comparação de técnicas de testes, que descrevem condições necessárias para fazer a comparação de técnicas de teste, mas que efetivamente não efetuam qualquer comparação;
6. **CE$_6$** - Artigos que fazem comparação entre técnicas de teste, mas que não incluem técnicas de teste funcional entre as técnicas comparadas;
7. **CE$_7$** - Artigos que abordam técnicas de teste funcional exclusivamente em relação a especificações formais;
8. **CE$_8$** - Artigos focados em análise teórica, sem pelo menos exemplificar o uso prático da abordagem.

3.1.7 Processo de Seleção dos Estudos Primários

3.1.7.1 Processo de seleção preliminar

Nesta etapa, foram construídas *strings* de busca formadas pela combinação dos sinônimos das palavras-chaves identificadas. Essas *strings* foram utilizadas para se realizar as respectivas consultas nas máquinas de busca mencionadas. Os trabalhos recuperados por meio das respectivas consultas foram analisados pelos revisores, que foram responsáveis pela leitura dos títulos e dos resumos dos trabalhos, identificada

a relevância de um trabalho, e existindo consenso entre os revisores, o referido trabalho foi selecionado para ser lido na íntegra. Não existindo consenso, o trabalho foi colocado em uma lista de espera, para definição futura pelos revisores.

3.1.7.2 Processo de seleção final

Foi realizada a leitura completa dos trabalhos selecionados na etapa de seleção preliminar por pelo menos um dos revisores, que redigiu um documento com o resumo, metodologia e técnicas de teste mencionadas no trabalho e outros conceitos relacionados.

3.1.7.3 Avaliação da qualidade dos estudos primários

Os estudos selecionados resultante da execução do processo de seleção dos estudos primários foram avaliados pelos pesquisadores envolvidos de acordo com os critérios de qualidade definidos por Ali et al. (2010):

1. Existe uma razão específica que motivou a realização do estudo?
2. Existe uma descrição adequada do contexto (por exemplo indústria, laboratório, produtos utilizados e etc) em que a pesquisa foi realizada?
3. Existe uma justificativa e uma descrição para o projeto de pesquisa?
4. O pesquisador explicou como a amostra do estudo (participantes ou casos) foi identificada e selecionada e qual foi a justificativa para essa seleção?
5. Está claro como os dados foram coletados (por exemplo, por meio de entrevistas, formulários, observação, ferramentas e etc)?
6. O estudo fornece uma descrição e justificativa dos métodos de análise de dados utilizados?
7. Existem dados suficientes que foram apresentados com o objetivo de sustentar as conclusões?
8. Há uma declaração clara dos resultados?
9. O pesquisador analisou criticamente o seu próprio papel, o viés potencial e influência na formulação de questões de investigação, o recrutamento da amostra, coleta de dados, análise e seleção de dados para apresentação?
10. Os autores discutem a credibilidade dos seus resultados?
11. As limitações do estudo foram discutidas explicitamente?

A avaliação da qualidade de cada estudo definiu sua exclusão ou inclusão na lista de estudos que foi utilizada para extrair os dados. Ao avaliar cada estudo segundo

os critérios acima, foi obtido uma correspondente nota final numa escala de 0 a 11 pontos, sendo que cada questão foi pontuada da seguinte forma: se a resposta foi "Sim" (1 ponto), se foi "Não" (0 ponto) e se foi "Parcialmente" (0.5 ponto). Ao final da avaliação, foram excluídos estudos que com avaliação igual a "Muito fraco", aqueles cujas notas ficaram entre 0 e 2.4 pontos, ou "Fraco", aqueles cujas notas ficaram entre 2.5 e 4.4 pontos, restando assim apenas estudos com avaliação "Regular" (4.5 a 5.9 pontos), "Bom" (6 a 8.4 pontos) e "Muito Bom" (8.5 a 11 pontos).

3.1.8 Estratégias de Extração e Sumarização dos Resultados

Para cada estudo primário selecionado, foi utilizado a ferramenta JabRef para armazenar os dados (ALVER, 2008).

3.1.8.1 Sumarização dos resultados

A Tabela 3.2 apresenta o esquema para extração de informações, que sintetiza as informações dos estudos primários. A Coluna 1 descreve os aspectos de interesse para os estudos primários; a Coluna 2 explica tais aspectos e, em alguns casos, apresenta questões que devem ser respondidas. Os aspectos de interesse são instanciados para cada estudo primário, conforme posto no Apêndice B, desta dissertação.

3.1.9 Força das evidências

A força geral de um corpo de evidência é normalmente referido como a força da evidência. Uma análise da força da evidência é muito importante para que os leitores de uma revisão sistemática tenham condições de identificar o grau de confiança que se pode colocar nas conclusões e recomendações resultantes dessas revisões (ALI et al., 2010) e (DYBÅ; DINGSØYR, 2008).

Existem diversos sistemas para avaliar a força das evidências, porém neste trabalho foi escolhido o GRADE (*Grading of Recommendations Assessment, Development and Evaluation*), visto que as definições prevista no GRADE aborda a maioria das fragilidades dos sistemas de classificação de evidências baseado em hierarquia e também por ser utilizado por outros pesquisadores em engenharia de software (ALI et al., 2010).

GRADE define quatro graus de força das evidências: alta, moderada, baixa e muito baixa (conforme Tabela 3.3). A força das evidências é determinada por meio

Tabela 3.2: *Esquema para extração de informações*

Aspecto de Interesse	Descrição
1. Título e Referência	Título e Referência bibliográfica do estudo primário analisado.
2. Descrição sucinta	Apresentação do contexto e breve descrição do artigo (qual o propósito do artigo?).
3. Critério(s)/técnica(s) de teste explorado(s)	Identificação do(s) critério(s)/técnica(s) de teste investigado(s).
4. Abordagem para o teste	Identificação da abordagem para o teste, que pode ser, por exemplo: "geração de dados de teste", "seleção de dados de teste", "avaliação de dados de teste existentes", que inclui adequação ao teste ou "avaliação de especificações de teste". Pode incluir informações adicionais sobre a abordagem.
5. Proposição de novo critério de teste	Identificação e breve descrição sobre o critério proposto.
6. Classificação e descrição sucinta da análise realizada	Breve apresentação da análise realizada em relação ao método de validação da abordagem ou dos resultados obtidos nos estudos comparativos. Esclarecimento se houve ou não validação dos dados. Em caso afirmativo, identificar se a validação foi efetuada através de experimento, estudo de caso, simulação, asserção, *survey*, etc. As definições destes métodos de validação constam do glossário, presente no Apêndice A, deste trabalho.
7. Comparação entre critérios/técnicas de teste	Identificação dos critérios/técnicas de teste comparados, descrevendo os atributos de avaliação e resultados.
8. Cenário de aplicação de cada critério/técnica	Cenário em que os critérios/técnicas foram aplicados, durante a análise empírica.
9. Automação do teste	Se houve esforço de automação para o teste, descrevendo e indicando, se possível, seu impacto no custo de aplicação.
10. Utilização conjunta de critérios/técnicas	Se houve a aplicação conjunta de critérios/técnicas, visando à melhoria de qualidade e redução de custo. Ou seja, o artigo explora a aplicação complementar de critérios/técnicas de teste. Utilização conjunta significa que mais de um critério/técnica de teste foi aplicado para a geração de um conjunto de testes.
11. Síntese dos resultados e contribuições	Apresentação dos resultados e contribuições (quais os resultados a partir da análise realizada?).
12. Observações complementares	Outras observações pertinentes às questões de pesquisa.

da combinação de quatro elementos: características do estudo, qualidade do estudo, consistência e objetividade (*directness*).

Tabela 3.3: *Definições utilizadas para classificar a força das evidências*

Grau	Definição
Alta	Pesquisas futuras são muito improváveis que mude a confiança na estimativa do efeito
Moderada	Pesquisas futuras são susceptíveis que provoque um impacto importante sobre a confiança na estimativa do efeito, podendo assim alterar a estimativa
Baixa	Pesquisas futuras são muito susceptíveis que provoque um impacto importante sobre a confiança na estimativa do efeito e é susceptível que altere a estimativa
Muito Baixa	Qualquer estimativa do efeito é muito incerto

3.2 Considerações Finais

Neste capítulo foi definido e descrito o protocolo do planejamento da revisão sistemática, abordando, dentre outros, os objetivos da pesquisa, formulações das questões de pesquisa, estratégias para busca e seleção dos estudos primários, forma de extração de informações e formas de avaliação da qualidade destes estudos primários. Os resultados, as respostas às questões de pesquisa e a avaliação dos estudos primários constam do Capítulo 4. As informações extraídas de cada um dos estudos primários selecionados constam do Apêndice B. Os detalhes da condução da revisão sistemática, tais como as *strings* utilizadas nas buscas, o quantitativo de estudos incluídos e excluídos, constam do apêndice C.

CAPÍTULO 4

Análises e Resultados de Revisão Sistemática Sobre Teste Funcional

Neste capítulo são apresentados os trabalhos relativos à análise dos estudos primários selecionados, desde a extração de informações, passando pela análise de cada a uma das questões de pesquisa e concluindo com uma análise destes estudos em relação à força das evidências ali contidas. As análises sobre os estudos primários seguem o mesmo modelo utilizado por Barbosa (2011).

4.1 Análise dos Trabalhos Selecionados

Nesta seção, os itens constantes do esquema de extração de informações (Tabela 3.2) são tratados isoladamente na Subseção 4.1.1 e na Seção 4.5, conforme a análise dos trabalhos selecionados, sendo que nas Seções 4.2 a 4.4, a análise dá-se em relação às questões de pesquisa.

4.1.1 Critérios e técnicas de teste explorados

A Tabela 4.1 apresenta os critérios, técnicas de teste e abordagens de inspeção identificados em cada um dos estudos primários. Ressalta-se que a primeira coluna lista os artigos por data de publicação e a segunda coluna elenca os critérios, técnicas e, em alguns casos, as abordagens de teste não necessariamente identificadas na forma de um critério definido na literatura. Sobre a tabela, observa-se: (i) os estudos, em geral, abordam mais de um critério/técnica de teste; (ii) em muitos casos, critérios funcionais, estruturais e outras técnicas de teste ou de inspeção de código são comparados em um mesmo artigo; (iii) os critérios Análise do Valor Limite e

Particionamento em Classes de Equivalência estão presentes em quase todos os anos das publicações.

A Tabela 4.2 apresenta os critérios/técnicas de teste explorados pelos estudos primários analisados, e que são do interesse desta revisão sistemática, conforme definido na Subseção 3.1.3.2, juntamente com outras técnicas e abordagens de teste. A primeira coluna identifica o critério/técnica; a segunda coluna determina a quantidade de estudos primários que os referenciam. Os Critérios Particionamento Aleatório e Particionamento Dinâmico são derivados do Particionamento em Classes de Equivalência.

Os Critérios Análise do Valor Limite, Particionamento em Classes de Equivalência, Tabela de Decisão e Testes Baseados em Casos de Uso são os mais explorados, mostrando acerto em relação aos critérios/técnicas de interesse desta pesquisa, conforme o planejamento da revisão sistemática. O somatório da quantidade de estudos mencionados na Tabela 4.2 é superior ao número de estudos primários analisados (43 contra 27), constatando-se numericamente que os estudos primários abordam em sua maioria vários critérios/técnicas de teste funcional.

4.1.2 Abordagem para o teste

Dos 27 estudos primários analisados, 22 abordam a geração de casos de teste, seja para comparação com outros critérios/técnicas, seja para a validação de alguma abordagem/ferramenta. Dois estudos, Gutierrez et al. (2006) e Escalona et al. (2011), conduzem *surveys* explorando abordagens para a geração de casos de testes a partir dos requisitos funcionais. Apenas Cai et al. (2005) aborda a seleção de casos de teste. Jones (2005) foca a geração e avaliação de dados de teste. E, por fim, Murnane et al. (2005) aborda a geração e seleção de dados de teste.

4.1.3 Proposição de novo critério de teste

Três estudos propõem novo critério de teste: Cai et al. (2005) propõe o Particionamento Dinâmico, utilizado na seleção de casos de teste. Linkman et al. (2003) propõe o Teste Funcional Sistemático e Vidal (2011) propõe o Teste Funcional Sistemático Estendido, ambos derivados dos dois critérios de teste mais abordados: Análise do Valor Limite e Particionamento em Classes de Equivalência.

Tabela 4.1: *Critérios, técnicas e abordagens de teste explorados pelos estudos analisados*

Referência	Critérios/Técnicas e Abordagens de Teste
(MYERS, 1978)	Inspeção de Código, Teste Estrutural (sem critério específico) e Teste Funcional (sem critério específico)
(BASILI; SELBY, 1987)	Análise do valor limite, Particionamento em Classes de Equivalência, Cobertura de Comandos e Leitura de Código
(NURSIMULU; PROBERT, 1995)	Grafo de Causa e Efeito e Tabela de Decisão
(KAMSTIES; LOTT, 1995)	Análise do valor limite, Particionamento em Classes de Equivalência, Cobertura de Condições e Leitura de Código
(REID, 1997)	Análise do valor limite, Particionamento em Classes de Equivalência e Teste Aleatório
(WOOD et al., 1997)	Análise do valor limite, Particionamento em Classes de Equivalência, Cobertura de Condições e Leitura de Código
(JURISTO; VEGAS, 2003)	Análise do valor limite, Particionamento em Classes de Equivalência, Cobertura de Condições e Leitura de Código
(LINKMAN et al., 2003)	Análise do valor limite, Particionamento em Classes de Equivalência e Teste Funcional Sistemático
(NEBUT et al., 2003)	Teste Baseado em Casos de Uso
(RAMACHANDRAN, 2003)	Análise do Valor Limite
(CAI et al., 2005)	Particionamento Dinâmico, Particionamento Aleatório e Teste Aleatório
(JONES, 2005)	Tabela de Decisão e Particionamento em Classes de Equivalência
(MURNANE et al., 2005)	Análise do Valor Limite e Particionamento em Classes de Equivalência
(GUTIERREZ et al., 2006)	Teste Funcional (sem critério específico)
(HIERONS, 2006)	Análise do Valor Limite e Particionamento em Classes de Equivalência
(NEBUT et al., 2006)	Teste Baseado em Casos de Uso
(ROUBTSOV; HECK, 2006)	Teste Baseado em Casos de Uso
(SEO; CHOI, 2006)	Teste Baseado em Casos de Uso, Teste Baseado em Casos de uso Estendidos, Teste a partir de Requisitos Formalizados com OCL, Teste a partir de Requisitos Convertidos em Objetc-Z e Teste a partir de Diagrama de Colaboração
(ZIELCZYNSKI, 2006)	Teste Baseado em Casos de Uso
(GUTIERREZ et al., 2008)	Teste Baseado em Casos de Uso
(NOIKAJANA; SUWANNASART, 2008)	Tabela de Decisão
(VIJ; FENG, 2008)	Análise do Valor Limite
(SRIVASTAVA et al., 2009)	Grafo de Causa e Efeito e Tabela de Decisão
(VALLESPIR; HERBERT, 2009)	Análise do Valor Limite, Particionamento em Classes de Equivalência, Inspeção de Área de Trabalho, Tabela de Decisão, Caminho Linear Independente e Cobertura de Múltiplas Condições
(SHARMA; CHANDRA, 2010)	Grafo de Causa e Efeito, Tabela de Decisão, Análise do Valor Limite e Particionamento em Classes de Equivalência
(ESCALONA et al., 2011)	Teste Funcional (sem critério específico)
(VIDAL, 2011)	Teste Funcional Sistemático e Teste Funcional Sistemático Estendido

Tabela 4.2: *Critérios, técnicas e abordagens de teste explorados pelos estudos analisados e que são do interesse desta revisão sistemática*

Critério/Técnica	Quantidade de Estudos
Análise do valor limite	12
Particionamento em classes de equivalência	11
Tabela de Decisão	6
Teste Baseado em Casos de Uso	6
Grafo de Causa e Efeito	3
Teste Funcional Sistemático	2
Particionamento Dinâmico	1
Teste Baseado em Casos de uso estendidos	1
Teste Funcional Sistemático Estendido	1
Total	43

4.1.4 Automação do teste

Dos 27 estudos primários analisados, um terço deles (9 estudos) descrevem ferramentas de suporte à automatização da utilização dos critérios/técnicas de teste explorados:

1. quatro estudos que abordam Teste Baseado em Casos de Uso: (NEBUT et al., 2003), (NEBUT et al., 2006), (ZIELCZYNSKI, 2006) e (GUTIERREZ et al., 2008);
2. dois estudos que abordam Análise do Valor Limite, sem que o foco seja na definição das partições: (RAMACHANDRAN, 2003), (VIJ; FENG, 2008);
3. um estudo que aborda Tabela de Decisão e Particionamento em Classes de Equivalência: (JONES, 2005);
4. um estudo que aborda Tabela de Decisão, isoladamente (NOIKAJANA; SUWANNASART, 2008);
5. um estudo que que aborda Tabela de Decisão, Análise do Valor Limite e Particionamento em Classes de Equivalência: (SHARMA; CHANDRA, 2010).

É possível observar que, de acordo com a quantidade de citações nos estudos que apresentam ferramentas de suporte para automatização, Teste Baseado em Casos de Uso é o mais propenso à automatização, aparecendo individualmente em 4 dos 9 estudos referenciados. Análise do Valor Limite e Tabela de Decisão aparecem com 3 citações individuais ou conjuntamente com outros critérios. Por fim, Particionamento em Classes de Equivalência aparece em dois dos 9 estudos.

4.1.5 Utilização conjunta de critérios/técnicas

A utilização conjunta de critérios/técnicas denota a aplicação complementar de critérios/técnicas de teste, onde critérios/técnicas são empregados em conjunto para a redução de custo e/ou aumento da eficácia do teste. Nesta perspectiva, os critérios de teste Particionamento em Classes de Equivalência e Análise do Valor Limite foram utilizados em conjunto em praticamente todos os estudos que os exploram. Os estudos Ramachandran (2003) e Vij e Feng (2008) abordam Análise do Valor Limite sem um foco direcionado à definição das partições, requeridas para a aplicação deste critério de teste funcional. Jones (2005) aborda o critério Particionamento em Classes de Equivalência juntamente com o critério Tabela de Decisão.

Em Linkman et al. (2003) estes critérios foram combinados para a proposição do critério Teste Funcional Sistemático (TFS), o que resultou numa geração de casos de teste com maior eficácia em relação à utilização isolada dos dois critérios: usando a análise de mutantes como uma medida de eficácia, o critério TFS alcançou 100% de mutantes não equivalentes, enquanto se obteve escores significativamente menores com os outros critérios/técnicas.

Vidal (2011) utilizou o TFS para a proposição do critério Teste Funcional Sistemático Estendido (TFSE), que representa uma evolução do TFS em relação à sua capacidade de cobertura de tipos de dados, tal como a inclusão dos tipos data e hora. Podemos observar que: (i) os critérios TFS e TFSE aumentam o número de casos de teste em relação aos critérios Particionamento em Classes de Equivalência e Análise do Valor Limite; e (ii) os critérios TFS e TFSE incluem critérios Particionamento em Classes de Equivalência e Análise do Valor Limite, significando que um conjunto de casos de teste que satisfaz os dois primeiros também satisfaz os dois últimos.

Jones (2005) utiliza conjuntamente Tabela de Decisão e Particionamento em Classes de Equivalência, como uma nova forma de empregar o critério de cobertura baseado em tabela de decisão proposto por Binder (2000), que foca na cobertura dos elementos de decisão, como condições e combinações de condições. Neste estudo, estes elementos condicionais são representados pelas regras definidas na tabela de decisão, considerando que cada regra particiona a função testada em classes de equivalência. A medida de cobertura é dada pela divisão da quantidade de regras testadas dividido pela quantidade de regras constantes da tabela de decisão.

Os critérios Grafo de Causa e Efeito e Tabela de Decisão foram utilizados em conjunto nos estudos de:

1. Nursimulu e Probert (1995) propõem uma nova abordagem denominada *BPST - Basic Path Sensitization Technique* para a geração da Tabela de Decisão a partir do Grafo de Causa e Efeito. Esta nova abordagem representa uma evolução da apresentada por Myers (1979). Apresenta uma maior coerência sintática e semântica, sendo coerência sintática expressa na relação entre a Tabela de Decisão e o Grafo de Causa e Efeito. E coerência semântica na relação entre o Grafo de Causa e Efeito e a especificação de requisitos.
2. Srivastava et al. (2009) também propõem um novo algoritmo para a geração da Tabela de Decisão a partir do Grafo de Causa e Efeito, buscando solucionar problemas constantes de abordagens anteriores, dentre as quais, as apresentadas em Nursimulu e Probert (1995) e Mathur (2008). O novo algoritmo gera todas as possíveis combinações entre causas e efeitos com uma complexidade $((O(n^2)))$, em relação ao trabalho descrito por Mathur (2008) que não gera todas as possíveis combinações e ainda tem uma complexidade $((O(n^3)))$, representa uma evolução significativa.
3. Sharma e Chandra (2010) combina Tabela de Decisão e Grafo de Causa e Efeito com Particionamento em Classes de Equivalência e Análise do Valor Limite para a construção de um *framework* genérico para a automatização da geração de casos de testes a partir da Tabela de Decisão. O processo de automatização objetiva de diminuir a quantidade de casos de teste gerados, sem prejuízo da cobertura. A aplicação deste *framework* resulta num conjunto de casos de teste mínimo, completo e sem redundancia.

Por fim, Seo e Choi (2006) recomendam a utilização conjunta do Teste Baseado em Casos de Uso Estendidos e Teste Derivado de Requisitos Formalizados com a Linguagem OCL, pois estes apresentaram o melhor resultado nos dois experimentos realizados. Teste Baseado em Casos de Uso Estendidos apresentou cobertura de 84% e 81% nos experimentos I e II respectivamente, ao passo que Teste Derivado de Requisitos Formalizados com a Linguagem OCL apresentou cobertura de 74% e 66%, respectivamente.

4.2 Questão Primária: Que comparações têm sido realizadas entre os critérios/técnicas de teste funcional?

O objetivo original desta questão de pesquisa é a identificação de estudos primários que efetuem comparações entre critérios/técnicas de teste funcional, a partir

de qualquer perspectiva. A resposta a esta questão ficou prejudicada, tendo em vista a quase inexistência de estudos primários com esse objetivo. Dentre os estudos analisados, apenas Seo e Choi (2006) e Vallespir e Herbert (2009) executam tais comparações: o primeiro compara critérios de teste aplicáveis a sistemas desenvolvidos a partir do paradigma da orientação a objetos; e o segundo utiliza conjuntamente Análise do Valor Limite e Particionamento em Classes de Equivalência (abordado somente como Particionamento por Equivalência) comparando-os com outros critérios/técnicas de teste, dentre eles, Tabela de Decisão.

Considerando que a quantidade de apenas dois estudos é pouco representativa para esta questão de pesquisa, foi decidido pela ampliação do escopo do critério de inclusão $\mathbf{CI_3}$, tornando-o sensível às técnicas e critérios de teste estrutural, o que possibilitou a adição de estudos primários que comparam técnicas/critérios funcionais com outras técnicas/critérios não funcionais. Com esse objetivo, foram acrescentados nove estudos aos dois anteriores, totalizando em 11 os selecionados para a questão de pesquisa primária, listados na Tabela 4.3, juntamente com os critérios/técnicas comparados.

Tabela 4.3: *Critérios, técnicas e abordagens de teste comparados nos estudos analisados*

Referência	Critérios, Técnicas e Abordagens de Teste Comparados
(MYERS, 1978)	Inspeção de Código, Teste Estrutural (sem critério específico) e Teste Funcional (sem critério específico)
(BASILI; SELBY, 1987)	Análise do valor limite, Particionamento em Classes de Equivalência, Cobertura de Comandos e Leitura de Código
(KAMSTIES; LOTT, 1995)	Análise do valor limite, Particionamento em Classes de Equivalência, Cobertura de Condições e Leitura de Código
(REID, 1997)	Análise do valor limite, Particionamento em Classes de Equivalência e Teste Aleatório
(WOOD et al., 1997)	Análise do valor limite, Particionamento em Classes de Equivalência, Cobertura de Condições e Leitura de Código
(JURISTO; VEGAS, 2003)	Análise do valor limite, Particionamento em Classes de Equivalência, Cobertura de Condições e Leitura de Código
(CAI et al., 2005)	Particionamento Dinâmico, Particionamento Aleatório e Teste Aleatório
(HIERONS, 2006)	Análise do Valor Limite e Particionamento em Classes de Equivalência
(SEO; CHOI, 2006)	Teste Baseado em Casos de Uso, Teste Baseado em Casos de uso Estendidos, Teste a Partir de Requisitos Formalizados com OCL, Teste a Partir de Requisitos Convertidos em Objetc-Z e Teste a Partir de Diagrama de Colaboração
(VALLESPIR; HERBERT, 2009)	Análise do Valor Limite, Particionamento em Classes de Equivalência, Inspeção de Área de Trabalho, Tabela de Decisão, Caminho Linear Independente e Cobertura de Múltiplas Condições
(SHARMA; CHANDRA, 2010)	Análise do Valor Limite, Particionamento em Classes de Equivalência, Grafo de Causa e Efeito e Tabela de Decisão

A Tabela 4.4 destaca quais aspectos são comparados em cada um destes estudos; a primeira e a segunda colunas representam um número identificador sequencial e a referência em si pertinentes aos estudos, respectivamente; a última coluna destaca os aspectos de comparação entre critérios/técnicas. É caracterizado como **Aspecto de Comparação** o que pode ser quantificado ou classificado durante as comparações, o que Juristo e Vegas (2003) denominou como variável de resposta. A maioria dos estudos trabalha com o aspecto eficácia (seis dos onze), referindo-se à quantidade de defeitos detectados. Outros aspectos identificados são eficiência, custo, probabilidade, cobertura e tipo de defeito.

Tabela 4.4: *Aspectos de comparação entre critérios/técnicas*

id	Referência	Aspectos de comparação
1	(MYERS, 1978)	**Eficácia**: quantidade de defeitos detectados e **custo**: tempo e esforço empreendido
2	(BASILI; SELBY, 1987)	**Eficácia**: quantidade de defeitos detectados, **eficiência**: eficácia / tempo e **classe de defeitos detectados**
3	(KAMSTIES; LOTT, 1995)	a) **Eficácia**: em termos da quantidade de falhas detectadas e defeitos isolados. **Eficácia**: em termos da quantidade de defeitos isolados por tipo e b) **custo**: tempo de detecção e isolamento e c) **eficiência**: eficácia / custo
4	(REID, 1997)	**Probabilidade** de detecção de defeitos.
5	(WOOD et al., 1997)	**Eficácia**: quantidade de falhas observadas e defeitos isolados e **eficiência**: eficácia dividida pelo tempo de detecção
6	(JURISTO; VEGAS, 2003)	**Eficácia**: quantidade de testadores que detectam um dado defeito, presente nos programas
7	(CAI et al., 2005)	**Custo**: quantidade de casos de testes selecionados
8	(HIERONS, 2006)	**Tipo do defeito**
9	(SEO; CHOI, 2006)	**Cobertura**: objetos instanciados, métodos executados
10	(VALLESPIR; HERBERT, 2009)	**Eficácia**: quantidade de defeitos detectados, **custo**: tempo de execução de cada técnica de teste e **eficiência**: eficácia / custo
11	(SHARMA; CHANDRA, 2010)	**Custo**: quantidade de casos de testes gerados.

As Tabelas 4.5 e 4.6 buscam quantificar os estudos considerados com respeito à Quantidade de Programas, Tamanho dos Programas (LOC), Quantidade de Defeitos, Linguagem e Experiência do Testador. A ausência de alguns estudos nestas tabelas é justificada pela impossibilidade de coleta de dados.

Dentre os critérios/técnicas classificados como de interesse desta revisão sistemática, Vallespir e Herbert (2009) conclui que em relação aos três aspectos de comparação, quantidade de defeitos, tempo de detecção e eficiência (quantidade / tempo), Particionamento por Equivalência obteve melhores resultados que Tabela de Decisão. Seo e Choi (2006) conclui que o Teste Baseado em Casos de Uso Estendidos e Teste Derivado dos Requisitos Formalizados com a Linguagem OCL são os mais efetivos, inclusive sugere a utilização conjunta destes dois tipos de teste funcional.

Myers (1978), Basili e Selby (1987), Kamsties e Lott (1995), Wood et al. (1997) e Juristo e Vegas (2003) foram unânimes em afirmar que em aspectos gerais os critérios de teste funcional Análise do Valor Limite e Particionamento em Classes de Equivalência apresentaram melhores resultados (mais defeitos detectados em menos

Tabela 4.5: *Características dos programas utilizados nas comparações entre critérios/técnicas*

Características	1	2	3	4	5	6	7	9	10
Quantidade de Programas	1	4	3	1	3	3	2	2	2
Tamanho dos Programas (LOC)	63 Comandos	169, 145, 147 e 365	10 a 30	20.000	10 a 30	200	Não especificado e 3559	Não especificado	20
Quantidade de Defeitos	15	34 ao todo	não especificado	não especificado	Não especificado	7 em cada programa	66 ao todo	Não especificado	13
Linguagem	PL/I	Fortran e Simpl-T	C	ADA	C	C	C++	Java	Java

Tabela 4.6: *Características dos testadores nas comparações entre critérios/técnicas*

Características	1	2	3	5	6	10
Quantidade	59	74	50	47	46	17
Experience	49 programadores/-testadores experientes e 10 iniciantes	8 experientes, 24 intermediários e 42 inexperientes	Considerado somente um nível	2 anos de programação	Considerado somente um nível	Estudantes do 4º período de Engenharia da Computação
Nível de Experiência	Todos foram estudantes do Instituto de Pesquisas de Sistemas da IBM	Estudantes da Universidade de Maryland, Programadores profissionais da NASA e da *Sciences Corporation*	Estudantes do 3º e 4º período de graduação	Estudantes da Universidade de Strathclyde	Estudantes do 5º período de graduação	Universidade de La República

tempo) em relação aos outros com que foram comparados. Porém, quase todos também são unânimes em afirmar que a efetividade dos resultados observados são dependentes do tipo do programa, da experiência do testador e do tipo do defeito detectado.

Wood et al. (1997) inicia sua replicação a Basili e Selby (1987) e Kamsties e Lott (1995) observando que até aquela data, 1997, (i) não existe evidência consistente de que uma técnica de detecção de defeitos seja mais forte que outra, pelo contrário, as evidências atuais sugerem que cada técnica possua seus próprios méritos; (ii) a evidência atual sugere que as técnicas de teste funcional, estrutural e leitura de código são complementares ao invés de alternativas e como resultado devem ser utilizadas em combinação.

Em 2009, Juristo et al. (2009) observam-se praticamente as mesmas conclusões de 12 anos atrás em relação a Wood et al. (1997). Os *surveys* realizados por Gutierrez et al. (2006) e Escalona et al. (2011) também não apresentam resultados conclusivos, sugerindo novas pesquisas na área.

Em síntese, os aspectos de comparação pertinentes à questão de pesquisa foram apresentados para os estudos selecionados. Contudo, os resultados obtidos pela aplicação desses aspectos não são definitivos, levando em consideração duas questões principais: a) os programas testados são muito pequenos e simples e b) os defeitos são semeados pelo testador. Os autores decretam seus resultados como mais um passo contributivo à consolidação do conhecimento sobre técnicas/critérios de teste. Neste sentido, é possível analisar os resultados não como conclusões, mas como tendências, onde há uma lacuna para a generalização dos mesmos.

4.3 Questão Secundária 1: Qual o cenário para a aplicação de cada critério/técnica de teste funcional?

Os critérios/técnicas de testes explorados pelos estudos primários foram descritos na Tabela 4.1. Nesta seção serão abordados os cenários em que estes critérios/técnicas são aplicados. Foram identificados 13 cenários, conforme descrito abaixo:

1. **Sistema de informação comercial crítico**: englobam sistemas críticos em termos de segurança de acesso, tempo de resposta, robustez, etc. Este cenário é abordado em Roubtsov e Heck (2006) e Vidal (2011);
2. **Sistema de aviação embarcado crítico**: *Thalles Airbone Systems* sistemas utilizados em aviões militares franceses (*Rafalle* e *Mirage*). Este cenário é abordado em Nebut et al. (2006);
3. **Sistemas comerciais embarcados (componentes)**: Este cenário é identificado em Ramachandran (2003), onde exemplifica a geração de casos de testes para componentes eletrônicos móveis;
4. **Sistemas Financeiros Críticos**: este cenário é abordado em Seo e Choi (2006), onde exemplifica a geração de casos de testes para a funcionalidade de um saque em um terminal bancário;
5. **Sistema de informação comercial em geral**: que engloba diversos sistemas citados nos estudos, como por exemplo: sistemas de pagamento de salários, sistemas de vendas em livraria, sistemas de transações comerciais em geral,

etc. Este cenário é abordado em Hierons (2006), Juristo e Vegas (2003), Jones (2005) e Sharma e Chandra (2010);

6. **Sistema operacional de aviões**: abordado em Reid (1997), onde se exemplifica a geração de casos de teste para um sistema operacional de avião, escrito em linguagem ADA e contendo aproximadamente 20.000 linhas de código.
7. **Sistema de gerenciamento estratégico**: abordado em Vidal (2011), onde se descreve a geração de casos de testes para o sistema "EPA - Estratégia para Ação";
8. **Utilitário de sistema operacional**: Programa *Cal*, um calendário disponível no Sistema Operacional Unix e no Linux. Este cenário é abordado em Linkman et al. (2003);
9. **Controle espacial**: É abordado em Cai et al. (2005), sem maiores detalhes do programa em si;
10. ***Web Services***: abordado em Noikajana e Suwannasart (2008), onde exemplifica a geração de casos de testes a partir dos documentos descritivos do *web service*;
11. **Teleconferência**: abordado em Nebut et al. (2003), onde exemplifica a geração de casos de teste para um sistema denominado *Virtual Meeting*;
12. **Sistema *Web* crítico**: abordado em Zielczynski (2006), exemplificando a geração de casos de teste para um livraria online;
13. **Sistema didático/acadêmico**: Este cenário está presente nos estudos de: Kamsties e Lott (1995), Wood et al. (1997), Juristo e Vegas (2003), Vij e Feng (2008), Myers (1978), Vallespir e Herbert (2009), Srivastava et al. (2009), Gutierrez et al. (2008), Seo e Choi (2006), Gutierrez et al. (2008) e Nursimulu e Probert (1995), onde os sistemas foram testados em ambiente acadêmico e/ou em laboratório, independentemente dos seus reais cenários de utilização.

A Tabela 4.7 ilustra os cenários nos quais cada critério/técnica de teste é aplicado. A tabela apresenta os estudos ordenados por quantidade de estudos que os referenciam, similarmente à Tabela 4.2. É possível notar a repetição de vários cenários em vários critérios/técnicas, indicando a existência de multiplicidade (n:n - "muitos para muitos") no relacionamento entre cenários e critérios/técnicas, ou seja, os estudos não identificam exclusividade entre um cenário A e um critério/técnica B. Isto pode ser visto como um fator positivo tendo em vista a não restrição do escopo de aplicação dos critérios/técnicas, dentro dos cenários identificados.

Em relação aos cenários, existe uma predominância para aqueles que envolvem sistemas que foram testados em "ambiente didático/acadêmico", aos quais foram

Tabela 4.7: *Cenários por critério/técnica de teste*

Critério/Técnica de Teste	Cenário de Teste
Análise do valor limite	Sistema didático/acadêmico, Sistema de informação comercial não crítico, Sistema Operacional de aviões, Utilitário de sistema operacional e Sistemas Comerciais Embarcados
Particionamento em classes de equivalência	Sistema didático/acadêmico, Sistema de informação comercial não crítico, Sistema Operacional de aviões e Utilitário de sistema operacional.
Tabela de Decisão	Sistema didático/acadêmico, Sistemas de Informação comercial não crítico e *web service*.
Teste Baseado em Casos de uso	Teleconferência, Sistema de aviação embarcado crítico, Sistema de informação comercial crítico, Sistema Financeiro Crítico, Sistema *web* crítico e Sistema didático/acadêmico
Grafo de Causa e Efeito	Sistema didático/acadêmico
Teste Funcional Sistemático Estendido	Sistema de Gerenciamento estratégico e Sistema de Informação Comercial Crítico
Particionamento Dinâmico	Controle aéreo espacial.
Teste Baseado em Casos de uso estendidos	Sistema Financeiro Crítico
Teste Funcional Sistemático	Utilitário de Sistema Operacional.

aplicados seis critérios/técnicas de teste; em segundo lugar, estão os "sistemas de informação comercial não crítico", em que quatro critérios/técnicas foram empregados. Isto decorre do fato de que a maioria (70,38%) dos estudos analisados foram desenvolvidos dentro de um ambiente acadêmico ou em laboratório, conforme descrito nas Características dos Estudos (Subseção 4.5.2, página 73). Apesar da predominância da perspectiva acadêmico/experimental, também foi observada a aplicação dos critérios/técnicas em cenários críticos de segurança, tempo de resposta, robustez, em ambiente reais de utilização, como pode ser visto em Nebut et al. (2006), Roubtsov e Heck (2006), Noikajana e Suwannasart (2008); esses cenários envolvem software embarcado em aviões militares, teste de *web services*, sistemas de administração de vendas de bilhetes de passagens para sistemas de transportes interligados em grandes regiões metropolitanas e teste de componentes eletrônicos (dispositivos móveis, celulares, controles remotos, televisores, etc.).

Em relação aos critérios/técnicas de teste, observa-se a predominância da aplicação do Teste Baseado em Casos de Uso em cenários que envolvam sistemas críticos (cinco dentre seis cenários). Para o Teste Funcional Sistemático Estendido e Teste Aleatório também foram identificados somente cenários que envolvem sistemas estratégicos ou críticos. O critério Grafo de Causa e Efeito foi usado apenas no cenário didático/acadêmico. Os demais critérios/técnicas foram aplicados predominantemente em cenários didático/acadêmico e em cenários que envolvam sistemas não críticos.

É possível observar a heterogeneidade de cenários em que são empregados os critérios/técnicas mais explorados nos estudos primários, conforme demonstrado nas cinco primeiras linhas da Tabela 4.7.

A identificação do Teste Baseado em Casos de Uso, como o mais utilizado em sistemas críticos, aparentemente representa uma contradição aos problemas inerentes à utilização desta técnica de teste funcional. Estes problemas constam do final da Subseção 2.9, na Página 44. No entanto, todos os estudos primários que abordam o Teste Baseado em Casos de Uso, definem uma etapa de formalização dos requisitos funcionais, para então, a partir daí, torná-los aptos à geração dos casos de teste, o que elimina esta potencial contradição.

4.4 Questão Secundária 2: Que critérios/técnicas de teste funcional têm sido aplicados para avaliar roteiros (especificações) de teste?

Objetivamente não foi detectada, dentre os estudos analisados, nenhuma utilização direta de critérios/técnicas de teste para avaliação de especificações de testes. Por outro lado foram identificadas abordagens que auxiliam na avaliação e melhoria de especificações de requisitos, para posterior geração dos casos de testes. Por exemplo, Nursimulu e Probert (1995) e Srivastava et al. (2009) utilizam os critérios de teste Grafo de Causa e Efeito e Tabela de Decisão sobre as especificações de requisitos. Nebut et al. (2003) e Nebut et al. (2006) utilizam Teste Baseado em Caso de Uso, abordando o projeto por contrato, onde a primeira etapa da abordagem passa pela validação e formalização dos requisitos, para automatizar a geração de casos de teste. Jones (2005) utiliza Tabela de Decisão como uma linguagem de especificação de requisitos.

4.5 Características dos Estudos

A análise descrita nesta seção refere-se ao Item 5 da Tabela 3.2, presente no planejamento desta revisão sistemática.

4.5.1 Tipo de Estudo Experimental

Na Tabela 4.8 são apresentadas informações quantitativas sobre o tipo de estudo experimental empregado nos trabalhos selecionados, esta classificação foi realizada segundo a terminologia definida por Dybå e Dingsøyr (2008). Alguns resultados são: (i) 33,33% dos estudos analisados são de natureza experimental, (ii) estudo de caso foi apresentado em 22,22% do estudos primários, (iii) 18,52% foi o percentual alcançado pelos estudos cuja validação da abordagem foi efetuada através de simulação, mesmo percentual dos estudos que desenvolveram apenas análise teórica, sem a apresentação de resultados empíricos.

Tabela 4.8: *Tipo de Estudo Experimental*

Estudo Experimental	IEEE	ACM	Google Acad	Outros	Quant	%
Experimento	5	3	1	0	9	33,33
Estudo de Caso	4	0	0	2	6	22,22
Simulação	3	0	2	0	5	18,52
Análise Teórica	2	3	0	0	5	18,52
Survey	0	1	1	0	2	7,41
Total	14	7	4	2	27	100
Média	3,25	2,33	1,25	2,00	5,40	
Desvio Padrão	1,92	1,52	0,84	0,89	2,51	

4.5.2 Escopo de Atuação dos Estudos

Na Tabela 4.9 são apresentadas informações quantitativas sobre o escopo de atuação dos estudos selecionados. Ao examinar a tabela, constata-se uma predominâcia de estudos realizados em ambiente academico ou em laboratório (70.38%).

Tabela 4.9: *Escopo de atuação dos estudos*

Escopo	IEEE	ACM	Google Acad.	Outros	Quant.	%
Indústria	7	0	1	0	8	29,62
Academia/Laboratório	7	7	3	2	19	70,38
Total	14	7	4	2	27	100
Média	7	7	2	2	13,5	
Desvio Padrão	0	0	1,41	0	7,78	

4.5.3 Dígrafo de Citação Interna

Para os estudos primários que referenciam um ou mais estudos pertencentes ao conjunto selecionado pela revisão sistemática, é possível construir um grafo direcionado (dígrafo) e identificar seus respectivos graus de entrada e de saída. Na Tabela 4.10 são apresentadas informações referentes ao respectivo dígrafo, já na

Figura 4.1 é apresentada a representação gráfica do grafo direcionado, construído com o auxílio da ferramenta case JUDE, ChangeVision (2011).

O grau de entrada neste dígrafo corresponde ao total de vezes em que o referido estudo foi citado. Esta métrica revela os estudos mais referenciados no contexto da revisão sistemática. Na Figura 4.1, constata-se que os estudos EP1, EP2, EP4 e EP6 são os mais citados, um desses estudos (EP2) consta da lista dos artigos de controle definidos no planejamento da revisão sistemática. Os estudos EP1 e EP2 cronologicamente são os primeiros estudos que exploram a comparação entre técnicas de teste funcional, estrutural e leitura de código, servindo de base para os demais estudos que abordam tais comparações. Os estudos EP7, EP24 e EP26 possuem o maior grau de saída, sendo que EP7 replica os estudos EP2, EP4 e EP6 e referencia EP1. EP24 apresenta uma retrospectiva dos estudos que abordam comparação de técnicas de teste. EP26, sendo um *survey* também referencia estes estudos comparativos.

Tabela 4.10: *Dígrafo dos estudos primários selecionados*

Identificador	Estudo	Grau de Entrada	Grau de Saída
EP1	(MYERS, 1978)	6	0
EP2	(BASILI; SELBY, 1987)	5	1
EP3	(NURSIMULU; PROBERT, 1995)	1	0
EP4	(KAMSTIES; LOTT, 1995)	3	2
EP5	(REID, 1997)	0	2
EP6	(WOOD et al., 1997)	2	3
EP7	(JURISTO; VEGAS, 2003)	0	4
EP8	(LINKMAN et al., 2003)	1	0
EP9	(NEBUT et al., 2003)	3	0
EP10	(RAMACHANDRAN, 2003)	0	0
EP11	(CAI et al., 2005)	0	0
EP12	(JONES, 2005)	0	0
EP13	(MURNANE et al., 2005)	0	0
EP14	(GUTIERREZ et al., 2006)	1	1
EP15	(HIERONS, 2006)	0	0
EP16	(NEBUT et al., 2006)	2	1
EP17	(ROUBTSOV; HECK, 2006)	1	0
EP18	(SEO; CHOI, 2006)	0	0
EP19	(ZIELCZYNSKI, 2006)	1	0
EP20	(GUTIERREZ et al., 2008)	1	2
EP21	(NOIKAJANA; SUWANNASART, 2008)	0	0
EP22	(VIJ; FENG, 2008)	0	0
EP23	(SRIVASTAVA et al., 2009)	1	1
EP24	(VALLESPIR; HERBERT, 2009)	0	4
EP25	(SHARMA; CHANDRA, 2010)	0	1
EP26	(ESCALONA et al., 2011)	0	4
EP27	(VIDAL, 2011)	0	1

Na Figura 4.1 é possível observar a presença de algumas regiões de concentração de citações entre os estudos primários, por exemplo é possível identificar a região de citações onde o EP1 concentra o maior grau de entrada, isto em virtude de ser um dos primeiros estudos publicados abordando comparações entre técnicas de teste, conforme observado anteriormente. Outra região possível de ser identificada é aquela

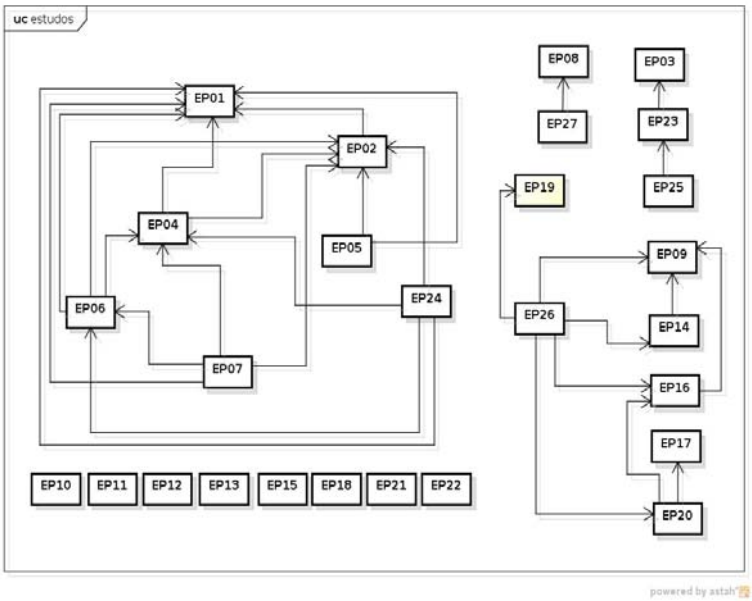

Figura 4.1: *Grafo direcionado das citações entre os estudos*

.

em que consta o EP26, demonstrando que este estudo possui o maior grau de saída, pois trata-se de um *survey* com referência a vários outros estudos primários. Por fim, pode-se ainda observar a região em que estão presentes os estudos EP25, EP23 e EP03, o fator comum de união entre estes três estudos primários são os critérios de teste funcional Tabela de Decisão e Grafo de Causa e Efeito, abordados por todos eles.

4.6 Força das Evidências

Com relação às *características dos estudos*, dois terços são de natureza observacional, e um terço corresponde a experimentos, conforme Tabela 4.8. Desta forma, segundo as definições em GRADE, apresentadas na Subseção 3.1.9, na Página 3.1.9, a força das evidências desta Revisão Sistemática, relativamente às *características dos estudos*, é considerada **baixa** (ALI et al., 2010).

Com relação à *qualidade dos estudos*, as abordagens de análise de dados nos respectivos estudos foram explicadas de forma moderada: questões como viés potencial, credibilidade e limitações dos estudos (Questões nove, dez e onze, respectivamente), descritas na Subseção 3.1.7.3 e instanciadas na Tabela 4.11. Somente em seis dos vinte e sete estudos analisados, houve análise crítica do pesquisador em relação ao seu papel desempenhado durante a pesquisa. Houve discussão a respeito da credibilidade dos resultados obtidos pelo estudo em 85.19% deles. Em relação às limitações

dos estudos, esta discussão esteve presente em 88.89% destes estudos. Baseado nestes resultados, os estudos apresentam evidências **moderadas**, em relação à *qualidade* dos mesmos.

Com relação ao critério *consistência*, foram identicadas similaridades entre os estudos, pois 100.00% abordam teste funcional, seja através de um ou vários critérios/técnicas, na forma do emprego individual ou em conjunto, num determinado cenário ou em experimentos comparativos com outros critérios de outras abordagens de teste. Em virtude disso, entende-se que a força das evidências no que se refere à *consistência* pode ser classificada como **alta**.

Com relação ao critério *objetividade* (*directness*), que consiste em avaliar se as pessoas envolvidas, as intervenções e os resultados dos estudos estão de acordo com a área de interesse, constatou-se que a maioria dos estudos (70,38%) foi no contexto da academia/laboratório, conforme apresentado na Tabela 4.9; mesmo os experimentos, a maioria foi executada em ambiente acadêmico. Com relação à intervenção, observou-se a predominância de estudos abordando critérios e técnicas de teste funcional, conforme definido no planejamento. Com relação aos resultados obtidos, pelo fato que a maioria dos estudos serem de natureza observacional, tais estudos requerem mais validação empírica com respeito a aplicações reais. Então, a força das evidências no que se refere à *objetividade* pode ser considerada entre **moderada** e **baixa**.

Combinando os quatros elementos para se determinar a força das evidências, pode-se afirmar que a força das evidências para esta Revisão Sistemática pode ser classificada como **moderada**, considerando também a ausência de resposta objetiva para uma das três questões de pesquisa. Portanto, pesquisas futuras são susceptíveis que provoquem um impacto importante sobre a confiança na estimativa do efeito da revisão sistemática.

Na Subseção 3.1.7.1, o planejamento da revisão sistemática prevê um conjunto de questões para a avaliação da qualidade dos estudos primários. Na Tabela 4.11 é apresentado a avaliação individual de cada estudo primário sobre tais questões. As oito primeiras questões dizem respeito à qualidade e o rigor dos estudos, já as Questões 9 a 11 dizem respeito à credibilidade das evidências e limitações apresentadas no estudo primário.

Na Tabela 4.12 é apresentado de forma agrupada o resultado da avaliação das oito primeiras questões do questionário de avaliação da qualidade dos estudos. Sendo que foram classificados segundo a pontuação total obtida nas respectivas questões.

Tabela 4.11: *Avaliação da qualidade dos estudos primários*

EP	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	SUB	Q9	Q10	Q11	SUB	TOT
EP1	1	1	1	1	1	1	1	1	8	1	0	1	2	10
EP2	1	1	1	1	1	1	1	1	8	1	1	1	3	11
EP3	1	1	1	0	0	0	0	1	4	0	1	1	2	6
EP4	1	1	1	1	1	1	1	1	8	1	1	1	3	11
EP5	1	1	1	1	1	1	1	1	8	1	0	1	2	10
EP6	1	1	1	1	1	1	1	1	8	1	1	1	3	11
EP7	1	1	1	1	1	1	1	1	8	1	1	1	3	11
EP8	1	0,5	0	1	1	1	1	1	6,5	0	1	0	1	7,5
EP9	1	0	0	0	1	1	1	1	5	0	1	1	2	7
EP10	1	1	1	0	0	0	0	1	4	0	1	1	2	6
EP11	1	0,5	0,5	0,5	1	0,5	1	1	6	0	1	1	2	8
EP12	1	0,5	0	0	0	1	1	1	4,5	0	1	1	2	6,5
EP13	1	1	1	0	0	0	0	1	4	0	1	1	2	6
EP14	1	1	0,5	1	1	1	1	1	7,5	0	1	1	2	9,5
EP15	1	1	1	0	0	0	0	1	4	0	1	0	1	5
EP16	1	1	1	1	1	1	1	1	8	0	1	1	2	10
EP17	1	1	1	1	1	1	1	1	8	0	1	1	2	10
EP18	1	0,5	0	1	1	1	0	0,5	5	0	1	1	2	7
EP19	1	1	0,5	0,5	1	0,5	1	1	6,5	0	1	1	2	8,5
EP20	1	1	1	1	1	1	1	1	8	0	1	1	2	10
EP21	1	1	1	1	1	1	1	1	8	0	1	1	2	10
EP22	1	0,5	0	1	1	1	1	1	6,5	0	1	1	2	8,5
EP23	1	1	1	0	0	0	0	1	4	0	1	1	2	6
EP24	1	1	1	1	1	1	1	1	8	0	1	1	2	10
EP25	1	0,5	0	0	1	1	1	1	5,5	0	1	1	2	7,5
EP26	1	0	1	1	1	1	0,5	1	6,5	0	1	1	2	8,5
EP27	1	1	1	1	1	0,5	1	0,5	7	0	1	1	2	9

Tabela 4.12: *Nível de qualidade da estrutura e rigor dos estudos*

Índice de Qualidade	**Fraco (0 - 4)**	**Regular (4,5 - 6,5)**	**(Bom 7 - 8)**
ACM	3	1	3
IEEE	2	5	7
Google Acad.		2	2
UFG		1	1
Total	5	9	13
Percentual	18,52	33,33	48,15
Média	1	2,25	3,25
Desvio Padrão	0,71	1,89	2,63

Na Tabela 4.13 é apresentado de forma agrupada o resultado da avaliação das três últimas questões do questionário de avaliação da qualidade dos estudos. Sendo que foram classificados segundo a pontuação total obtida nas respectivas questões.

4.7 Ameaças à Validade

Uma revisão sistemática, conforme Budgen et al. (2011), possui pelo menos duas ameaças evidentes à sua validade: (i) limitações relativas às fontes de pesquisa e (ii) a formulação das questões de pesquisa em consonância com as publicações da comunidade científica particularmente à área de conhecimento investigada. Estas ameaças também estão presentes nesta revisão sistemática. Relativamente a ameaça

Tabela 4.13: *Nível de credibilidade das evidências dos estudos*

Índice de Qualidade	Fraco (0 - 1)	Regular (1,5 - 2)	Bom (2,5 - 3)
ACM	1	4	2
IEEE	1	12	1
Google Acad.		3	1
UFG	1	1	
Total	3	20	4
Percentual	11,11	74,07	14,81
Média	1	5,00	1,33
Desvio Padrão	0	4,83	0,58

(i), somente as bases indexadas *IEEExplore* e *ACM Digital Library* foram utilizadas, o que pode ocasionar na não identificação de estudos primários relevantes às questões de pesquisa que não estejam publicados nestas duas fontes. Em relação à ameaça (ii), comparações entre técnicas e critérios de teste (objeto de investigação da questão primária) são bastantes estudadas pela comunidade científica, no entanto as questões de pesquisa secundária I e II não tiveram muitos estudos primários identificados que contemplassem a totalidade do escopo destas duas questões.

Uma terceira ameaça pode ser identificada particularmente a esta revisão sistemática, a saber, o fato de não se ter identificado uma base objetiva para comparação entre os critérios e técnicas de teste, ao invés disto, foram utilizados como critérios de comparação fatores como eficácia, custo e eficiência. No entanto foi observado que estes fatores são altamente dependentes de outros, tais como: a experiência do testador, o tipo e o tamanho do programa em teste e o tipo do defeito presente no programa, dentre outros.

A não identificação de outras revisões sistemáticas com foco de pesquisa parecido com esta, pode representar uma quarta ameaça à sua validade. Seis revisões sistemáticas foram identificadas, contudo cada uma com um foco específico, seja na particularização de uma técnica, uma abordagem de teste ou mesmo um determinado tipo de programa em teste, conforme listadas a seguir:

1. Souza et al. (2011) foca nas pesquisas sobre teste de softwares concorrentes;
2. Afzal et al. (2009) investiga o teste baseado em buscas relativamente às propriedades não funcionais dos sistemas em teste;
3. Neto et al. (2007) conduz um *survey* a respeito das abodagens de teste baseadas em modelos;
4. Brito et al. (2010) também conduzem uma revisão sistemática sobre o teste de sistemas concorrentes;
5. Shafique e Labiche (2010) foca nas pesquisas relativamente às ferramentas de suporte para a técnica de teste baseada em modelos e

6. Amar e Shabbir (2008) investiga os desafios, as técnicas e a efetvidade do teste de programas orientados a aspectos.

4.8 Considerações Finais

Esta revisão sistemática foi planejada e conduzida com o objetivo conhecer a aplicação de critérios/técnicas de teste funcional:

- Questão de Pesquisa Primária: que comparações têm sido realizadas entre os critérios/técnicas de teste funcional?
- Questão de Pesquisa Secundária 1: qual o cenário para a aplicação de cada critério/técnica de teste funcional?
- Questão de Pesquisa Secundária 2: que critérios/técnicas de teste funcional têm sido aplicados para avaliar roteiros (especificações) de teste?

Um conjunto de 27 estudos primários foram estudados extraindo-se informações relevantes de cada um para o suporte às conclusões que embasaram as respostas às questões de pesquisa.

Em relação à Questão Primária, apenas dois estudos compararam teste funcional entre si, pouco contribuindo para a consolidação do conhecimento e da prática da utilização dos critérios/técnicas funcionais. Vários outros estudos, Seção 4.2, Página 65, efetuam comparações entre os critérios/técnicas de teste funcional e outros critérios/técnicas de teste, tais como critérios de teste estrutural. Estes estudos apontam situações e cenários em que um critério/técnica se apresenta mais efetivo, concluindo que, em geral, as técnicas e critérios de teste são complementares e não concorrentes e devem ser aplicadas em conjunto para o obtenção de um resultado mais efetivo durante o processo de teste. Os resultados destas comparações foram influenciados por fatores como por exemplo a experiência do testador, o tipo e o tamanho do programa testado e o tipo de defeito presente nestes programas.

Em relação à Questão Secundária 1, foi observado que o critério de teste Análise do Valor Limite foi o mais utilizado, pois foi analisado em maior número de cenários. Vários cenários de aplicação/utilização dos critérios e técnicas de teste funcional foram identificados, dentre estes o cenário didático/acadêmico esteve presente na maior parte dos estudos analisados. O Teste Baseado em Caso de Uso foi o mais empregado em cenários críticos. Não foi identificada a exclusividade entre cenário e critério/técnica de teste. A experiência e critatividade do testador são

fundamentais para o emprego de um critério/técnica, mesmo quando a sua aplicação em determinado cenário não for recomendada.

Em relação à Questão Secundária 2, não foi possível a identificação de resposta objetiva, contudo foi detectada a utilização dos critérios de teste funcional Tabela de Decisão, Grafo de Causa e Efeito, da técnica de Teste Baseado em Casos de Uso na avaliação dos requisitos de software, visando a torná-los mais consistentes e aptos para serem testados.

Após as considerações em relação a cada uma das questões de pesquisa, uma avaliação dos estudos primários foi efetuada no sentido de estabelecer a força das evidências e definir um grau de confiabilidade nos resultados apresentados. Concluiu-se que a força das evidências para esta Revisão Sistemática pode ser classificada como moderada, considerando também a ausência de resposta objetiva para uma das três questões de pesquisa.

CAPÍTULO 5

Uma Estratégia para a Aplicação do Teste Funcional de Software

Nos Capítulos 3 e 4 foi apresentada uma revisão sistemática sobre teste funcional, onde foi possível observar:

1. os critérios/técnicas mais estudados e utilizados;
2. os pontos fortes e fracos dos critérios/técnicas;
3. algumas comparações entre estes critérios/técnicas, avaliando, dentre outros a capacidade de detecção de defeitos e o custo associado à utilização de cada um destes critérios/técnicas de teste funcional analisados;
4. a utilização conjunta de alguns destes critérios/técnicas, conforme destacado nos trabalhos de Linkman et al. (2003), Jones (2005), Sharma e Chandra (2010) e Vidal (2011);
5. a inexistência de exclusividade de cenário para utilização específica de um determinado critério de teste;
6. algumas adaptações de critérios/técnicas para a utilização em cenários para os quais normalmente não são recomendados, como por exemplo em: Vij e Feng (2008).

Foram analisados durante a revisão sistemática estudos primários que abordam a utilização conjunta de alguns critérios/técnicas de teste funcional. Por exemplo, em Linkman et al. (2003) é introduzido o critério Teste Funcional Sistemático (TFS), o qual é derivado dos critérios Análise do Valor Limite e Particionamento em Classe de Equivalência. Foi demonstrada a sua eficácia com foco na avaliação de qualidade pertinente ao escore de mutação. Contudo, se o conjunto completo de dados de teste gerado pelo TFS for muito grande, tornando seu custo de aplicação alto, vale considerar o escopo do software para definir um subconjunto dos dados de teste tal

que possa ser aplicado a sistemas não críticos. Este é o foco da estratégia proposta, a qual será apresentada mais adiante neste capítulo.

Tabela de decisão é o terceiro critério de teste funcional mais utilizado, de acordo com a revisão sistemática. Além da sua vocação natural para auxiliar na elicitação de requisitos através da análise de regras de negócio, este critério promove o teste econômico, isto é, tem na sua aplicação a capacidade de reduzir o conjunto necessário de dados de teste, ao mesmo tempo que não perde a qualidade em termos de cobertura. Estas características estão presentes neste critério de teste devido ao rigor lógico necessário à sua aplicação, possibilitando a identificação de redundâncias e dados de teste potencialmente não necessários.

Existem outras abordagens propondo a utilização conjunta de critérios. Em Sharma e Chandra (2010) a utilização dos critérios Particionamento em Classes de Equivalência, Análise do Valor Limite e Tabela de Decisão, é abordada na composição de um *framework* para automatizar o processo de geração de dado de testes. Os autores observam que com a utilização do critério Tabela de Decisão, é possível reduzir em 1,5 vezes a quantidade de dado de testes em relação ao critério Particionamento em Classes de Equivalência e em até 5 vezes em relação ao critério Análise do Valor Limite.

Um critério de teste promissor seria aquele capaz de reunir em si mesmo as vantagens dos critérios mais difundidos com a lógica e economia da tabela de decisão. Nessa perspectiva, é proposto um critério de Teste Funcional, denominado **Teste Funcional Sistemático com Aplicação de Tabela de Decisão - TFS-TD**, que visa justamente a diminuição de custo pela redução do conjunto de dados de teste adequado ao TFS, mantendo-se a qualidade do conjunto de teste, em relação à capacidade de detecção de defeitos. A proposição do TFS-TD mantém sintonia com os resultados da revisão sistemática, ao mesmo tempo que se apresenta como solução ao problema atribuído à pesquisa, descrito na Seção 1.1, na Página 20. Nas próximas seções são apresentadas a definição do TFS-TD e estudos de caso visando a sua validação.

5.1 Teste Funcional Sistemático com Tabela de Decisão - TFS-TD

Linkman et al. (2003) empregou o conjunto de diretrizes definidas para o TFS, conduzindo a geração de dados de teste eficazes com respeito à detecção de defeitos.

Os autores mencionam que tal resultado foi obtido, dentre vários fatores, devido ao tratamento dispensado à co-incidência de defeitos e aos testes das fronteiras e seus arredores em cada classe. Contudo, não evidenciam o custo da aplicação do TFS, nem comentam se os dados de teste adequados são redundantes para cobrir uma mesma classe de equivalência; por exemplo, a Seção 2.3 apresenta um exemplo em que uma classe de equivalência é exercitada por quatro dados de teste.

Na definição do TFS, a diretriz *gerar dois dados de teste por classe de equivalência* não especifica se é aplicável às classes válidas e inválidas. O critério *Particionamento em Classes de Equivalência* recomenda a derivação de somente um dado de teste por classe inválida, conforme pode ser verificado em Copeland (2003). A Tabela 5.1 sintetiza os pontos fortes e fracos do TFS, observando que esta potencial redundância de dados de teste se apresenta com maior ênfase em relação às classes inválidas.

Tabela 5.1: *Pontos fortes e fracos do TFS*

Teste Funcional Sistemático	
Pontos Fortes	Pontos Fracos
Cobertura (tipos de dados)	Não explora variáveis interdependentes
Capacidade de detecção de defeitos	Potencial redundância de dados de teste

O **Teste Funcional Sistemático com Aplicação de Tabela de Decisão - TFS-TD** é uma abordagem que busca preservar os pontos fortes do TFS, ao mesmo tempo em que explora a diminuição do custo associado à sua aplicação, ou seja, redução de dados de teste sem a perda de qualidade (quantidade de defeitos revelados) em relação ao conjunto de dados de teste adequado ao TFS.

5.1.1 Diretrizes do TFS-TD

A essência do TFS-TD é racionalizar a geração de dados de teste: (i) gerar no mínimo dois dados de teste para cobrir as classes válidas; (ii) gerar no mínimo um dado de teste para cobrir as classes inválidas; e (iii) computar os valores limites para a cobertura de classes.

Um aspecto importante é que as partições inválidas são caracterizadas com um valor inválido para uma variável de entrada em combinação com valores válidos para as demais variáveis. Por exemplo, se um programa possuir **n** variáveis de entrada, dados de teste que cobrem partições inválidas possuem **um** valor inválido e **n-1** valores válidos. Do ponto de vista do teste de software, a descoberta da presença de um defeito pode ocorrer isoladamente ou em conjunto com outros defeitos. O enfoque adotado usa a seguinte premissa: o conjunto de dados de teste que cobre

todas as classes de equivalência com um único valor inválido para as variáveis de entrada também cobre as classes que requerem a combinação de valores inválidos para essas variáveis.

Para ilustar esse conceito, considere a Tabela 5.2, extraída de Linkman et al. (2003), a qual apresenta classes de equivalência para duas variáveis: *Mês* e *Ano*. Os valores válidos para as variáveis *Mês* e *Ano* são definidos, respectivamente, por $1 \leq Mês \leq 12$ e $1 \leq Ano \leq 9999$. As classes que possuem um único valor inválido são: I(10), I(14), I(18), I(19), I(20) e I(21). As classes definidas pela combinação de valores inválidos são: I(7), I(8), I(9), I(11), I(12), I(13), I(15), I(16) e I(17).

Tabela 5.2: *Combinação mês e ano*

Mês e Ano	Ano não inteiro	Ano < 1	Ano > 9999	$1 \leq$ Ano ≤ 9999
Mês não inteiro	I(7)	I(8)	I(9)	I(10)
Mês < 1	I(11)	I(12)	I(13)	I(14)
Mês > 12	I(15)	I(16)	I(17)	I(18)
$1 \leq$ Mês ≤ 12	I(19)	I(20)	I(21)	V(22)

O conjunto de diretrizes definidas para o TFS-TD é apresentado a seguir:

Diretriz Dir_1: não considerar as classes inválidas definidas pela combinação de valores inválidos; no exemplo da Tabela 5.2, não serão gerados dados de teste para as partições I(7), I(8), I(9), I(11), I(12), I(13), I(15), I(16) e I(17);

Diretriz Dir_2: não considerar as partições incluídas por outras partições; por exemplo, a Partição A inclui a Partição B quando qualquer dado de teste que cobre A também cobre B; nesse sentido, não serão gerados dados de teste para cobrir B; no exemplo da Tabela 5.2, se houver uma outra partição válida definida pelo mês 2 e qualquer ano bissexto, então a geração de dados para esta partição cobrirá também a Partição V(22);

Diretriz Dir_3: gerar pelo menos dois dados de teste por partição válida, para minimizar a co-incidência de defeitos que mascaram a manifestação de falhas; gerar pelo menos um dado de teste para partição inválida;

Diretriz Dir_4: exercitar as fronteiras de cada partição, conforme critério *Análise do Valor Limite*;

Diretriz Dir_5: no caso de partições compostas por valores discretos (conjunto finito composto por poucos elementos), exercitar cada valor existente;

Diretriz Dir_6: exercitar valores especiais, tal como o valor zero;

A Tabela 5.3 apresenta de forma comparativa as diretrizes do TFS e do TFS-TD.

Tabela 5.3: *TFS x TFS-TD*

Diretrizes	TFS	TFS-TD
Exercitar partições válidas e inválidas	SIM	SIM
Derivar pelo menos dois dados de teste por partição válida	SIM	SIM
Derivar pelo menos dois dados de teste por partição inválida	SIM	NÃO
Derivar pelo menos um dado de teste para o interior por partição	SIM	NÃO
Exercitar as fronteiras das partições	SIM	SIM
Derivar dados de teste para valores inválidos	SIM	SIM
Derivar dados de teste para valores especiais	SIM	SIM
Derivar dados de teste para tipos diferentes de dados	SIM	SIM
Derivar dados de teste para combinação de classes inválidas	SIM	NÃO
Derivar dados de teste para cada elemento de um conjunto discreto	SIM	SIM
Derivar dados de teste para as partições incluídas por outras partições	SIM	NÃO
Racionalizar o número de dados de teste para cobrir os requisitos acima	NÃO	SIM

Observar o conjunto de diretrizes Dir_1 a Dir_6 é o primeiro passo para o uso do TFS-TD. Em adição, é pertinente a formalização de um processo simples, visando à **aplicação do TFS-TD** segundo tais diretrizes.

5.1.2 Aplicação do TFS-TD

A geração de dados de teste segundo o TFS-TD requer a definição do conjunto de classes de equivalência a partir da especificação do software. Após essa definição, as etapas abaixo são sequencialmente aplicadas:

Etapa E_1: marcar as partições que se encaixam no que prescrevem às Diretrizes Dir_1 e Dir_2;

Etapa E_2: gerar um conjunto de descrições de dados de teste, conforme as Diretrizes Dir_4, Dir_5 e Dir_6, para todas as partições, excetuando-se aquelas marcadas na Etapa E_1; tais descrições buscam definir os dados de teste, mas objetivamente não há o compromisso de instanciá-los; alguns exemplos são (mês-válido, ano-válido) e (2, ano-bissexto);

Etapa E_3: empregar o recurso **Tabela de Decisão**, conforme observado na Tabela 5.4: (i) as primeiras linhas referem-se às classes de equivalência (partições) de entrada, representando as condições e as últimas linhas referem-se às partições de saída, representando as ações, (ii) o preenchimento (0 ou 1) denota que partições de entrada e de saída são cobertas por cada dado descrito, e (iii) a última coluna determina quantas descrições de dados cobrem cada partição de entrada e de saída; avaliar se o conjunto de descrições atende à Diretriz Dir_3;

Etapa E_4: quando for o caso, acrescentar novas descrições de dados de teste até que se cumpra à diretriz Dir_3; vale ressaltar que a Tabela de Decisão possibilita a descoberta de descrições de dados que cobrem mais de uma partição, reduzindo, portanto, o conjunto de dados de teste adequado;

Tabela 5.4: *Exemplo de tabela de decisão com descrições de dados de teste*

	descrição 1	descrição 2	descrição 3	...	descrição n	Σ
CE_1	0	0	0	...	1	1
CE_2	1	0	1	...	0	2
CE_3	0	0	0	...	0	0
...	...	...	...	...	...	...
CE_m	1	1	1	...	0	3
A_1	0	0	0	...	1	2
A_2	1	0	1	...	0	2
...	...	...	...	...	...	...
A_x	1	1	1	...	0	3

Etapa E_5: elaborar dados de teste para atender às descrições de dados de teste, e construir a matriz *descrições versus dados*, visando a alcançar dados de teste que atendam a mais de uma descrição, conforme pode ser observado na Tabela 5.5: (i) o preenchimento (0 ou 1) denota que descrições são atendidas por um dado de teste, (ii) a última coluna representa quantas descrições são cobertas por cada dado de teste, e (iii) para que haja racionalização de dados de teste, é desejável que a última coluna possua valores maiores do que um, pois, dessa forma, estar-se-á alcançando um número de dados de teste inferior à quantidade de descrições (m < n).

Tabela 5.5: *Exemplo de matriz descrições versus dados.*

	descrição 1	descrição 2	descrição 3	...	descrição n	Σ
DT_1	0	0	0	...	1	1
DT_2	1	1	0	...	0	2
...	...	...	...	...	...	...
DT_m	0	0	1	...	0	1
Σ	1	1	1	...	1	n

5.1.3 Exemplo de aplicação do TFS-TD

Esta seção apresenta uma prova de conceito para reforçar o entendimento do TFS-TD. Um exemplo do cálculo para a concessão de desconto sobre o prêmio anual de seguro de automóvel será utilizado. Neste sistema hipotético o desconto é concedido levando em consideração o sexo e a idade, portanto são duas as variáveis de entrada, que possuem os seguintes domínios:

- sexo = {masculino, feminino};
- idade = $\{18 \leq \text{idade} < 70\}$

O cálculo do desconto observa um conjunto de regras de negócio definidas em função da combinação das variáveis de entrada, conforme especificado na Tabela 5.6:

Tabela 5.6: *Regras para desconto seguro veículos*

Regras	Desconto
Mulher $<$ 31 anos	10.00%
Mulher $\geq$ 31 anos	30.00%
Homem $<$ 31 anos	0%
Homem $\geq$ 31 anos	15.00%

Considerando as duas variáveis de entrada, seus domínios e as regras de negócio, um conjunto de classes de equivalência é definido: a Tabela 5.7 refere-se ao número de parâmetros de entrada; a Tabela 5.8 refere-se às classes inválidas; e a Tabela 5.9 refere-se às classes válidas.

Tabela 5.7: *Seguro de veículos: classes de equivalência pertinentes ao número de parâmetros de entrada.*

Parâmetros	z = 2	z $\neq$ 2
z	V(1)	I(2)

Tabela 5.8: *Seguro de veículos: classes de equivalência inválidas.*

Sexo / Idade	idade não inteiro	idade $<$ 18	idade $\geq$ 70	18 $\leq$ idade $<$ 70
sexo $\notin$ { feminino, masculino }	I(3)	I(4)	I(5)	I(6)
sexo $\in$ { feminino, masculino }	I(7)	I(8)	I(9)	V(10)

Tabela 5.9: *Seguro de veículos: classes de equivalência válidas.*

Sexo	Idade	Desconto
feminino	18 $\leq$ idade $<$ 31	10% V(11)
feminino	31 $\leq$ idade $<$ 70	30% V(12)
masculino	18 $\leq$ idade $<$ 31	0% V(13)
masculino	31 $\leq$ idade $<$ 70	15% V(14)

Na execução da Etapa E_1, foram marcadas as partições: V(1) e V(10), pois representam partições incluídas por outras partições (Diretriz Dir_2); I(3), I(4) e I(5), pois são definidas pela combinação de valores inválidos (Diretriz Dir_1).

Na execução da Etapa E_2, foram geradas descrições de dados de teste apresentadas na Tabela 5.10.

Na execução da Etapa E_3, a tabela de decisão com o conjunto de descrições de dados de teste está disposta na Tabela 5.12. É possível verificar que todas as classes são plenamente cobertas pelas descrições de dados de teste, sendo que as classes válidas são cobertas por pelo menos duas descrições e as inválidas por pelo menos uma, conforme pode ser observado na última coluna da tabela.

Tabela 5.10: *Seguro de veículos: descrições de dados de teste.*

Id	Descrição	Id	Descrição
D_1	(feminino)	D_{11}	(feminino, 30)
D_2	(*sexo-inválido*, *idade-válida*)	D_{12}	(feminino, 31)
D_3	(*sexo-válido*, *idade-não-inteiro*)	D_{13}	(feminino, 69)
D_4	(*sexo-válido*, 0)	D_{14}	(feminino, 70)
D_5	(*sexo-válido*, 17)	D_{15}	(masculino, 17)
D_6	(*sexo-válido*, 18)	D_{16}	(masculino, 18)
D_7	(*sexo-válido*, 69)	D_{17}	(masculino, 30)
D_8	(*sexo-válido*, 70)	D_{18}	(masculino, 31)
D_9	(feminino, 17)	D_{19}	(masculino, 69)
D_{10}	(feminino, 18)	D_{20}	(masculino, 70)

Na execução da Etapa E_4, não foi detectada a necessidade de se adicionar novas descrições, conforme prescreve a Diretriz Dir_4. Vale ressaltar que as partições V(11), V(12), V(13) e V(14) possuem cobertura condicional pertinente às descrições D_6 e D_7, conforme ressaltado na Tabela 5.12. Contudo, esse aspecto não representa problema, pois tais partições já possuem duas outras descrições que as cobrem.

Na Etapa E_5, são elaborados os dados para atender às descrições de dados de teste, e construída a matriz *descrições versus dados*, conforme a apresentada na Tabela 5.13: (i) a primeira e segunda colunas apresentam, respectivamente, uma identificação e o valor dos dados de teste; (ii) as demais colunas apresentam valor 1 (um) quando um dado de teste atende à uma descrição. Para cada descrição existe exatamente um dado de teste correspondente, conforme totalização na última linha da tabela. Os dados de teste DT_5, DT_6, DT_7 e DT_8 cobrem duas descrições cada um, reduzindo o número de dados de teste com respeito ao número de descrições, conforme totalização na última coluna da tabela.

Em síntese, foi possível gerar um conjunto com 16 dados de teste para atender às 20 descrições existentes e, consequentemente, cobrir as 14 classes de equivalência da aplicação segundo as diretrizes do TFS-TD.

Ao utilizar o TFS à mesma aplicação, Seguro de Veículos, é necessário acrescentar pelo menos mais 15 dados de teste, elevando para 31 o tamanho do conjunto adequado. As classes de equivalência e a quantidade de dados de teste adicionais requeridos, para cada uma delas, estão listados na Tabela 5.11.

Assim, nesse exemplo, o custo de aplicação do TFS-TD representa 51,61% do custo associado ao TFS, em relação ao tamanho do conjunto de dados de teste adequado (16 contra 31).

Tabela 5.11: *Seguro de Veículos - Classes Adicionais TFS*

Classe	**Quantidade de Dados de Teste**
V(1)	2
I(3)	2
I(4)	2
I(5)	2
I(6)	1
I(7)	1
I(8)	1
V(11)	1
V(12)	1
V(13)	1
V(14)	1
Total	15

Tabela 5.12: *Seguro Veículos - Tabela de decisão com descrições de dados de teste.*

Classes	Descrições																				
	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	D_{16}	D_{17}	D_{18}	D_{19}	D_{20}	Σ
I(2)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
I(6)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
I(7)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
I(8)	0	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	4
I(9)	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	3
V(11)	0	0	0	0	0	1*	0	0	0	1	1	0	0	0	0	0	0	0	0	0	2+1*
V(12)	0	0	0	0	0	0	1*	0	0	0	0	1	1	0	0	0	0	0	0	0	2+1*
V(13)	0	0	0	0	0	1**	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2+1**
V(14)	0	0	0	0	0	0	1**	0	0	0	0	0	0	0	0	0	1	1	0	0	2+1**
	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Sem Desconto	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	2
Desconto 10%	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	2
Desconto 15%	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	2
Desconto 30%	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	2

** cobertura condicional à instanciação da descrição, sexo-válido=feminino.*

*** cobertura condicional à instanciação da descrição, sexo-válido=masculino.*

Tabela 5.13: *Seguro Veículos - Matriz descrições versus dados.*

Casos de teste		Descrições																				
		D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	D_{16}	D_{17}	D_{18}	D_{19}	D_{20}	Σ
DT_1	(feminino)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
DT_2	(xyz, 40)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
DT_3	(feminino, 35.0)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
DT_4	(feminino, 0)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
DT_5	(feminino, 17)	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2
DT_6	(feminino, 18)	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2
DT_7	(feminino, 69)	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	2
DT_8	(feminino, 70)	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	2
DT_9	(feminino, 30)	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
DT_{10}	(feminino, 31)	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
DT_{11}	(feminino, 69)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
DT_{12}	(feminino, 70)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
DT_{13}	(masculino, 17)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
DT_{14}	(masculino, 18)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
DT_{15}	(masculino, 30)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
DT_{16}	(masculino, 31)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Σ		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

5.2 Estudo de Caso 1 - Teste do programa *cal*

Em Linkman et al. (2003) o TFS foi aplicado ao teste do programa aplicativo *cal*, calendário do sistema operacional *Unix*, mostrando dados pertinentes a custo e eficácia. Esta seção emprega o TFS-TD no teste do mesmo programa, visando a realizar comparações com os resultados relativos ao TFS.

O programa *cal* exibe na tela um mês específico ou todos os meses de um determinado ano de acordo com os parâmetros recebidos como entrada, que podem ser os seguintes:

- nenhum parâmetro - exibe na tela o mês corrente;
- um parâmetro (representa o ano) - exibe na tela todos os meses do ano informado. Um ano válido é dado por $1 \leq$ ano ≤ 9999. Observe-se que 83 refere-se ao ano 83 e não a 1983;
- dois parâmetros (representam mês e ano) - exibe na tela o mês do ano informado. Um mês válido é dado por $1 \leq$ mês ≤ 12;
- um caso especial é o ano 1752 em que foram suprimidos 11 dias do mês de setembro.

As Tabelas 5.14, 5.15, 5.16, 5.17 e 5.18 apresentam as classes de equivalência para este aplicativo, conforme definição em Linkman et al. (2003).

Tabela 5.14: *Programa cal: classes de equivalência pertinentes ao número de parâmetros de entrada.*

Parâmetros	$0 \leq z \leq 2$	$z > 2$
z	V (1)	I (2)

Tabela 5.15: *Programa cal: classes de equivalência inválidas para um único parâmetro (ano).*

ano	aaaa não inteiro	aaaa < 1	aaaa > 9999	$1 \leq$ aaaa ≤ 9999
aaaa	I (3)	I (4)	I (5)	V (6)

Tabela 5.16: *Programa cal: classes de equivalência inválidas para dois parâmetros (mês, ano).*

mês/ano	aaaa não inteiro	aaaa < 1	aaaa > 9999	$1 \leq$ aaaa ≤ 9999
não inteiro	I (7)	I (8)	I (9)	I (10)
mm < 1	I (11)	I (12)	I (13)	I (14)
mm > 12	I (15)	I (16)	I (17)	I (18)
$1 \leq$ mm ≤ 12	I (19)	I (20)	I (21)	V (22)

Tabela 5.17: *Programa cal: classes de equivalência válidas para um único parâmetro (ano).*

ano	número de dias
1752	356 V(23)
qualquer ano não bissexto	365 V(24)
qualquer ano bissexto	366 V(25)

Tabela 5.18: *Programa cal: classes de equivalência válidas para dois parâmetros (mês, ano).*

Mês e ano	número de dias
01,03,05,07,08,10,11/qualquer ano	31 V(26)
04,06,09,11/qualquer ano	30 V(27)
02/ano não bissexto	28 V(28)
02/ano bissexto	29 V(29)
09/1752	19 V(30)

O TFS foi aplicado ao teste do programa *cal*, onde o conjunto adequado, o qual é composto por 76 dados de teste, é exibido na Tabela 5.19, retirada de Linkman et al. (2003), conservando a mesma forma de identificação e descrição dos dados de teste. Linkman et al. ressaltam que foram gerados 4624 mutantes para o programa, dos quais 335 eram mutantes equivalentes. O conjunto adequado ao TFS matou 100% dos mutantes não equivalentes. Portanto, é desejavel que o conjunto adequado ao TFS-TD tenha qualidade similar ao conjunto adequado ao TFS.

5.2.1 TFS-TD aplicado ao Teste do Programa *cal*.

Na execução da Etapa E_1, as classes de equivalência foram analisadas e as seguintes foram marcadas: V(6) e V(22), pois representam partições incluídas por outras partições (Diretriz Dir_2); I(7), I(8), I(9), I(11), I(12), I(13), I(15), I(16) e I(17), pois são definidas pela combinação de valores inválidos (Diretriz Dir_1).

Na execução da Etapa E_2, foram geradas as descrições de dados de teste apresentadas na Tabela 5.20.

Na execução da Etapa E_3, foi construída a tabela de decisão com descrições de dados de teste, conforme pode ser observado na Figura 5.1; Na tabela constante desta figura, a coluna rotulada por $\sum$ apresenta a quantidade de descrições que cobrem cada classe de equivalência do programa *cal*.

Vale ressaltar que a cobertura das classes V(26), V(27) e V(28) pelas descrições D_{15} e D_{16} está condicionada a instanciação do mês, tendo em vista que para a cobertura dessas classes é requerido o mês pertencente a {1, 3, 5, 7, 8, 10, 12}, {4, 6, 9, 11} e {2}, respectivamente.

Tabela 5.19: *Programa cal: Conjunto de dados de teste adequado ao TFS.*

Caso de Teste	Parâmetros de Entrada	Partitições cobertas	Caso de Teste	Parâmetros de Entrada	Partitições cobertas
TC_1	9, 1752	1, 22, 30	TC_{39}	3, -9999	1, 20
TC_2	2, 1200	1, 22, 29	TC_{40}	3, -10000	1, 20
TC_3	2, 1000	1, 22, 29	TC_{41}	3, 10000	1, 21
TC_4	2, 1900	1, 22, 28	TC_{42}	a, 2000	1, 10
TC_5	2, 1104	1, 22, 29	TC_{43}	1.0, 2000	1, 10
TC_6	2, 2000	1, 22, 29	TC_{44}	3, z	1, 19
TC_7		1	TC_{45}	3, 2.0	1, 19
TC_8	1	1, 6, 24	TC_{46}	10, 1000, 5	2
TC_9	1999	1, 6, 24	TC_{47}	+10, 1000	1, 22, 26
TC_{10}	7999	1, 6, 24	TC_{48}	'(10)', 1000	1, 10
TC_{11}	1, 1	1, 22, 26	TC_{49}	10, +1000	1, 22, 26
TC_{12}	1, 1999	1, 22, 26	TC_{50}	10, '(1000)'	1, 19
TC_{13}	1, 7999	1, 22, 26	TC_{51}	0012, 2000	1, 22, 26
TC_{14}	1, 9999	1, 22, 26	TC_{52}	012, 2000	1, 22, 26
TC_{15}	12, 1999	1, 22, 26	TC_{53}	10, 0083	1, 22, 26
TC_{16}	12, 1	1, 22, 26	TC_{54}	10, 083	1, 22, 26
TC_{17}	12, 7999	1, 22, 26	TC_{55}	10, 2000, A	2
TC_{18}	12, 9999	1, 22, 26	TC_{56}	10, A, 2000	2
TC_{19}	6, 1	1, 22, 27	TC_{57}	A, 10, 2000	2
TC_{20}	6, 1999	1, 22, 27	TC_{58}	2.0, 10, 2000	2
TC_{21}	6, 7999	1, 22, 27	TC_{59}	10, 2.0, 2000	2
TC_{22}	6, 9999	1, 22, 27	TC_{60}	10, 2000, 2.0	2
TC_{23}	9, 1	1, 22, 27	TC_{61}	9999	1, 6, 24
TC_{24}	9, 1999	1, 22, 27	TC_{22}	0	1, 4
TC_{25}	9, 7999	1, 22, 27	TC_{63}	10000	1, 5
TC_{26}	9, 9999	1, 22, 27	TC_{64}	-9999	1, 4
TC_{27}	8, 1752	1, 22, 26	TC_{65}	a	1, 3
TC_{28}	10, 1752	1, 22, 26	TC_{66}	A, b	1, 7
TC_{29}	9, 1751	1, 22, 27	TC_{67}	a, -1	1, 8
TC_{30}	9, 1753	1, 22, 27	TC_{68}	a, 10000	1, 9
TC_{31}	2, 1752	1, 22, 29	TC_{69}	-1, a	1, 11
TC_{32}	0, 2000	1, 14	TC_{70}	-1, -1	1, 12
TC_{33}	-1, 2000	1, 14	TC_{71}	-1, 10000	1, 13
TC_{34}	-14, 2000	1, 14	TC_{72}	13, a	1, 15
TC_{35}	-12, 2000	1, 14	TC_{73}	13, -1	1, 16
TC_{36}	13, 2000	1, 18	TC_{74}	13, 10000	1, 17
TC_{37}	3, 0	1, 20	TC_{75}	1752	1, 6, 23
TC_{38}	3, -1	1, 20	TC_{76}	2000	1, 6, 25

Na execução da Etapa E_4, foi observado que as Classes V(23), V(25), V(29) e V(30) possuem apenas uma descrição de dados, contrariando a Diretriz Dir_3 no que diz respeito a gerar pelo menos dois dados de teste para cobrir cada classe válida. Para solucionar essa violação, vale ponderar: (i) as Classes V(23) e V(30) são caracterizadas por valores pontuais e, por definição, são cobertas por um único dado de teste; (ii) foram acrescidas as Descrições D22a e D37a para a cobertura das Classes V(25) e V(29), conforme pode ser observado nas duas últimas colunas da tabela constante da Figura 5.1.

Na execução da Etapa E_5, foram considerados os dados gerados a partir do TFS, conforme apresentados na Tabela 5.19, para fins de comparação de custo e qualidade. A intenção é utilizar os dados de teste do conjunto adequado ao TFS para atender as descrições de dados elaboradas a partir do TFS-TD. A matriz *descrições versus*

Tabela 5.20: *Programa cal: Descrições para dados de teste*

Id	Descrição	Id	Descrição
D_1	*nenhum parâmetro*	D_{23}	(1, ano-válido)
D_2	(mês-válido, ano-válido, outro-valor)	D_{24}	(3, ano-válido)
D_3	(ano-não-inteiro)	D_{25}	(5, ano-válido)
D_4	(0)	D_{26}	(7, ano-válido)
D_5	(1)	D_{27}	(8, ano-válido)
D_6	(9999)	D_{28}	(10, ano-válido)
D_7	(10000)	D_{29}	(12, ano-válido)
D_8	(mês-não-inteiro, ano-válido)	D_{30}	(4, ano-válido)
D_9	(0 , ano-válido)	D_{31}	(6, ano-válido)
D_{10}	(1 , ano-válido)	D_{32}	(9, ano-válido)
D_{11}	(12, ano-válido)	D_{33}	(11, ano-válido)
D_{12}	(13, ano-válido)	D_{34}	(2 , ano-não-bissexto)
D_{13}	(mes-válido , ano-não-inteiro)	D_{35}	(1, ano-não-bissexto)
D_{14}	(mês-válido , 0)	D_{36}	(3, ano-não-bissexto)
D_{15}	(mês-válido , 1)	D_{37}	(2 , ano-bissexto)
D_{16}	(mês-válido , 9999)	D_{37a}	(2 , ano-bissexto)
D_{17}	(mâs-válido , 10000)	D_{38}	(1, ano-bissexto)
D_{18}	(1752)	D_{39}	(3, ano-bissexto)
D_{19}	(1751)	D_{40}	(9, 1752)
D_{20}	(1753)	D_{41}	(9, 1751)
D_{21}	(ano-não-bissexto)	D_{42}	(9, 1753)
D_{22}	(ano-bissexto)	D_{43}	(8, 1752)
D_{22a}	(ano-bissexto)	D_{44}	(10, 1752)

dados é apresentada nas tabelas constante da Figura 5.2. Os dados presentes nas primeiras colunas foram extraídos do conjunto adequado ao TFS, mantida a mesma forma de identificação e descrição.

Pode-se observar na Figura 5.2 que 27 dados de teste do conjunto adequado ao TFS foram utilizados. Onze descrições de dados, dentre as 46 definidas, não foram cobertas, pois não foi possivel obter dados para atendê-las a partir do conjunto adequado ao TFS, conforme demonstrado na última linha da segunda tabela (a presença de valor zero), a saber: D_{19}, D_{20}, D_{22a}, D_{24}, D_{25}, D_{26}, D_{30}, D_{33}, D_{36}, D_{38} e D_{39}. Esse fato ocorre potencialmente devido à interpretação do testador com respeito às classes de equivalência; por exemplo, V(26) pode ser interpretada como uma classe de entrada com valores discretos, ou como uma classe de saída. A cobertura dessas 11 descrições requer a adição de 10 novos dados de teste, para compor um conjunto adequado ao TFS-TD..

Os 27 dados de teste foram aplicados aos 4289 mutantes não equivalentes do programa *cal*. Foram mortos 4286 desse mutantes, restando somente 3 mutantes vivos, resultando no escore de mutação igual a 99,93%. Sobre tais dados, pode-se afirmar que: (i) o número de dados de teste aplicados ao programa *cal* representa 35,53% (27 contra 76) do conjunto adequado ao TFS; (ii) a qualidade obtida é similar ao TFS, o qual obteve 100% de mutantes mortos.

Para gerar um conjunto adequado ao TFS-TD, foram acrescentados os seguintes dados de teste: (1751), (1753), (1000), (3, 1900), (1, 2000), (3, 2000), (5, 1), (7,

1), (4, 1) e (11, 1). Assim, o conjunto adequado ao TFS-TD possui 37 dados de teste, representando 48,68% do conjunto adequado ao TFS. Esse conjunto também foi aplicado aos 4289 mutantes não equivalentes do programa *cal*, resultando na morte de 4286 mutantes não equivalentes, mantendo-se o escore de mutação muito próximo daquele alcançado pelo TFS. Os três mutantes não eliminados são gerados pela combinação de valores inválidos (mês e ano menores que 1). O atendimento à Diretriz Dir_1, resultou na não eliminação destes mutantes.

Em síntese:

- o conteúdo da matriz *descrição versus dados* apresenta que houve a racionalização na geração de dados de teste, visto que a cobertura de 35 descrições foi alcançada com 27 dados de teste;
- o conjunto adequado ao TFS-TD possui cerca de metade do tamanho conjunto adequado ao TFS;
- o conjunto adequado ao TFS-TD possui qualidade similar ao conjunto adequado ao TFS, devido à proximidade dos escores de mutação obtidos.

Classes	Descrições																																														
	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	D_{16}	D_{17}	D_{18}	D_{19}	D_{20}	D_{21}	D_{22}	D_{23}	D_{24}	D_{25}	D_{26}	D_{27}	D_{28}	D_{29}	D_{30}	D_{31}	D_{32}	D_{33}	D_{34}	D_{35}	D_{36}	D_{37}	D_{38}	D_{39}	D_{40}	D_{41}	D_{42}	D_{43}	D_{44}	Σ	D_{22a}	D_{37a}
V (1)	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	43	1	1
I (2)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
I (3)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
I (4)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
I (5)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
I (10)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
I (14)	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
I (18)	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
I (19)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
I (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
I (21)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Ações																																															
V (23)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
V (24)	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0
V (25)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
V (26)	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1*	1*	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	1	1	16+1*	0	0
V (27)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1**	1**	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0	7+1**	0	0
V (28)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1***	1***	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2+1***	0	0
V (29)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1
V (30)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

Figura 5.1: *Programa cal: Tabela de decisão com descrições de dados de teste*

** cobertura condicional à instanciação da descrição, mês* $\in$ *{1, 3, 5, 7, 8, 10, 12}*
*** cobertura condicional à instanciação da descrição, mês* $\in$ *{4, 6, 9, 11}*
**** cobertura condicional à instanciação da descrição, mês = 2*

Dados de Teste			Descrições																																												
Seq	Id	valor	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	D_{16}	D_{17}	D_{18}	D_{19}	D_{20}	D_{21}	D_{22}	D_{22a}	D_{23}	D_{24}	D_{25}	D_{26}	D_{27}	D_{28}	D_{29}	D_{30}	D_{31}	D_{32}	D_{33}	D_{34}	D_{35}	D_{36}	D_{37}	D_{37a}	D_{38}	D_{39}	D_{40}	D_{41}	D_{42}	D_{43}	D_{44}	Σ
1	TC_1	{9, 1752}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
2	TC_2	{2, 1200}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
3	TC_4	{2 , 1900}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
4	TC_6	{2 , 2000}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
5	TC_7	{ }	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
6	TC_8	{ 1 }	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
7	TC_{11}	{1 , 1}	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	4
8	TC_{18}	{12, 9999}	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
9	TC_{19}	{6 , 1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
10	TC_{23}	{9, 1}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11	TC_{27}	{8 , 1752}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
12	TC_{28}	{10 , 1752}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2
13	TC_{29}	{9 , 1751}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	2
14	TC_{30}	{9 , 1753}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	2
15	TC_{32}	{0, 2000}	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
16	TC_{36}	{13 , 2000}	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
17	TC_{37}	{3 , 0}	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
18	TC_{41}	{3, 10000}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
19	TC_{42}	{a , 2000}	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
20	TC_{44}	{3, z}	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
21	TC_{46}	{10 , 1000, 5}	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
22	TC_{61}	{ 9999 }	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
23	TC_{62}	{ 0 }	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
24	TC_{63}	{ 10000 }	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
25	TC_{65}	{ a }	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
26	TC_{75}	{ 1752 }	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
27	TC_{76}	{ 2000 }	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		Σ	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	2	1	0	1	0	0	0	1	1	1	0	1	3	0	1	1	0	1	1	0	0	1	1	1	1	1	

Figura 5.2: *Programa cal: Matriz descrições versus dados*

5.3 Estudo de Caso 2 - Teste do PAF-ECF

O segundo estudo de caso explora a utilização do TFS-TD na geração de dados para o teste do Programa Aplicativo Fiscal - Emissor de Cupom Fiscal (PAF-ECF). O PAF é um aplicativo comercial responsável pelo envio de comandos ao Emissor de Cupom Fiscal (ECF), popularmente conhecido como impressora fiscal. Este aplicativo é desenvolvido de acordo com uma especificação de requisitos definida pelo Conselho Nacional de Política Fazendária - CONFAZ, através da Comissão Técnica Permanente do Imposto sobre Operações Relativas à Circulação de Mercadorias e sobre Prestações de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação - COTEPE/ICMS. A especificação de requisitos, juntamente com a legislação regulamentadora deste aplicativo encontra-se publicada inicialmente através do Ato Copete 06/08 (CONFAZ, 2008) e do Convênio do ICMS 15/08 (CONFAZ, 2008(a)), sendo atualizada oportunamente sempre que houver alterações na legislação e quando a COTEPE/ICMS detecta a necessidade de alguma mudança, por meio da publicação de novos atos.

Este aplicativo é utilizado por diversos segmentos comerciais dentre os quais supermercados, drogarias, postos de combustíves, prestadores de serviços, existindo especificidades e particularidades aos utilizados por bares, restaurantes e similares, postos de combustíveis, farmácias de manipulação e prestadores de serviços de transporte de passageiros. A utilização deste aplicativo requer a liberação por parte da Secretaria de Fazenda do Estado do domicilio fiscal do utilizador. Esta liberação dá-se pela submissão e aprovação do aplicativo ao processo de certificação efetuado por unidades certificadores devidamente credenciadas junto ao CONFAZ. É aprovado e, portanto certificado, o aplicativo que apresentar o resultado esperado (condição de requisito atendido) para todos os testes de todos os requisitos.

Esta certificação utiliza como guia o **Roteiro de Teste PAF-ECF** (CONFAZ, 2010), que contêm um conjunto requisitos e testes dispostos em sete blocos, sendo:

- Blocos I e VII contêm os testes relativos aos requisitos que são obrigatórios a todos os tipos de PAF-ECF;
- Bloco II contêm os testes relativos aos requisitos que são obrigatórios ao PAF-ECF destinado a Postos de Combustíveis;
- Bloco III contêm os testes relativos aos requisitos que são obrigatórios ao PAF-ECF destinado a Bares, Restaurantes e Similares;
- Bloco IV contêm os testes relativos aos requisitos que são obrigatórios ao PAF-ECF destinado a Farmácias de Manipulação;

- Bloco V contêm os testes relativos aos requisitos que são obrigatórios ao PAF-ECF destinado a Oficinais de Conserto;
- Bloco VI contêm os testes relativos aos requisitos que são obrigatórios ao PAF-ECF destinado a Prestadores de Serviços de Transporte de Passageiros;

O **Roteiro de Teste PAF-ECF** é organizado em *Requisitos*, numerados sequencialmente em algarismos romanos e que são desmembrados em *Itens*. Os testes destes requisitos são descritos no roteiro na forma de passos que devem ser seguidos sequencialmente. Cada teste possui suas condições de atendimento e de não atendimento. O conjunto de testes previsto para cada requisito não segue qualquer estratégia de teste funcional. Em adição, esse conjunto busca atestar o correto funcionamento do PAF-ECF para o requisito, não estando em sintonia com o objetivo primário da atividade de teste que é revelar a presença de defeitos. Além deste conjunto de teste, constam ainda do roteiro, instruções sobre o ambiente de teste (por exemplo, funcionamento em rede para acesso a dados remotos), layout de arquivos de configuração, operação de periféricos (por exemplo, alteração de data de impressora fiscal), etc.

5.3.1 Teste do PAF-ECF com o TFS

Vidal (2011) introduziu o Teste Funcional Sistemático Estendido (TFSE), uma extensão do TFS para contemplar os tipos de dados data e hora. Foi apresentada a aplicação do TFSE, e consequentemente do TFS, à geração de dados para o teste dos Requisitos XII e XXI do **Roteiro de Teste PAF-ECF**.

Esta subseção introduz as classes de equivalência e dados de teste do PAF-ECF, conforme publicado em Vidal (2011). O TFS-TD, na Subseção 5.3.2, é empregado no testes destes mesmos requisitos, por considerá-los como bons representantes dos demais requisitos constantes do referido roteiro. As funcionalidades contempladas estão abaixo transcritas a partir de Confaz (2010).

5.3.1.1 *Requisito XII*

***ITEM 1**: O PAF-ECF deve disponibilizar tela para registro e emissão de Comprovante Não Fiscal relativo às operações de retirada e de suprimento de caixa.*

***TESTE 041**: Registro de Suprimento de Caixa.*

***Passo 1**: Localize nos menus do programa a opção que permite registrar suprimento de caixa.*

***Passo 2**: Registre um suprimento de caixa no valor de R$ 1,00. Observe se o ECF emitiu o Comprovante Não Fiscal relativo ao suprimento de caixa corretamente.*

***Condição para requisito atendido**: Emissão do Comprovante Não Fiscal de Suprimento de Caixa no valor de R$ 1,00.*

***Condição para requisito não atendido**: Inexistência de função para registro de Suprimento de Caixa ou falta de emissão do Comprovante Não Fiscal de Suprimento de Caixa.*

***TESTE 042**: Registro de Sangria ou Retirada de Caixa.*

***Passo 1**: Localize nos menus do programa a opção que permite registrar sangria ou retirada de caixa.*

***Passo 2**: Registre uma sangria ou retirada de caixa no valor de R$ 0,50. Observe se o ECF emitiu o Comprovante Não Fiscal relativo à sangria de caixa corretamente.*

***Condição para requisito atendido**: Emissão do Comprovante Não Fiscal de Sangria ou Retirada de Caixa no valor de R$ 0,50.*

***Condição para requisito não atendido**: Inexistência de função para registro de Sangria ou Retirada de Caixa ou falta de emissão do Comprovante Não Fiscal de Sangria ou Retirada de Caixa.*

5.3.1.2 *Requisito XXI*

***ITEM 3**: No registro de venda, o PAF-ECF deve recusar valor negativo ou nulo nos campos:*

a) valor unitário da mercadoria ou do serviço;

b) quantidade da mercadoria ou do serviço;

c) meios de pagamento;

***TESTE 058**: Emissão de Cupom Fiscal com valor negativo ou nulo (zero) na quantidade do item.*

***Passo 1**: Abra um Cupom Fiscal.*

***Passo 2**: Registre um item comercializado.*

***Passo 3**: No campo relativo à quantidade comercializada, tente digitar um valor nulo (zero) e depois tente digitar um valor negativo.*

***Condição para requisito atendido**: Rejeição de valor nulo (zero) e de valor negativo.*

***Condição para requisito não atendido**: Permissão do registro com valor nulo (zero) ou negativo.*

Para o Requisito XII/Teste 041, registro de um suprimento de caixa, cuja faixa de valores válidos para tal suprimento foi convencionada entre R$ 0,01 e R$ 9.999.999,99, foi definido um conjunto de 4 classes de equivalência, descritas na Tabela 5.21, em que a primeira coluna refere-se ao identificador da classe, a segunda refere-se à sua descrição e a terceira contêm um indicador se a classe é válida ou inválida.

Tabela 5.21: *Programa PAF-ECF: Classes de equivalência pertinentes ao Requisito XII/Teste 041.*

id	Descrição	Válida/Inválida
V(1)	0,01 $\leq$ Suprimento $\leq$ 9.999.999,99	Válida
I(2)	Suprimento $<$ 0,01	Inválida
I(3)	Suprimento $>$ 9.999.999,99	Inválida
I(4)	Suprimento com tipo de dado diferente	Inválida

Para exercitar as classes do Requisito XII/Teste 041 foi gerado o conjunto de dados de teste, que é apresentado na Tabela 5.22, onde a primeira coluna refere-se ao identificador do dado de teste, a segunda coluna apresenta o valor do dado de teste

e a terceira identifica qual classe de equivalência o dado exercita. Neste conjunto, constam 14 dados de teste, dos quais 8 são para exercitar a classe V(1) e 6 para as demais classes, sendo 2 dados de teste para cada classe inválida.

Tabela 5.22: *Programa PAF-ECF: dados de teste para o Requisito XII/Teste 041, de acordo com o TFS.*

Id	Valor	Classe	Id	Valor	Classe
DT_1	0,01	V (1)	DT_8	1.987.876,09	V(1)
DT_2	9.999.999,99	V(1)	DT_9	0,00	V(2)
DT_3	0,02	V(1)	DT_{10}	-0,01	V(2)
DT_4	0,05	V(1)	DT_{11}	10.000.000,00	V(3)
DT_5	9.999.999,98	V(1)	DT_{12}	10.000.000,01	V(3)
DT_6	9.999.999,00	V(1)	DT_{13}	AE	V(4)
DT_7	1.000,02	V(1)	DT_{14}	!%$“”“”@!#*≤≥=≡≈∼=∝	V(4)

Para o Requisito XII/Teste 042, registro de uma sangria ou retirada de caixa, a faixa de valores válidos também foi convencionada entre R$ 0,01 e R$ 9.999.999,99. O teste deste requisito, além dos valores válidos e inválidos para a retirada de caixa, exige a geração de valores especiais para testar a relação de dependência existente entre a retirada e o saldo em caixa. O valor da retirada nunca pode ser superior ao valor do saldo em caixa e este nunca pode ser negativo. A Tabela 5.23 introduz as classes de equivalência definidas para o teste deste requisito, exibindo o identificador da classe, a sua descrição e um indicador se a classe é válida ou inválida.

Tabela 5.23: *Programa PAF-ECF: classes de equivalência pertinentes ao Requisito XII/Teste 042.*

id	Descrição	Válida/Inválida	id	Descrição	Válida/Inválida
V(5)	retirada = caixa = 0	Válida	I(12)	caixa < 0	Inválida
V(6)	retirada = 0 < caixa	Válida	V(13)	0,01 ≤ retirada ≤ 9.999.999,99	Válida
V(7)	0 < retirada = caixa	Válida	I(14)	retirada < 0,01	Inválida
V(8)	0 < retirada < caixa	Válida	I(15)	retirada > 9.999.999,99	Inválida
I(9)	caixa = 0 < retirada	Inválida	I(16)	retirada com tipo de dado diferente	Inválida
I(10)	0 < caixa < retirada	Inválida			
I(11)	retirada < 0	Inválida			

Para exercitar as classes atribuídas ao Requisito XII/Teste 042 foi gerado o conjunto de dados de teste apresentado na Tabela 5.24. Neste conjunto constam 26 dados de teste, dos quais 14 são para exercitar classes válidas e 12 para classes inválidas.

O Requisito XXI/Teste 058 avalia a quantidade de produto vendida em relação à quantidade em estoque, sendo válida a situação em que a quantidade vendida é menor ou igual a quantidade em estoque, e inválida em qualquer outra situação. A quantidade em estoque não pode conter valores negativos. A quantidade vendida deve ser maior do que zero, estabelecendo como válido o limte entre 1 e 9.999.999. A Tabela 5.25 apresenta as classes de equivalência definidas para este requisito.

Tabela 5.24: *Programa PAF-ECF: dados de teste para o Requisito XII/Teste 042, conforme o TFS.*

id	Retirada	Caixa	Classe	id	Retirada	Caixa	Classe
DT_{15}	0,00	0,00	V(5)	DT_{28}		-0,02	I(12)
DT_{16}	0,00	1,00	V(6)	DT_{29}		-0,10	I(12)
DT_{17}	0,00	500.000,00	V(6)	DT_{30}	9.999.999,99		V(13)
DT_{18}	5,00	5,00	V(7)	DT_{31}	0,01		V(13)
DT_{19}	300.999,99	300.999,99	V(7)	DT_{32}	0,02		V(13)
DT_{20}	9,99	10,00	V(8)	DT_{33}	9.999.999,98		V(13)
DT_{21}	300.999,99	500.999,99	V(8)	DT_{34}	9.999.999,00		V(13)
DT_{22}	50,00	0,00	I(9)	DT_{35}	1.000,02		V(13)
DT_{23}	0,50	0,00	I(9)	DT_{36}	1.987.876,09		V(13)
DT_{24}	0,02	0,01	I(10)	DT_{37}	10.000.000,00		I(15)
DT_{25}	10,00	5,00	I(10)	DT_{38}	10.000.000,01		I(15)
DT_{26}	-0,01		I(11)	DT_{39}	AE		I (16)
DT_{27}	-10,00		I(11)	DT_{40}	l%$“”“”@!#*≤≥=≡≈∼=∝		I (16)

Tabela 5.25: *Programa PAF-ECF: classes de equivalência pertinentes ao Requisito XXI/Teste 058*

id	Descrição	Válida/Inválida	id	Descrição	Válida/Inválida
V(17)	venda = estoque = 1	Válida	I(24)	estoque < 0	Inválida
V(18)	venda = 1 < estoque	Válida	V(25)	1 ≤ venda ≤ 9.999.999	Válida
V(19)	0 < venda = estoque	Válida	I(26)	venda < 1	Inválida
V(20)	0 < venda < estoque	Válida	I(27)	venda > 9.999.999	Inválida
I(21)	estoque = 1 < venda	Inválida	I(28)	venda com tipo de dado diferente	Inválida
I(22)	1 < estoque < venda	Inválida			
I(23)	venda < 0	Inválida			

Para exercitar as classes pertinentes ao Requisito XXI/Teste 058 foi gerado o conjunto de dados de teste apresentado na Tabela 5.26, contendo 26 dados de teste, dos quais 14 são para exercitar classes válidas e 12 para classes inválidas.

Em síntese, segundo apresentado em Vidal (2011), foram gerados 66 dados de teste para o conjunto adequado ao TFS, visando ao teste das funcionalidades presentes no **Roteiro de Teste PAF-ECF** que foram contempladas no estudo.

5.3.2 Teste do PAF-ECF com o TFS-TD

Na execução da Etapa E_1, não foram identificadas classe geradas a partir da combinação de valores inválidos e classes incluídas por outras.

Na execução da Etapa E_2, foram gerados os conjuntos de descrições de dados de teste para o Requisito XII/Teste 041, o Requisito XII/Teste 042 e o Requisito XXI/-Teste 058, que estão apresentados nas Tabelas 5.27, 5.28 e 5.29, respectivamente. Tais tabelas contêm o identificador da descrição de dados de teste e a descrição propriamente dita.

Na execução da Etapa E_3, foram geradas Tabelas de Decisão para a verificar se todas as classes de equivalência são efetivamente cobertas pelas descrições de dados

Tabela 5.26: *Programa PAF-ECF: dados de teste para o Requisito XXI/Teste 058, de acordo com o TFS*

id	Venda	Estoque	Classe	id	Venda	Estoque	Classe
DT_{41}	1	1	V(17)	DT_{54}		-2	I(24)
DT_{42}	1	2	V(18)	DT_{55}		-10	I(24)
DT_{43}	1	500.001	V(18)	DT_{56}	9.999.999		V(25)
DT_{44}	10	10	V(19)	DT_{57}	2		V(25)
DT_{45}	459.999	459.999	V(19)	DT_{58}	3		V(25)
DT_{46}	9	10	V(20)	DT_{59}	9.999.999		V(25)
DT_{47}	305	506	V(20)	DT_{60}	9.999.998		V(25)
DT_{48}	50	1	I(21)	DT_{61}	1.005		V(25)
DT_{49}	1	1	I(21)	DT_{62}	8.987.659		V(25)
DT_{50}	20	5	I(22)	DT_{63}	10.000.000		I(27)
DT_{51}	15	4	I(22)	DT_{64}	10.000.001		I(27)
DT_{52}	-1		I(23)	DT_{65}	$$$$		I(28)
DT_{53}	-50		I(23)	DT_{66}	venda%$“”“”@!#*≤≥=≡≈∼=		I(28)

Tabela 5.27: *Programa PAF-ECF: Descrições de dados para o Requisito XII/Teste 041*

id	Descrição
D_1	0,01
D_2	9.999.999,99
D_3	Suprimento < 0,01
D_4	Suprimento > 9999999,99
D_5	Suprimento com tipo de dado diferente

Tabela 5.28: *Programa PAF-ECF: Descrições de dados para o Requisito XII/Teste 042*

id	Descrição	id	Descrição
D_6	(retirada = 0, caixa = 0)	D_{13}	(Caixa < 0)
D_7	(0, caixa-válido)	D_{14}	(retirada = 0,01)
D_8	(retirada-válido, caixa = retirada)	D_{15}	(retirada = 9.999.999,99)
D_9	(retirada-válido, caixa > retirada)	D_{16}	(retirada = 0,00)
D_{10}	(retirada-válido, 0)	D_{17}	(retirada = 10.000.000,00)
D_{11}	(retirada-válido, caixa < retirada)	D_{18}	(retirada com tipo de dado diferente)
D_{12}	(retirada < 0)		

Tabela 5.29: *Programa PAF-ECF: Descrições de dados para o Requisito XXI/Teste 058*

id	Descrição	id	Descrição
D_{19}	(venda = 1, estoque = 1)	D_{26}	(Estoque < 0)
D_{20}	(1, estoque-válido)	D_{27}	(venda = 1)
D_{21}	(venda-válida, estoque = venda)	D_{28}	(venda = 9.999.999)
D_{22}	(venda-válida, estoque > venda)	D_{29}	(venda = 0)
D_{23}	(venda-válida, 0)	D_{30}	(venda = 10.000.000)
D_{24}	(venda-válida, estoque < venda)	D_{31}	(venda com tipo de dado diferente)
D_{25}	(venda < 0)		

de teste. A Tabela 5.30 apresenta a tabela de decisão relativa ao Requisito XII/Teste 041. A Tabela 5.31 apresenta a tabela de decisão relativa ao Requisito XII/Teste 042 e a Tabela 5.32 apresenta a tabela de decisão relativa ao Requisito XXI/Teste 058.

Na execução da Etapa E_4, observou-se que as classes V(5), V(6), V(7), V(8), V(17), V(18), V(19) e V(20) requerem descrições adicionais à sua cobertura. Para solucionar essa necessidade, foram acrescentadas novas descrições de dados. As Tabelas de Decisão 5.31 e 5.32 referem-se a tais descrições em suas últimas colunas. Vale ressaltar que as Classes V(5) e V(17) são cobertas por uma única descrição de dados, pois representam partições de um único dado de teste.

- D_{7a}: (0, caixa-válido)
- D_{8a}: (retirada-válida, caixa=retirada)
- D_{9a}: (retirada-válida, caixa>retirada)
- 2_{0a}: (1, estoque-válido)
- 2_{1a}: (venda-válida, estoque=venda)
- 2_{2a}: (venda-válida, estoque>venda)

Tabela 5.30: *Programa PAF-ECF: Tabela de decisão com descrições de dados de teste para Requisito XII/Teste 041*

Classes	Descrições					
	D_1	D_2	D_3	D_4	D_5	$\sum$
V(1)	1	1	0	0	0	2
I(2)	0	0	1	0	0	1
I(3)	0	0	0	1	0	1
I(4)	0	0	0	0	1	1
Suprimento efetuado	1	1	0	0	0	2
Reportar Erro	0	0	1	1	1	3

Na execução da Etapa E_5, foram selecionados dados de teste do conjunto adequado ao TFS para instanciar as descrições de dados pertinentes ao conjunto adequado ao TFS-TD. As matrizes *descrições versus dados* com os dados selecionados são apresentadas nas Tabelas 5.33, 5.28 e 5.35. Vale salientar que não foi possível instanciar à Descrição D_{29} a partir do conjunto adequado ao TFS, conforme pode ser observado na última linha da matriz presente na Tabela 5.35. Esse fato levou a geração de um novo dado de teste, visando a tornar o conjunto adequado ao TFS-TD.

O conjunto adequado ao TFS-TD possui 35 dados de teste, em relação às funcionalidades presentes no **Roteiro de Teste PAF-ECF** que foram contempladas

Tabela 5.31: *Programa PAF-ECF: Tabela de decisão com descrições de dados de teste para Requisito XII/Teste 042*

Classes	Descrições																
	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	D_{16}	D_{17}	D_{18}	$\sum$	D_{7a}	D_{8a}	D_{9a}
V(5)	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
V(6)	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
V(7)	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
V(8)	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
I(9)	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
I(10)	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
I(11)	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
I(12)	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
V(13)	0	0	0	0	0	0	0	0	1	1	0	0	0	2	0	0	0
I(14)	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
I(15)	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
I(16)	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
Valor válido	1	1	1	1	0	0	0	0	1	1	0	0	0	6	1	1	1
Valor inválido	0	0	0	0	1	1	1	1	0	0	1	1	1	7	0	0	0

Tabela 5.32: *Programa PAF-ECF: Tabela de decisão com descrições de dados de teste para Requisito XXI/Teste 058*

Classes	Descrições																
	D_{19}	D_{20}	D_{21}	D_{22}	D_{23}	D_{24}	D_{25}	D_{26}	D_{27}	D_{28}	D_{29}	D_{30}	D_{31}	$\sum$	D_{20a}	D_{21a}	D_{22a}
V(17)	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
V(18)	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
V(19)	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
V(20)	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
I(21)	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
I(22)	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
I(23)	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
I(24)	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
V(25)	0	0	0	0	0	0	0	0	1	1	0	0	0	2	0	0	0
I(26)	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
I(27)	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
I(28)	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
Valor válido	1	1	1	1	0	0	0	0	1	1	0	0	0	6	1	1	1
Valor inválido	0	0	0	0	1	1	1	1	0	0	1	1	1	7	0	0	0

Tabela 5.33: *Programa PAF-ECF: Matriz descrições versus dados - Requisito XII/Teste 041.*

Dados Teste			Descrições					
Seq	id	Valores	D_1	D_2	D_3	D_4	D_5	$\sum$
1	DT_1	(0,01)	1	0	0	0	0	1
2	DT_2	(9.999.999,99)	0	1	0	0	0	1
3	DT_9	(0,00)	0	0	1	0	0	1
4	DT_{11}	(10.000.000,00)	0	0	0	1	0	1
5	DT_{13}	AE	0	0	0	0	1	1
$\sum$			1	1	1	1	1	

Tabela 5.34: *Programa PAF-ECF: Matriz descrições versus dados - Requisito XII/Teste 042.*

Dados Teste			Descrições																
Seq	id	Valores	D_6	D_7	D_{7a}	D_8	D_{8a}	D_9	D_{9a}	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	D_{16}	D_{17}	D_{18}	Σ
1	DT_{15}	(0,00 , 0,00)	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	2
2	DT_{16}	(0,00 , 1,00)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	DT_{17}	(0,00 , 500.000,00)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	DT_{18}	(5,00 , 5,00)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
5	DT_{19}	(300.999,99 , 300.999,99)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
6	DT_{20}	(9,99 , 10,00)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
7	DT_{21}	(300.999,99 , 500.999,99)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
8	DT_{22}	(50,00 , 0,00)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
9	DT_{24}	(0,02 , 0,01)	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
10	DT_{26}	(-0,01)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
11	DT_{28}	(-0,02)	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
12	DT_{31}	(0,01)	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
13	DT_{30}	(9.999.999,99)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
14	DT_{37}	(10.000.000,00)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
15	DT_{39}	(AE)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Σ			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Tabela 5.35: *Programa PAF-ECF: Matriz descrições versus dados - Requisito XXI/Teste 058.*

Dados Teste			Descrições																
Seq	id	Valores	D_{19}	D_{20}	D_{20a}	D_{21}	D_{21a}	D_{22}	D_{22a}	D_{23}	D_{24}	D_{25}	D_{26}	D_{27}	D_{28}	D_{29}	D_{30}	D_{31}	Σ
1	DT_{41}	(1 , 1)	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2
2	DT_{42}	(1 , 2)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	DT_{43}	(1 , 500.001)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	DT_{44}	(10 , 10)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
5	DT_{45}	(459.999 , 459.999)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
6	DT_{46}	(9 , 10)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
7	DT_{47}	(305 , 506)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
8	DT_{48}	(50 , 1)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
9	DT_{50}	(20 , 5)	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
10	DT_{52}	(-1)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
11	DT_{54}	(-2)	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
12	DT_{56}	9.999.999	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
13	DT_{63}	10.000.000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
14	DT_{65}	$$$$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Σ			1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	

no estudo. O conjunto adequado ao TFS-TD representa 53,03% (35 contra 66) do conjunto adequado ao TFS. Um resumo comparativo por requisito com respeito ao tamanho do cunjunto adequado encontra-se na Tabela 5.36.

Tabela 5.36: *Quantidade de dados de teste - TFS x TFS-TD*

Requisito/Teste	TFS	TFS-TD
Requisito XII/041	14	5
Requisito XII/042	26	15
Requisito XXI/058	26	15
Σ	66	35

Vidal (2011) observa que se a quantidade de dados de teste requerida pelo TFS for estendida aos demais requisitos do **Roteiro de Teste PAF-ECF**, seria necessário aproximadamente 2000 dados de teste para o conjunto adequando ao TFS. Aplicando-se o mesmo raciocínio ao TFS-TD, esta quantidade poderia ser reduzida para cerca de 1061 dados de teste (2000 * 53,05%), mantendo-se praticamente a mesma capacidade de revelar defeitos, com exceção para aqueles potenciais defeitos revelados pela combinação de entradas inválidas.

5.4 Considerações Finais

Neste capítulo foi introduzido o Teste Funcional Sistemático com Aplicação de Tabela de Decisão (TFS-TD), um critério de teste funcional que estende o TFS, por: (i) redefinição do número de casos de teste para a cobertura de classes de equivalência e a aplicação de Tabela de Decisão; (ii) derivação de descrições de dados para os valores limites; (iii) racionalização de geração de dados de teste e uso da Matriz *descrições versus dados de teste*. O TFS-TD busca a redução do tamanho do conjunto de dados adequado, evitando perda da qualidade em termos de detecção de defeitos. Para alcançar este objetivo um processo e um conjunto de diretrizes foram definidos para a aplicação deste critério.

Foram apresentados três exemplos de comparação do TFS-TD com o Teste Funcional Sistemático (TFS): Seguro de Veículos, Programa *Cal* e **Roteiro de Teste PAF-ECF**. Em todos casos, a economia em relação ao tamanho do conjunto adequado foi significativa: 51,61% em relação ao primeiro, 48,68% em relação ao segundo e 53,05% em relação ao terceiro. Para o programa *Cal*, foi possível comparar também a qualidade do conjunto adequado pela análise de mutantes: a qualidade não foi afetada significativamente, tendo em vista que o escore de mutação alcançado pelo TFS-TD foi de 99,93%.

A conclusão que se chega a partir dos dados obtidos pelas três comparações é que o TFS-TD alcançou resultados importante em relação a sua proposta, pois mantém em si pontos fortes destacados em relação ao TFS, ao mesmo tempo que incorpora ao seu processo o rigor lógico peculiar à utilização da tabela de decisão.

CAPÍTULO 6

Conclusões e Trabalhos Futuros

Este trabalho teve seu foco no teste funcional de software, visando a contribuir com a sua avaliação e evolução. Especificamente, buscou-se solucionar a questão de como aplicar o teste funcional visando a redução de custo, sem perdas relevantes com respeito à qualidade do teste em termos de defeitos revelados. Para tal, foi preciso conhecer alguns dos critérios e técnicas funcionais mais utilizados, a forma como são avaliados, os seus cenários de aplicação e a maneira como podem ser empregados em conjunto.

Um estudo minucioso de vários critérios e técnicas funcionais foi efetuado. Uma revisão sistemática foi executada, procurando a identificação dos principais cenários de aplicação, comparações entre critérios e técnicas funcionais e utilização dos mesmos na avaliação de especificações de teste. Um conjunto de 27 estudos primários foram estudados, extraindo-se informações relevantes de cada um para o suporte às conclusões que embasaram as respostas às questões de pesquisa formuladas.

Um novo critério de teste funcional foi proposto, **Teste Funcional Sistemático com Aplicação de Tabela de Decisão (TFS-TD)**, o qual estende o critério *Teste Funcional Sistemático* (TFS), e possui características de três dos critérios funcionais mais utilizados, de acordo com os resultados da revisão sistemática: *Particionamento em Classes de Equivalência*, *Análise do Valor Limite* e *Tabela de Decisão*. Em relação ao TFS, o TFS-TD incorpora o seguinte: (i) redefinição do número de casos de teste para a cobertura de classes de equivalência e a aplicação de Tabela de Decisão; (ii) derivação de descrições de dados para os valores limites; (iii) racionalização de geração de dados de teste e uso da Matriz descrições versus dados de teste. O TFS-TD busca a redução do tamanho do conjunto de dados adequado, evitando perda da qualidade em termos de detecção de defeitos. Para alcançar este objetivo um processo e um conjunto de diretrizes foram definidos para a aplicação deste critério.

Foram apresentados três estudos que comparam o TFS-TD com o TFS: Seguro de Veículos, Programa *Cal* e Roteiro de Teste PAF-ECF. Em todos casos, a economia em relação ao tamanho do conjunto adequado foi significativa, em torno de 50%. Para o programa *Cal*, foi possível avaliar também a qualidade do conjunto adequado pela análise de mutantes: a qualidade não foi afetada significativamente, tendo em vista que o escore de mutação alcançado pelo TFS-TD foi de 99,93%. O TFS-TD alcançou resultados importantes em relação a sua proposta, pois mantém em si pontos fortes destacados em relação ao TFS, ao mesmo tempo que reduz o custo de aplicação. Enfim, a proposição e a aplicação do TFS-TD buscam solucionar preliminarmente o problema de pesquisa atribuído a este trabalho.

6.1 Contribuições

Algumas contribuições deste trabalho estão elencadas abaixo.

1. a revisão sistemática contribuiu para se conhecer melhor os aspectos de aplicação e de avaliação dos critérios e técnicas de teste funcional; alguns dos resultados obtidos foram:
    - os critérios e técnicas de teste funcional são aplicados numa grande variedade de cenários, desde sistemas desenvolvidos em ambiente acadêmico a sofisticados softwares embarcados em aviões militares;
    - os critérios *Análise do Valor Limite* e *Particionamento em Classes de Equivalência* foram os mais estudados e utilizados pelos estudos primários;
    - o critério *Análise do Valor Limite* foi utilizado em maior número de cenários;
    - o *Teste Baseado em Caso de Uso* foi o mais empregado em cenários críticos;
    - vários cenários de aplicação/utilização dos critérios de teste funcionais foram identificados, dentre estes o cenário didático/acadêmico esteve presente na maior parte dos estudos analisados;
    - não foi identificada a exclusividade de aplicação de um critério ou técnica x a um cenário y, pelo contrário, foram identificadas adaptações tornando os critérios e técnicas aplicáveis a cenários em que normalmente não são aplicados;

- a experiência e critatividade do testador são fundamentais para o emprego de um critério ou técnica, mesmo quando a sua aplicação em determinado cenário não for recomendada;
- em geral os critérios funcionais tiveram melhor avaliação em relação aos estruturais, nos estudos que abordaram suas comparações.

2. a definição do TFS-TD contribuiu com uma estratégia para a utilização conjunta de critérios difundidos na literatura; vale ressaltar que houve:
   - redução de custo, conforme exemplos constantes das Subseções 5.1.3, 5.2 e 5.3;
   - redução de custo sem perda da eficácia do teste, conforme exemplo constante da Subseção 5.2;
   - a definição de diretrizes para o teste funcional;
   - a sistematização de um processo, que permite a avaliação e a evolução do teste funcional, levando em consideração que a definição de um subconjunto com uma quantidade de casos de teste menor que o conjunto original e com medidas similares em relação à capacidade de detecção de defeitos, representa uma evolução ao teste;
   - a aplicação preliminar com resultados promissores para a avaliação de especificação de teste de aplicativos fiscais.

6.2 Trabalhos Futuros

Alguns desdobramentos para a pesquisa são:

- estender a avaliação empírica do TFS-TD a outros cenários reais;
- desenvolver estudos visando a construção de ferramentas para automatizar o processo de utilização do TFS-TD;
- aplicar o TFS-TD a todos os requisitos do Roteiro PAF-ECF;
- ampliar as bases de pesquisa da revisão sistemática, para enriquecer as respostas às questões de pesquisa, principalmente sobre a avaliação de roteiros (especificações) de teste.

Bibliografia

AFZAL, W.; TORKAR, R.; FELDT, R. A systematic review of search-based testing for non-functional system properties. *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 51, n. 6, p. 957–976, jun 2009. ISSN 0950-5849. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2008.12.005>.

ALI, M. S.; BABAR, M. A.; CHEN, L.; STOL, K. J. A systematic review of comparative evidence of aspect-oriented programming. *Information and Software Technology*, v. 52, n. 9, p. 871–887, 2010. ISSN 0950-5849.

ALVER, M. O. *JabRef*. 2008. Disponível em: <http://jabref.sourceforce.net>.

AMAR, M.; SHABBIR, K. *Systematic Review on Testing Aspect-oriented Programs - Challenges, Techniques and Their Effectiveness*. Dissertação (Mestrado) — School of Engineering Blekinge Institute of Technology, 2008.

AMMANN, P.; OFFUTT, J. *Introduction to Software Testing*. New York: Cambridge University Press, 2008.

BACH, J. *ALLPAIRS Test Case Generation Tool (Version 1.2.1)*. 2011. Disponível em: <http://www.satisfice.com/tools.shtml>. Acesso em: 22.08.2011.

BACH, J.; SCHROEDER, P. J. *Pairwise Testing: A Best Practice That Isn't*. 2004. Disponível em: <http://www.testingeducation.org>. Acesso em: 22.08.2011.

BARBOSA, J. R. *Estudo e Definição de uma Metodologia de Teste de Software no Contexto de Sistemas Embarcados Críticos*. Dissertação (Mestrado) — Universidade Federal de Goiás, Goiânia, 2011.

BARROS, V. S. S. *Simulador de Ambientes*. 2011. Disponível em: <http://lvelho-.impa.br/i3d07/demos/victor/DIAGRAMA DE CASO DE USO.htm>. Acesso em: 28.08.2011.

BASILI, V. R.; SELBY, R. W. Comparing the Effectiveness of Software Testing Strategies. *Software Engineering, IEEE Transactions on*, SE-13, n. 12, p. 1278–1296, dec. 1987.

BINDER, R. V. *Testing Object-Oriented Systems: Models, Patterns, and Tools*. [S.l.]: Addison-Wesley, 2000.

BIOLCHINI, J. C. d. A.; MIAN, P. G.; NATALI, A. C. C.; CONTE, T. U.; TRAVASSOS, G. H. Scientific research ontology to support systematic review in software engineering. *Adv. Eng. Inform.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 21, p. 133–151, April 2007. ISSN 1474-0346.

BOX, G. E. P.; HUNTER, W. G.; HUNTER, J. S. Book. *Statistics for experimenters : an introduction to design, data analysis, and model building*. [S.l.]: Wiley, New York, 1978. ISBN 0471093157.

BRITO, M. A. S.; FELIZARDO, K. R.; SOUZA, P. S. L.; SOUZA, S. R. S. *Concurrent Software Testing: A Systematic Review*. [S.l.], 2010.

BUDGEN, D.; BURN, A. J.; BRERETON, O. P.; KITCHENHAM, B. A.; PRETORIUS, R. Empirical evidence about the uml: a systematic literature review. *Softw. Pract. Exper.*, John Wiley & Sons, Inc., New York, NY, USA, v. 41, n. 4, p. 363–392, abr. 2011. ISSN 0038-0644. Disponível em: <http://dx.doi.org/10.1002/spe.1009>.

CAI, K.-Y.; GU, B.; HU, H.; LI, Y.-C. Adaptive software testing with fixed-memory feedback. *J. Syst. Softw.*, Elsevier Science Inc., New York, NY, USA, v. 80, p. 1328–1348, August 2007. ISSN 0164-1212.

CAI, K.-Y.; JING, T.; BAI, C.-G. Partition testing with dynamic partitioning. In: *Computer Software and Applications Conference, 2005. COMPSAC 2005. 29th Annual International*. [S.l.: s.n.], 2005. v. 1, p. 113–116.

CHANGEVISION. *Jude*. 2011. Disponível em: <http://jude.change-vision.com/jude-web/index.html>. Acesso em: 21.02.2012.

CHEN, T.; LEUNG, H.; MAK, I. Adaptive random testing. In: MAHER, M. (Ed.). *Advances in Computer Science - ASIAN 2004. Higher-Level Decision Making*. [S.l.]: Springer Berlin / Heidelberg, 2005, (Lecture Notes in Computer Science, v. 3321). p. 3156–3157.

CHEN, T. Y.; KUO, F.-C.; LIU, H. On test case distributions of adaptive random testing. In: *SEKE 07*. [S.l.: s.n.], 2007. p. 141–144.

CHEN, T. Y.; KUO, F.-C.; MERKEL, R. G.; TSE, T. H. Adaptive random testing: The art of test case diversity. *J. Syst. Softw.*, Elsevier Science Inc., New York, NY, USA, v. 83, p. 60–66, Jan 2010.

CIUPA, I.; PRETSCHNER, A.; LEITNER, A.; ORIOL, M.; MEYER, B. On the predictability of random tests for object-oriented software. In: *Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation*. Washington, DC, USA: IEEE Computer Society, 2008. (ICST '08), p. 72–81. ISBN 978-0-7695-3127-4. Disponível em: <http://dx.doi.org/10.1109/ICST.2008.20>.

CONFAZ, C. N. d. P. F. *Ato Cotepe 006*. 2008. Disponível em: <http://www-.fazenda.gov.br/confaz/confaz% -/atos/atos_cotepe/2008/ac006_08.htm>. Acesso em: 21.02.2012.

CONFAZ, C. N. d. P. F. *Convênio ICMS 15*. 2008(a). Disponível em: <http://www-.fazenda.gov.br/confaz/confaz% -/Convenios/ICMS/2008/cv015_08.htm>. Acesso em: 21.02.2012.

CONFAZ, C. N. d. P. F. *Roteiro de Análise Funcional de Programa Aplicativo Fiscal - Emissor de Cupom Fiscal*. 2010. Disponível em: <http://www.fazenda.gov.br/confaz-/default.htm>. Acesso em: 21.02.2012.

COPELAND, L. *A Practitioner's Guide to Software Test Design*. Boston: Artech House Publishers, 2003.

CZERWONKA, J. *Pairwise Testing - Combinatorial Test Case Generation*. 2011. Disponível em: <http://www.pairwise.org/>. Acesso em: 23.09.2011.

DEITEL, H. M.; DEITEL, P.; NIETO, T. *The Complete XML Programming Training Course*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN 0130895571.

DELAMARO, M. E.; MALDONADO, J. C.; VINCENZI, A. M. R. Proteum/im 2.0: An integrated mutation testing environment. In: *Proceedings of the 1st Workshop on Mutation Analysis (MUTATION'00)*. San Jose, California: [s.n.], 2001. p. 91–101.

DENGER, C.; MORA, M. M. *Test case derived from requirement specification*. [S.l.], 2003.

DURAN, J. W.; NTAFOS, S. C. An evaluation of random testing. *Software Engineering, IEEE Transactions on*, SE-10, n. 4, p. 438–444, july 1984. ISSN 0098-5589.

DYBÅ, T.; DINGSØYR, T. Strength of evidence in systematic reviews in software engineering. In: *Proceedings of the Second ACM-IEEE international symposium*

on Empirical software engineering and measurement. New York, NY, USA: ACM, 2008. (ESEM 08), p. 178–187. ISBN 978-1-59593-971-5. Disponível em: <http://doi.acm.org/10.1145/1414004.1414034>.

ESCALONA, M. J.; GUTIERREZ, J. J.; MEJíAS, M.; ARAGóN, G.; RAMOS, I.; TORRES, J.; DOMíNGUEZ, F. J. An overview on test generation from functional requirements. *J. Syst. Softw.*, Elsevier Science Inc., New York, NY, USA, v. 84, p. 1379–1393, August 2011.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. *Design patterns: elements of reusable object-oriented software*. Boston: Addison-Wesley Longman Publishing Co., Inc., 1995.

GERRARD, P.; THOMPSON, N. *Risk Based E-Business Testing*. Norwood, MA, USA: Artech House, Inc., 2002. ISBN 1580533140.

GUIMARãES, D. *Técnicas de Teste de Software*. 2011. Disponível em: <http://-/guimaraesdani.wordpress.com/testes-e-qualidade-de-software/tecnicas-de-teste-de-software/>. Acesso em: 21.08.2011.

GUTIERREZ, J. J. Towards a complete approach to generate system test cases. In: *ICEIS Doctoral Consortium*. [S.l.: s.n.], 2006. p. 38–50.

GUTIERREZ, J. J. *ObjectGen & ValueGen page*. 2007. Disponível em: <http://www-.lsi.us.es/~javierj/ObjectGen.html>. Acesso em: 24.06.2012.

GUTIERREZ, J. J.; ESCALONA, M. J.; MEJíAS, M.; TORRES, J. Generation of test cases from functional requirements. a survey. In: *4th Workshop on System Testing and Validation*. Potsdam, Germany: [s.n.], 2006.

GUTIERREZ, J. J.; ESCALONA, M. J.; MEJíAS, M.; TORRES, J. Derivation of test objectives automatically. In: WOJTKOWSKI, W.; WOJTKOWSKI, W. G.; ZUPANCIC, J.; MAGYAR, G.; KNAPP, G. (Ed.). *Advances in Information Systems Development*. [S.l.]: Springer US, 2007. p. 435–446.

GUTIERREZ, J. J.; ESCALONA, M. J.; MEJíAS, M.; TORRES, J.; CENTENO, A. H. A case study for generating test cases from use cases. In: *Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference on*. [S.l.: s.n.], 2008. p. 209–214.

HAMLET, D. When only random testing will do. In: *Proceedings of the 1st international workshop on Random testing*. New York, NY, USA: ACM, 2006. (RT '06), p. 1–9. ISBN 1-59593-457-X. Disponível em: <http://doi.acm.org/10.1145/1145735.1145737>.

HAMLET, D.; TAYLOR, R. Partition testing does not inspire confidence [program testing]. *Software Engineering, IEEE Transactions on*, v. 16, n. 12, p. 1402–1411, dec 1990.

HAREL, D. Statecharts: A visual formalism for complex systems. *Sci. Comput. Program.*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 8, p. 231–274, June 1987.

HIERONS, R. M. Avoiding coincidental correctness in boundary value analysis. *ACM Trans. Softw. Eng. Methodol.*, ACM, New York, NY, USA, v. 15, p. 227–241, July 2006.

HILL, E. F. *Jess in Action: Java Rule-Based Systems*. Greenwich, CT, USA: Manning Publications Co., 2003. ISBN 1930110898.

HUNT, T. *Cause-Effect Graphing*. 2007. Disponível em: <http://www.westfallteam-.com>. Acesso em: 21.09.2011.

IEEE. Ieee standard for software test documentation. *IEEE Std 829-1998*, IEEE Standards Board, New York, USA, 1998.

ISO/IEC. *ISO/IEC 9126. Software engineering – Product quality*. [S.l.]: ISO/IEC, 2001.

JACOBSON, I.; CHRISTERSON, M.; JONSSON, P.; OVERGAARD, G. *Object-Oriented Systems Engineering: A Use Case Driven Approach*. Addison-Wesley: CRC Press, 1992.

JONES, E. L. *Automated Support For Test-Driven Specification*. Phoenix, Arizona: [s.n.], nov. 14-16 2005. 218–223 p.

JORGENSEN, P. C. *Software Testing: A Craftman's Approach*. 2nd. ed. Boca Raton, FL, USA: CRC Press, Inc., 2001.

JORGENSEN, P. C. *Software Testing: A Craftman's Approach. (2nd ed.)*. [S.l.]: CRC Press, 2002.

JURISTO, N.; MORENO, A. M.; VEGAS, S.; SHULL, F. A look at 25 years of data. *Software, IEEE*, v. 26, n. 1, p. 15–17, jan/feb 2009.

JURISTO, N.; VEGAS, S. Functional testing, structural testing and code reading: What fault type do they each detect? In: CONRADI, R.; WANG, A. (Ed.). *Empirical Methods and Studies in Software Engineering*. [S.l.]: Springer Berlin / Heidelberg, 2003, (Lecture Notes in Computer Science, v. 2765). p. 208–232.

KAMSTIES, E.; LOTT, C. M. An empirical evaluation of three defect-detection techniques. In: *Proceedings of the 5th European Software Engineering Conference*. London, UK: Springer-Verlag, 1995. p. 362–383. ISBN 3-540-60406-5. Disponível em: <http://dl.acm.org/citation.cfm?id=645385.651507>.

KANER, C. The impossibility of complete testing. In: *in Software QA N. 4, online: http://www.kaner.com/articles.html*. [S.l.: s.n.], 1997. v. 4.

LINKMAN, S.; VINCENZI, A. M. R.; MALDONADO, J. C. An evaluation of systematic functional testing using mutation testing. *7th International Conference on Empirical Assessment in Software Engineering [EASE. [S.l.: s.n.]]*, 2003.

MARQUES, P. *Conjuntos*. 2011. Disponível em: <http://www.algosobre.com.br-/matematica/conjuntos.html>. Acesso em: 21.08.2011.

MASSOL, V.; HUSTED, T. *JUnit in Action*. Greenwich, CT, USA: Manning Publications Co., 2003. ISBN 1930110995.

MATHUR, A. P. *Software Testing*. 1st. ed. Nova York: Pearson Publication, 2008.

MAYER, J.; SCHNECKENBURGER, C. An empirical analysis and comparison of random testing techniques. In: *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. New York, NY, USA: ACM, 2006. (ISESE 06), p. 105–114.

MEYER, B. Applying "Design by Contract". *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 25, p. 40–51, October 1992. ISSN 0018-9162.

MURNANE, T.; HALL, R.; REED, K. Towards describing black-box testing methods as atomic rules. In: *Computer Software and Applications Conference, 2005. COMPSAC 2005. 29th Annual International*. [S.l.: s.n.], 2005. v. 2, p. 437–442.

MYERS, G. J. A Controlled Experiment in Program Testing and Code Walkthroughs/Inspections. *Commun. ACM*, ACM, New York, NY, USA, v. 21, p. 760–768, September 1978. ISSN 0001-0782.

MYERS, G. J. *The Art of Software Testing*. New York: John Wiley & Sons, 1979.

MYERS, G. J.; SANDLER, C. *The Art of Software Testing*. New Jersey: John Wiley & Sons, 2004.

NAUR, P. Programming by action clusters. *BIT Numerical Mathematics*, Springer Netherlands, v. 9, p. 250–258, 1969. ISSN 0006-3835.

NEBUT, C.; FLEUREY, F. *Technical and Experimental Material*. 2003. Disponível em: <http://www.irisa.fr/triskell/results/ISSRE03/>. Acesso em: 21.02.2012.

NEBUT, C.; FLEUREY, F.; TRAON, Y. L.; JEZEQUEL, J. M. Requirements by contracts allow automated system testing. In: *Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on*. [S.l.: s.n.], 2003. p. 85–96. ISSN 1071-9458.

NEBUT, C.; FLEUREY, F.; TRAON, Y. L.; JEZEQUEL, J. M. Automatic test generation: a use case driven approach. *Software Engineering, IEEE Transactions on*, v. 32, n. 3, p. 140–155, mar 2006.

NETO, A. C. D.; SUBRAMANYAN, R.; VIEIRA, M.; TRAVASSOS, G. H. A survey on model-based testing approaches: a systematic review. In: *Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007*. New York, NY, USA: ACM, 2007. (WEASELTech '07), p. 31–36. ISBN 978-1-59593-880-0. Disponível em: <http://doi.acm.org/10.1145/1353673.1353681>.

NOIKAJANA, S.; SUWANNASART, T. Web service test case generation based on decision table (short paper). In: *Quality Software, 2008. QSIC '08. The Eighth International Conference on*. [S.l.: s.n.], 2008. p. 321–326. ISSN 1550-6002.

NURSIMULU, K.; PROBERT, R. L. Cause-effect graphing analysis and validation of requirements. In: *Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research*. [S.l.]: IBM Press, 1995. (CASCON 95).

OMG, O. M. G. *UML - Unified Modeling Language*. 2011. Disponível em: <http://www.uml.org/>. Acesso em: 21.08.2011.

OMMERING, R. V.; LINDEN, F. V. D.; KRAMER, J.; MAGEE, J. The koala component model for consumer electronics software. *Computer*, v. 33, n. 3, p. 78–85, mar 2000. ISSN 0018-9162.

OSTRAND, T. J.; BALCER, M. J. The category-partition method for specifying and generating fuctional tests. *Commun. ACM*, ACM, New York, NY, USA, v. 31, p. 676–686, June 1988. ISSN 0001-0782.

PARADKAR, A.; TAI, K. C.; VOUK, M. A. Specification-based testing using cause-effect graphs. *Ann. Softw. Eng.*, J. C. Baltzer AG, Science Publishers, Red Bank, NJ, USA, v. 4, p. 133–157, January 1997.

PHADKE, M. S. *Design of Experiment for Software Testing*. 2000. Disponível em: <http://www.isixsigma.com/tools-templates/design-of-experiments-doe/design-experiment-software-testing/>. Acesso em: 22.08.2011.

PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach*. Nova York: McGraw-Hill, 2005.

RAMACHANDRAN, M. Testing software components using boundary value analysis. In: *Euromicro Conference, 2003. Proceedings. 29th*. [S.l.: s.n.], 2003. p. 94–98. ISSN 1089-6503.

RATIONAL, S. C. *Diretrizes: Caso de Teste*. 2010. Disponível em: <http://www-.wthreex.com/rup/process/modguide/md_tstcs.htm>. Acesso em: 24.08.2011.

REID, S. C. An empirical analysis of equivalence partitioning, boundary value analysis and random testing. In: *Software Metrics Symposium, 1997. Proceedings., Fourth International*. [S.l.: s.n.], 1997. p. 64–73.

ROUBTSOV, S.; HECK, P. Use case-based acceptance testing of a large industrial system: Approach and experience report. In: *Testing: Academic and Industrial Conference - Practice And Research Techniques, 2006. TAIC PART 2006. Proceedings*. [S.l.: s.n.], 2006. p. 211–220.

RYSER, J.; GLINZ, M. *SCENT: A Method Employing Scenarios to Systematically Derive TestCases for System Test*. [S.l.], 2000.

SEO, K. I.; CHOI, E. M. Comparison of five black-box testing methods for object-oriented software. In: *Software Engineering Research, Management and Applications, 2006. Fourth International Conference on*. [S.l.: s.n.], 2006. p. 213–220.

SHAFIQUE, M.; LABICHE, Y. *A Systematic Review of Model Based Testing Tool Support*. [S.l.], 2010.

SHARMA, M.; CHANDRA, B. Automatic generation of test suites from decision table - theory and implementation. In: *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on*. [S.l.: s.n.], 2010. p. 459–464.

SIMEON, E. e. D. *Software EPA*. 2010. Disponível em: <http://www.simeon.com.br>. Acesso em: 21.02.2012.

SODRé, U.; NETO, M. J. Q. *Ensino Superior: Álgebra: Relações*. 2004. Disponível em: <http://pessoal.sercomtel.com.br/matematica/superior/algebra/relacoes/relacoes-.htm>. Acesso em: 21.08.2011.

SOUZA, S. R. S.; BRITO, M. A. S.; SILVA, R. A.; SOUZA, P. S. L.; ZALUSKA, E. Research in concurrent software testing: a systematic review. In: *Proceedings of the Workshop on Parallel and Distributed Systems: Testing, Analysis, and Debugging*. New York, NY, USA: ACM, 2011. (PADTAD '11), p. 1–5. ISBN 978-1-4503-0809-0. Disponível em: <http://doi.acm.org/10.1145/2002962.2002964>.

SRIVASTAVA, P. R.; PATEL, P.; CHATROLA, S. Cause effect graph to decision table generation. *SIGSOFT Softw. Eng. Notes*, ACM, New York, NY, USA, v. 34, p. 1–4, February 2009.

TAI, K.-C.; PARADKAR, A.; SU, H.-K.; VOUK, M. A. Fault-based test generation for cause-effect graphs. In: *Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: software engineering - Volume 1*. IBM Press, 1993. (CASCON '93), p. 495–504. Disponível em: <http://dl.acm.org/citation-.cfm?id=962289.962329>.

TELCORDIA, A. T. S. *The AETG Web Service*. 2012. Disponível em: <http:/-/aetgweb.argreenhouse.com/>.

VALLESPIR, D. M.; HERBERT, J. Effectiveness and cost of verification techniques: Preliminary conclusions on five techniques. In: *Computer Science (ENC), 2009 Mexican International Conference on*. [S.l.: s.n.], 2009. p. 264–271.

VEGAS, S.; BASILI, V. A characterisation schema for software testing techniques. *Empirical Softw. Engg.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 10, n. 4, p. 437–466, oct 2005. ISSN 1382-3256. Disponível em: <http://dx.doi.org/10.1007-/s10664-005-3862-1>.

VIDAL, A. R. *Teste Funcional Sistemático Estendido: Uma Contribuição na Aplicação de Critérios de Teste Caixa-Preta*. Dissertação (Mestrado) — Universidade Federal de Goiás, Goiânia, 2011.

VIJ, K.; FENG, W. Boundary value analysis using divide-and-rule approach. In: *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on*. [S.l.: s.n.], 2008. p. 70–75.

WEYUKER, E. J.; JENG, B. Analyzing partition testing strategies. *Software Engineering, IEEE Transactions on*, v. 17, n. 7, p. 703–711, jul 1991.

WOOD, M.; ROPER, M.; BROOKS, A.; MILLER, J. Comparing and combining software defect detection techniques: a replicated empirical study. In: *Proceedings of the 6th European SOFTWARE ENGINEERING conference held jointly with the 5th*

ACM SIGSOFT international symposium on Foundations of software engineering. New York, NY, USA: Springer-Verlag New York, Inc., 1997. (ESEC '97/FSE-5), p. 262–277. ISBN 3-540-63531-9. Disponível em: <http://dx.doi.org/10.1145/267895.267915>.

ZELKOWITZ, M. V.; WALLACE, D. R. Experimental models for validating technology. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 31, p. 23–31, May 1998. ISSN 0018-9162.

ZHU, H.; HALL, P. A. V.; MAY, J. H. R. Software unit test coverage and adequacy. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 29, n. 4, p. 366–427, dec 1997. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/267580.267590>.

ZIELCZYNSKI, P. *Traceability from Use Cases to Test Cases*. 2006. Disponível em: http://www.ibm.com/developerworks/rational/library/04/r-3217/index.html?S_TACT=105AGX15&S_CMP=EDU, Acesso em 12.02.2012.

APÊNDICE A

Glossário

Neste glossário estão descritos termos e expressões cujas definições não constam no texto desta dissertação.

Caso de Teste - Um caso de teste define (formalmente) um conjunto específico de valores de entrada para o teste, as condições sob as quais o teste deve ser executado e os resultados esperados pela execução, identificados com a finalidade de avaliar um determinado aspecto de um item do sistema a ser testado, identificar e comunicar formalmente as condições específicas detalhadas que serão validadas para permitir a avaliação de determinados aspectos dos itens testados.

Cenário - A norma IEEE (1998) define cenário como: (A) Uma descrição de uma série de eventos que podem ocorrer concorrente ou sequencialmente; (B) Um relato ou sinopse de um curso de eventos ou ações projetadas; (C) Comumente usado para grupos de casos de teste; sinônimos são *script*, conjunto, ou suíte. Nesta dissertação, cenário de teste, relativamente a um critério ou técnica de teste, refere-se ao tipo de software em que este critério ou técnica é empregado para o teste deste software.

Critério de Teste - Um critério de teste é o que define quais propriedades precisam ser testadas para garantir a inexistência de erros (ZHU et al., 1997).

Custo do Teste - O custo associado à atividade de teste, normalmente, é medido pelo tempo gasto na execução de um dado conjunto de casos de teste, como por exemplo em Vallespir e Herbert (2009). No escopo deste trabalho, foi utilizada a classificação presente em Vegas e Basili (2005), onde o custo associado à utilização de um critério/técnica é medido pela quantidade de casos de testes gerados/selecionados.

Domínio de entrada - É o conjunto de todos os valores possíveis de entrada para uma variável ou um programa.

Domínio de saída - É o conjunto de todos os valores possíveis gerados a partir da execução de dada funcionalidade.

Dado de Teste - é qualquer elemento pertencente ao domínio da entrada ou da saída de uma variável ou sistema em teste.

Eficácia do teste - É determinada pela quantidade de defeitos revelados durante a execução do teste.

Eficiência do teste - É determinada pelo aumento quantidade de defeitos detectados e redução do custo associado ao teste.

Estudo de caso - trata-se do estudo de casos isolados, em que a análise deve ser feita com profundidade, detalhadamente e de forma exaustiva, considerando as influências internas e externas, conforme Zelkowitz e Wallace (1998).

Experimento - caracteriza-se pela manipulação das variáveis em estudo, com a finalidade de estabelecer relações de causalidade, conforme Zelkowitz e Wallace (1998).

Geração de casos de teste - É o processo de selecionar dados de entrada para o teste, definir condições de atendimento de acordo com o critério/técnica de teste utilizado na geração.

Geração de dados de teste - Escolha de dados de entrada que satisfaçam requisitos definidos por determinada técnica ou critério de teste.

PAF-ECF - PAF - Programa Aplicativo Fiscal, software responsável pelo gerenciamento da comunicação com a impressora fiscal (ECF - Emissor de Cupom Fiscal), (CONFAZ, 2010).

Qualidade do Teste - Qualidade é abordada como sinônimo da capacidade de detecção de defeitos do critério ou técnica de teste em análise.

Seleção de casos de teste - Procedimento para escolher casos de teste, a partir de um conjunto dado, para o teste de algum programa, de acordo com um critério definido.

Simulação - É a execução de uma tecnologia em um ambiente que simula o ambiente real em que a tecnologia será aplicada, de acordo com (ZELKOWITZ; WALLACE, 1998).

Survey - trata de uma investigação realizada em retrospectiva, em geral é conduzida quando algumas técnicas ou ferramentas já tenham sido utilizadas, conforme Zelkowitz e Wallace (1998).

APÊNDICE B

Síntese dos Trabalhos Selecionados

Nesta seção são apresentadas as informações extraídas de cada uma dos estudos primários analisados, de acordo com o esquema apresentado na Tabela 3.2, constante da Página 58, desta dissertação.

B.1 Estudo Primário 1 (EP1)

1. **Título e referência**: A Controlled Experiment in Program Testing and Code Walkthroughs/Inspections, (MYERS, 1978).
2. **Descrição sucinta**: Apresenta experimento que compara as técnicas de teste funcional, estrutural e a técnica de verificação Inspeção de Código. Testadores são divididos em grupos. Os Grupos **A** e **B** lidam com as técnicas funcional e estrutural, respectivamente, e os casos de testes derivados são executados com suporte computacional. O Grupo **C** lida com a inspeção da especificação e do código fonte do programa. O relato individual dos resultados de cada testador foi a base para a análise da efetividade das técnicas comparadas.
3. **Critério(s) de teste explorado(s)**: Não explora qualquer critério em particular, porém compara as abordagens de testes, representadas pelas técnicas citadas.
4. **Abordagem para o teste**: Geração de dados de teste. Geração manual de dados pelos testadores, visando à aplicação das técnicas citadas.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: A comparação é feita através de um experimento controlado, utilizando 59 programadores, dentre os quais 49 eram profissionais, porém todos com alguma experiência em teste de software.

7. **Comparação entre critérios de teste**: Compara as técnicas em relação à sua capacidade de detecção de defeitos, medida pelo total de defeitos detectados por cada uma das técnicas comparadas.
8. **Cenário de aplicação de cada critério**: Os testes foram aplicados em um programa com 15 defeitos em seu código-fonte, apresentado em (NAUR, 1969), cuja funcionalidade é a leitura de um arquivo texto e geração de um novo arquivo formatado de acordo com as regras presentes na especificação.
9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: Houve algumas combinações para verificar se haveria aumento de efetividade; por exemplo, a execução dos testes por mais de um testador aplicando a mesma técnica e utilizando técnicas diferentes (funcional e estrutural) e (funcional e inspeção).
11. **Síntese dos resultados e contribuições**: As três técnicas são iguais em termos de capacidade de detecção de defeitos. Os resultados verificados pelos testes aplicados por mais de um testador utilizando a mesma técnica ou técnicas diferentes foram mais efetivos, contudo não foram compensadores em termos de custo-benefício. Existe significante variabilidade entre testadores, tanto em termos de número quanto de tipo de defeitos encontrados. A análise de cada um dos defeitos mostrou que alguns tipos de defeitos foram mais difíceis de detectar (independentemente da técnica usada) e que a habilidade para detectar certos tipos de defeito varia, de alguma forma, de técnica para técnica.

B.2 Estudo Primário 2 (EP2)

1. **Título e referência**: Comparing the Effectiveness of Software Testing Strategies, (BASILI; SELBY, 1987).
2. **Descrição sucinta**: Compara critérios de teste funcional, estrutural e Técnica de Leitura de Código. A Leitura de Código foi aplicada para avaliar a especificação e abstrair suas funcionalidades, visando a posterior checagem do código-fonte.
3. **Critério(s) de teste explorado(s)**: Particionamento em Classes de Equivalência, Análise do Valor Limite, Cobertura de Comandos e Técnica de Leitura de Código com Abstração Gradual.
4. **Abordagem para o teste**: Geração de dados de teste. Cada testador é responsável por gerar, executar seus casos de testes e reportar os resultados.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.

6. **Classificação e descrição sucinta da análise realizada**: A comparação é feita através de um experimento controlado, utilizou-se o *fractional factorial design* Box et al. (1978) e análise fatorial de variância (ANOVA), em ambiente acadêmico e industrial (NASA e *Computer Science Corporation*). O experimento foi dividido em três fases, as quais foram subdivididas em outras cinco, envolvendo: treinamento, 3 sessões de teste e uma sessão de acompanhamento.
7. **Comparação entre critérios de teste**: A eficácia e o custo da detecção de defeitos foi auferida pela análise de variáveis dependentes: número e percentual de falhas detectados, tempo de detecção, dependência do tipo e da cobertura do programa e da expertise do programador. A caracterização dos tipos de defeitos foi analisada considerando a classificação definida por tipo (omissão e comissão) e por classe (inicialização, computação, controle, interface, dados e aparente *cosmetic*).
8. **Cenário de aplicação de cada critério**: Um programa de formatação de texto, o mesmo utilizado por Myers (1978); um programa para plotagem de um par ordenado (x,y), numa *grid*, dispostos nos eixos (x,y); uma implementação de um tipo de dados abstrato "lista"; e um programa para a mantutenção cadastral de um banco de dados (arquivo indexado) de referências bibliográficas.
9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: Os testadores experientes detectam mais defeitos e são mais eficientes quando utilizam Leitura de Código. O número de defeitos encontrados pela aplicação dos critérios funcionais foi maior que o encontrado pelos critérios estruturais. Testadores intermediários tiverem praticamente o mesmo desempenho com as três técnicas. Leitura de Código detecta mais defeitos de interface, ao passo que os critérios funcionais detectam mais defeitos do tipo controle. A utilização de Leitura de Código conduz a estimativas de detecção de defeitos mais exatas. Finalmente, de acordo com o número de defeitos detectados e o custo associado, Leitura de Código é tão eficaz quanto os critérios de teste funcional e estrutural. A eficácia, eficiência e custo dependem do tipo do programa em teste.

B.3 Estudo Primário 3 (EP3)

1. **Título e referência**: Cause-Effect Graphing Analysis and Validation of Requirements, (NURSIMULU; PROBERT, 1995).

2. **Descrição sucinta**: Trabalha com a relação causa-efeito, onde um conjunto de combinações de causas são derivados para cada efeito, tal que cada efeito e é sensibilizado para cada causa c, presente no seu conjunto de causas. Objetiva-se selecionar combinações de causas que aumentam sua importância sobre um efeito. É uma evolução da abordagem de Myers (1979).
3. **Critério(s) de teste explorado(s)**: Grafo de Causa e Efeito e Tabela de Decisão.
4. **Abordagem para o teste**: Geração de dados de teste. Grafo de Causa e Efeito é utilizado para a geração da Tabela de Decisão, a partir da qual são gerados os casos de testes.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica
6. **Classificação e descrição sucinta da análise realizada**: Foi desenvolvida uma análise teórica descrevendo o processo de geração da tabela de decisão a partir do grafo de causa e efeito, buscando superar os problemas ocorridos em Myers (1979).
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: Foi exemplificado em um hipotético cenário de transferência de arquivo em rede.
9. **Automação do teste**: Apenas foi citado o desenvolvendo uma ferramenta para automatizar a geração da Tabela de Decisão a partir do Grafo de Causa e Efeito.
10. **Utilização conjunta de critérios**: Grafo de causa e efeito é utilizado para a geração da Tabela de Decisão e, desta são gerados os casos de testes. Também é abordada a utilização do Grafo de Causa e Efeito na validação dos requisitos, descritos na forma de casos de uso. utilizando a forma: "**quando** <lista de causas> **então** <lista de efeitos>", os cenários de cada caso de uso são identificados e validados com a elaboração do Grafo de Causa e Efeito.
11. **Síntese dos resultados e contribuições**: Ressalta-se que os problemas encontrados em Myers (1979) foram atenuados, tais como: seja e um efeito dado por: $e = (a$ e $b)$ ou (textitc e $d)$. Refatorando temos $e = (a$ ou $c)$ e $(a$ ou $d)$ e $(b$ ou $c)$ e $(b$ ou $d)$. Em ambos os casos a semântica é a mesma, mas a sintaxe é diferente, o que leva a Grafos de Causa e Efeitos distintos, consequentemente a Tabelas de Decisão também diferentes, gerando casos de teste diferentes.

B.4 Estudo Primário 4 (EP4)

1. **Título e referência**: An Empirical Evaluation of Three Defect-Detection Techniques, (KAMSTIES; LOTT, 1995).
2. **Descrição sucinta**: Replica o estudo primário de Basili e Selby (1987), cuja análise consta da Seção B.2, desta dissertação. Esta replicação apresenta as seguintes diferenças em relação ao estudo primário replicado: (i) adiciona o passo do isolamento do defeito causador da falha detectada durante o teste; (ii) cobertura de 100% das condições lógicas do programa, ao invés da cobertura de todos os comandos.
3. **Critério(s) de teste explorado(s)**: Particionamento em Classes de Equivalência, Análise do Valor Limite, Cobertura de Condições e Leitura de Código com Abstração Gradual.
4. **Abordagem para o teste**: Geração de dados de testes. Cada testador é responsável pela geração, execução e reportagem dos resultados.
5. **Proposição de novo(s) critério(s) de teste**: não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: A replicação é efetuada através de um experimento, que foi executado por 50 estudantes, divididos em 3 grupos, durante 3 dias. Em cada dia, dois grupos testaram cada um dos programas utilizando cada uma das técnicas, de forma que no final todos os grupos testaram todos os programas, com todas as técnicas.
7. **Comparação entre critérios de teste**: Os critérios foram comparados para verificar se diferem em termos de eficácia, eficiência e custo. Busca-se definir qual a influência sofrida em virtude do tipo do defeito, da técnica e do testador, bem como qual a taxa de defeitos detectados e isolados.
8. **Cenário de aplicação de cada critério**: Os testes foram aplicados a três pequenos programas escritos na linguagem de programação C, muito parecidos com os utilizados no experimento de Basili e Selby (1987). Estes programas são os seguintes: (i) *ntree* que é uma implementação de um tipo de dado abstrato, uma árvore binária com ramificação ilimitada, (ii) *cmdline* que avalia um número de opções que são fornecidas via linha de comandos. As funões deste programa preenchem uma estrutura de dados com os resultados da avaliação e a imprime quando o preenchimento é concluído; (iii) *nametbl* que implementa outro tipo de dado abstrato, uma tabela simples. Os parâmetros de entrada para a execução das funcionalidades destes programas são obtidos a partir da leitura de arquivos textos.
9. **Automação do teste**: Não se aplica
10. **Utilização conjunta de critérios**: Não se aplica.

11. **Síntese dos resultados e contribuições**: Qualquer técnica pode ser tão efetiva quanto qualquer outra desde que o tempo e a experiência dos testadores em relação à linguagem de programação e em relação aos critérios de teste empregados, não sejam considerados aspectos importantes. Contudo se a eficiência é um fator a ser levado em consideração, a recomendação é pela utilização dos critérios de teste funcional, pois os testadores que os utilizaram foram os mais eficientes, detectaram/isolaram mais defeitos em uma quantidade menor de tempo.

B.5 Estudo Primário 5 (EP5)

1. **Título e referência**: An Empirical Analysis of Equivalence Partitioning, Boundary Value Analysis and Random Testing, (REID, 1997).
2. **Descrição sucinta**: Compara critérios de teste funcional com a Técnica de Teste Aleatório, aplicando uma metodologia experimental que contempla o ciclo de vida completo do teste e leva em consideração todas as possíveis entradas que satisfazem uma técnica de teste e todas as possíveis entradas potencialmente reveladoras de defeitos.
3. **Critério(s) de teste explorado(s)**: (i) Particionamento em Classes de Equivalência e suas variações: (EP - *Equivalence Partitioning* - 1:1) - onde apenas um caso de teste é selecionado por partição e (EP minimizada) - onde um conjunto mínimo de casos de testes é gerado para executar o maior número de partições possíveis; (ii) Análise do Valor Limite e suas variações: (BVA - *Boundary value analysis* - 1:1) e (BVA minimizada) e (iii) Técnica de Teste Aleatório.
4. **Abordagem para o teste**: Geração de dados de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: A comparação foi feita através de um experimento composto de cinco etapas: (i) Análise de defeitos, (ii) criação dos conjuntos de entradas detectoras de defeitos, (iii) criação dos conjuntos de casos de teste, (iv) derivação de probabilidades e (v) análise dos resultados. O item (ii) é criado a partir do resultado do item (i), a partir da identificação e entendimento de uma determinada falha é possível a identificação das potenciais entradas capazes de revelá-la. Os casos de testes (iii) são derivados pela mesma pessoa a partir das definições de cada técnica, de forma a minimizar o impacto do nível de experiência do testador no experimento.

7. **Comparação entre critérios de teste**: As técnicas foram comparadas em relação à probabilidade de detecção de defeitos, utilizando-se de cinco possíveis combinações entre elas:

   (a)(BVA 1:1) x (EP 1:1);
   (b)(EP 1:1) x (EP minimizada);
   (c)(BVA 1:1) x (BVA minimizada);
   (d)(EP 1:1) x teste aleatório e
   (e)(BVA 1:1) x teste aleatório.

   A probabilidade de detecção de defeitos é definida por $P_{DET} = m/d$. Onde d é o tamanho do domínio da entrada, m é a quantidade de entradas reveladoras de defeitos e P_{DET} é a probabilidade de uma entrada revelar um defeito. Esta entrada é selecionada aleatoriamente em m a partir de uma distribuição uniforme de probabilidades.
8. **Cenário de aplicação de cada critério**: Os testes foram efetuados em um sistema de controle de aviação escrito na linguagem de programação ADA, com aproximadamente 20.000 linhas de código, divididas em 17 módulos;
9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: Os resultados confirmam Análise do Valor Limite como o critério mais efetivo, alcançando a maior probabilidade de deteção de defeitos, 0,79 contra 0,33 do Particionamento por Equivalência. Contudo exige, em média, praticamento eo dobro da quantidade de casos de teste, 13,6 contra 7,6. A Técnica de Teste Aleatório requer uma quantiade de 50.000 casos de teste para alcançar o mesmo nível de efetividade que o critério Análise do Valor Limite.

    Resumindo, os resultados confirmam as expectativas constantes das hipóteses iniciais de que Análise do Valor Limite é mais efetiva que Particionamento em Classes de Equivalência, que é mais efetiva que Teste Aleatório.

B.6 Estudo Primário 6 (EP6)

1. **Título e referência**: Comparing and Combining Software Defect Detection Techniques: A Replicated Empirical Study, (WOOD et al., 1997).
2. **Descrição sucinta**: Replica o trabalho de Basili e Selby (1987), cuja análise consta da Seção B.2, desta dissertação. O pacote de programas organizado por Kamsties e Lott (1995) foi utilizado nesta replicação. O elemento novo

nesta replicação é a adição da hipótese de que o resultado do teste é influenciado pelo tipo de programa e pelo tipo do defeito.

3. **Critério(s) de teste explorado(s)**: Análise do Valor Limite, Particionamento em Classes de Equivalência, Cobertura de Condições e Técnica de Leitura de Código com Abstração Gradual.
4. **Abordagem para o teste**: Geração de dados de teste. Cada testador é responsável pela geração, execução e reportagem dos resultados.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: Foi descrito um experimento nos mesmos moldes dos estudos de Basili e Selby (1987) e Kamsties e Lott (1995), divido em 2 etapas (detecção e isolamento de defeitos), executado por 47 estudantes, divididos em 6 grupos, durante três semanas. No final todos os grupos testaram todos os programas com todas as técnicas.
7. **Comparação entre critérios de teste**: As técnicas e critérios foram comparados em relação à eficácia (quantidade de falhas observadas e defeitos isolados) e eficiência (eficácia dividida pelo tempo de detecção).
8. **Cenário de aplicação de cada critério**: Os testes foram aplicados aos mesmos programas descritos em Kamsties e Lott (1995).
9. **Automação do teste**: não se aplica.
10. **Utilização conjunta de critérios**: Não houve utilização conjunta, houve a combinação das três técnicas, que consiste em testar o mesmo programa com as três técnicas, o que apresentou uma melhora significativa em relação à detecção de defeitos.
11. **Síntese dos resultados e contribuições**: Os resultados obtidos confirmam aqueles apresentados pelos estudos de Basili e Selby (1987) e Kamsties e Lott (1995), além da conclusão de que a efetividade relativa de cada técnica de teste depende do programa e do tipo de defeito, o fator subjetividade também influencia na análise da efetividade de uma técnica de teste, pois dois testadores utilizando a mesma técnica podem não detectar os mesmos defeitos e que a efetividade em termos de detecção depende da combinação programa/técnica e da natureza dos defeitos. Porém, o fato novo é a observação da maior efetividade na detecção de defeitos apresentada pela combinação entre as técnicas analisadas.

B.7 Estudo Primário 7 (EP7)

1. **Título e referência**: Functional Testing, Structural Testing and Code Reading: What Fault Type do they Each Detect?, (JURISTO; VEGAS, 2003).
2. **Descrição sucinta**: Apresenta um experimento em que compara critérios de teste funcional, critérios de teste estrutural e técnica de leitura de código. Este experimento é dividido em duas partes, sendo a primeira uma replicação ao trabalho de Wood et al. (1997), investigando mais profundamente a observação ali contida de "que o relacionamento entre tipo de defeito e a efetividade da técnica de teste deve ser examinada mais detalhadamente", com o objetivo estabelecer uma taxonomia entre tipos de defeitos e técnicas de detecção. A segunda parte foi desenvolvida a partir dos resultados obtidos desta replicação, utilizando novas versões dos programas testados com o objetivo de confirmar ou negar tais resultados e chegar a conclusões que não foram possíveis na primeira parte.
3. **Critério(s) de teste explorado(s)**: Análise do Valor Limite, Particionamento em Classes de Equivalência, Cobertura de Condições e Técnica de Leitura de Código com Abstração Gradual.
4. **Abordagem para o teste**: Geração de casos de teste. Cada testador é responsável pela geração, execução e reportagem dos resultados.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: A replicação é feita através de um experimento, dividido em duas etapas, executado por 195 estudantes. Estes estudantes foram divididos em 12 grupos, onde cada membro de cada grupo testou um programa com uma das técnicas. Cada etapa do experimento utilizou uma versão diferente dos programas testados.
7. **Comparação entre critérios de teste**: As técnicas foram comparadas com o objetivo de determinar se sua eficácia na detecção de defeitos é dependente do tipo do programa e do tipo do defeito.
8. **Cenário de aplicação de cada critério**: Os programas testados foram os mesmos utilizados por Kamsties e Lott (1995). Um novo programa foi adicionado à replicação. Seu nome é *Trade* e sua finalidade é ler um arquivo com transações comerciais e gerar dados estatísticos sobre estas transações.
9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: Os principais resultados podem ser sumarizados em:

(a) A técnica de leitura de código sempre se comporta pior que os critérios de teste funcional e estrutural independentemente do tipo do defeito;

(b) Em relação aos critérios funcionais e estruturais, eles se comportam de forma idêntica;

(c) Houve diferença entre a quantidade de testadores que detectou o mesmo defeito em versões diferentes do mesmo programa, apontando assim a influência versão do programa na quantidade de defeitos detectados pelos testadores.

B.8 Estudo Primário 8 (EP8)

1. **Título e referência**: An Evaluation of Systematic Functional Testing Using Mutation Testing, (LINKMAN et al., 2003).
2. **Descrição sucinta**: Apresenta, exemplifica o uso e demonstra a efetividade de um novo critério de teste funcional, o Teste Funcional Sistemático, obtido pela composição dos critérios Particionamento em Classes de Equivalência e Análise do Valor Limite. Este novo critério visa a melhorar a geração de casos de teste e detecção de defeitos em relação aos critérios de teste que o compõem, sugerindo a geração de pelos menos dois casos de testes para as partições identificadas (válidas e inválidas). O estudo primário apresenta ainda um conjunto de diretrizes para o emprego do novo critério de teste na derivação de casos de teste para vários tipos de dados, dentre os quais números inteiros, números não inteiros (ponto flutuante), *strings* de caracteres, etc.
3. **Critério(s) de teste explorado(s)**: Particionamento em Classes de Equivalência e Análise do Valor Limite, Teste Funcional Sistemático.
4. **Abordagem para o teste**: Geração dos casos de testes.
5. **Proposição de novo(s) critério(s) de teste**: Teste funcional sistemático.
6. **Classificação e descrição sucinta da análise realizada**: É descrito um estudo de caso em que o Teste Funcional Sistemático é aplicado na geração de casos de teste para o programa *Cal*, um aplicativo de calendário do Sistema Operacional *Unix*. O conjunto de dados de teste gerado para este aplicativo, juntamente com outros 12 conjuntos de dados de testes gerados por estudantes, foram submetidos ao teste de mutantes gerados utilizando a ferramenta PROTEUM/IM 2.0. Delamaro et al. (2001).
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: Geração de casos de testes para o programa *Cal Unix programe*.

9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: O Teste Funcional Sistemático é resultado da utilização conjunta dos critérios de teste funcional Análise do Valor Limite e Particionamento em Classes de Equivalência.
11. **Síntese dos resultados e contribuições**: O conjunto de casos de teste gerados pelo Teste Funcional Sistemático conseguiu obter um *score* de eliminação de mutantes de 100%, resultado muito superior ao alcançado pelos outros onze conjuntos de casos de teste.
12. **Observações complementares**: Maiores detalhes sobre o TFS podem ser obtidos na Seção 2.3, na Página 30.

B.9 Estudo Primário 9 (EP9)

1. **Título e referência**: Requirements by Contracts allow Automated System Testing, (NEBUT et al., 2003).
2. **Descrição sucinta**: Apresenta um processo para automatização da geração de casos de testes a partir dos requisitos funcionais. O processo é constituído de três etapas: (i) formalização dos requisitos através de contratos expressos na forma de pré e pós condições, (ii) geração dos objetivos de teste (que são os cenários definidos pelas sequências de execuções definidas através de simulações auxiliadas pelo sistema de transição de casos de uso (UCTS - *Use Case Transition System*) Nebut e Fleurey (2003)) e (iii) geração dos casos de testes a partir dos objetivos de teste.
3. **Critério(s) de teste explorado(s)**: Teste Baseado em Casos de Uso.
4. **Abordagem para o teste**: Geração de casos de testes.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: Um estudo de caso é conduzido demonstrando a aplicação do processo à geração de casos de testes para um sistema de reuniões virtuais. A cobertura do teste é avaliada tendo como base os critérios definidos no sistema de transição de casos de uso.
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: O estudo de caso é aplicado a um sistema de reuniões virtuais *virtual meeting*.
9. **Automação do teste**: Todo o processo é suportado pelas ferramentas descritas em: Nebut e Fleurey (2003).
10. **Utilização conjunta de critérios**: Não se aplica.

11. **Síntese dos resultados e contribuições**: São duas as contribuições. A primeira é a apresentação da linguagem de contratos para requisitos funcionais expressos como casos de uso parametrizados. A segunda, proporcionar um método, um modelo formal e um protótipo de ferramenta para automaticamente derivar casos de teste a partir dos requisitos aprimorados com contratos.

B.10 Estudo Primário 10 (EP10)

1. **Título e referência**: Testing Software Components Using Boundary Value Analysis, (RAMACHANDRAN, 2003).
2. **Descrição sucinta**: Apresenta um método para testar componentes de software descrito a partir do Modelo Koala de Componentes Ommering et al. (2000), onde o modelo de componente é convertido em um modelo de objetos, levando em consideração suas interfaces, seus parâmetros de entrada e as funcionalidades providas. Os casos de teste são gerados a partir da especificação do componente mapeada em tabelas com os valores dos parâmetros de entrada e saída. O critério Análise do Valor Limite é empregado no teste dos limites destes parâmetros.
3. **Critério(s) de teste explorado(s)**: Análise do Valor Limite.
4. **Abordagem para o teste**: Geração de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: Foi desenvolvida uma análise teórica, apresentando o novo método, que faz parte do contexto de um projeto maior denominado "Testing Software Components", demonstrando com exemplos práticos os passos para a geração dos casos de teste.
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: o método apresentado destina-se ao teste de um componente *EUR-WST e US-WST teletex*, que interpreta, codifica/decodifica uma linguagem de codificação para componentes eletrônicos entre os padrões norte americanos e europeus.
9. **Automação do teste**: Utiliza a ferramenta de suporte StP/T Ommering et al. (2000) para a geração das tabelas e dos respectivos casos de teste.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: Como contribuição pode-se destacar a utilização do critério Análise do Valor Limite neste novo cenário. Em relação aos resultados, não foram apresentada a execução dos teste propriamente dita, não dispondo, desta forma de resultados concretos.

B.11 Estudo Primário 11 (EP11)

1. **Título e referência**: Partition Testing with Dynamic Partitioning, (CAI et al., 2005).
2. **Descrição sucinta**: Apresenta e exemplifica a utilização de um novo critério de seleção de casos de teste, o Particionamento Dinâmico, que é baseado no conceito de particionamento ideal e conveniente, isto é, sem intercessões entre as partições e com a capacidade potencial de detectar todos os defeitos presentes no domínio. Este novo critério particiona a suíte de testes em três conjuntos disjuntos (0, 1 e 2). Um caso de teste é selecionado a partir do conjunto 1, se encontrar um defeito é movido para o conjunto 2, senão para o 0. O defeito é corrigido e o programa testado novamente com o mesmo caso de teste. O processo continua até que o conjunto 1 esteja vazio.
3. **Critério(s) de teste explorado(s)**: Particionamento Aleatório, proposto por Cai et al. (2007), Particionamento Dinâmico e Teste Aleatório.
4. **Abordagem para o teste**: Seleção de dados de teste.
5. **Proposição de novo(s) critério(s) de teste**: Particionamento Dinâmico
6. **Classificação e descrição sucinta da análise realizada**: Dois experimentos são apresentados, onde o critério é comparado ao Teste Aleatório e ao Particionamento Aleatório na seleção de casos de testes para dois sistemas contendo um total de 36 e 28 defeitos, respectivamente.
7. **Comparação entre critérios de teste**: Este novo critério de seleção de casos de teste foi comparado com Teste Aleatório e Particionamento Aleatório, em relação ao custo (quantidade de casos de testes selecionados).
8. **Cenário de aplicação de cada critério**: Os testes foram aplicados aos programas: *Space Program* programa de exploração espacial desenvolvido pela Agência Espacial Européia e SESD um programa escrito em C++ com 3.179 linhas de código.
9. **Automação do teste**: Não se aplica
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: Este novo critério de seleção de casos de teste mostrou superioridade na capacidade de detectar a mesma quantidade de defeitos com uma quantidade de casos de testes bem menor, sendo que no experimento I houve uma redução de 31,73% e no experimento II 48,91%.
12. **Observações complementares**: Pode ser muito útil para execução de teste de regressão.

B.12 Estudo Primário 12 (EP12)

1. **Título e referência**: Automated Support for Test-Driven Specification, (JONES, 2005).
2. **Descrição sucinta**: Apresenta uma abordagem para a automação de testes baseados na especificação de requisitos do sistema. Esta abordagem é baseada no modelo W-*model* (GERRARD; THOMPSON, 2002), cujo objetivo principal é analisar a especificação de requisitos em relação a completeza e consistência, avaliar a adequação funcional do conjunto de teste e gerar o oráculo de teste. Especificação e testes são vistos como dois lados da mesma moeda - um descreve a intenção e o outro mostra se o software a satisfaz ou não.
3. **Critério(s) de teste explorado(s)**: Tabela de Decisão e Particionamento em Classes de Equivalência.
4. **Abordagem para o teste**: Geração e avaliação de dados de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não propõe um novo critério, mas utiliza a tabela de decisão como critério de cobertura, representando uma evolução ao critério de cobertura de decisões proposto por Binder (2000).
6. **Classificação e descrição sucinta da análise realizada**: A abordagem é validada através de uma simulação conduzida pelo autor exemplificando a aplicação a um sistema de pagamento de salários, onde os trabalhadores recebem por hora trabalhada, com cálculo de adicionais na forma de horas extras.
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: Tabela de Decisão é utilizada como uma linguagem de especificação de requisitos. Sua utilização é exemplificada na validação da especificação e geração de casos de testes para um sistema de pagamento de salários.
9. **Automação do teste**: É proposto um conjunto de ferramentas: TDST - *Test Driven Specification Toolset*, composto de:
   (a) *DTE - Decision Table Editor* que é a interface gráfica;
   (b) *DTAgen - Decision Table Analyzer Generator* que converte a Tabela de Decisão estática em um modelo executável equivalente (*C++ class*) e gera o
   (c) *FTA - Functional Test Analyzer* responsável pela análise de cobertura do teste.
10. **Utilização conjunta de critérios**: Utiliza Tabela de Decisão em conjunto com o citério Particionamento em Classes de Equivalência, onde a função

testada é divida em classes de equivalência representando cada regra constante da Tabela de Decisão.

11. **Síntese dos resultados e contribuições**: Fornece um *link* entre os estágios iniciais e finais do processo de desenvolvimento, ou seja especificação e os teste, de forma que satisfazendo um, satisfaz-se automaticamente o outro. Fornece um conjunto de ferramentas de suporte para a automação do processo.

B.13 Estudo Primário 13 (EP13)

1. **Título e referência**: Towards Describing Black-Box Testing Methods as Atomic Rules, (MURNANE et al., 2005).
2. **Descrição sucinta**: Apresenta um modelo para descrição das técnicas de teste funcional de forma atômica, cuja intenção é dar um caráter de objetividade ao entendimento de uma determinada técnica de teste funcional, diminuindo o fator subjetividade na interpretação da descrição da mesma. Sendo assim, as técnicas podem ser interpretadas da mesma forma por diversos testadores. O esquema para a decomposição em regras atômicas contém um conjunto de regras que definem os passos para a seleção da fonte de dados *DSSR - Data Set Selection Rule*; para a seleção de cada item de teste individualmente(*DISR - Data Item Selection Rule*) e para a contrução dos casos de teste (*TCCR - Test Case Construction Rule*.
3. **Critério(s) de teste explorado(s)**: Particionamento em Classes de Equivalência e Análise do Valor Limite.
4. **Abordagem para o teste**: Geração e seleção de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: Foi desenvolvida uma análise teórica demonstrando a simplicidade tanto do aprendizado quanto da utilização da metodologia proposta, contando com a participação de 33 estudantes que foram apresentados às duas formas de utilização dos critérios Análise do Valor Limite e Particionamento em Classes de Equivalência, a abordagem de Myers e Sandler (2004) e esta nova apresentada, Regras Atômicas.
7. **Comparação entre critérios de teste**: Critérios propriamente ditos não foram comparados, porém a forma de utilizar Análise do Valor Limite e Particionamento em Classes de Equivalência proposta pelos autores foi comparada com a forma proposta por Myers e Sandler (2004).

8. **Cenário de aplicação de cada critério**: Não é apresentada nenhum cenário de aplicação real da metodologia. A análise apresenta apenas a conceituação teórica.
9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: A nova abordagem torna mais consistente a assimilação dos conceitos relativos aos critérios Analise de Valor Limite e Particionamento em Classes de Equivalência, tendo em vista que 61% dos estudantes relataram que aprenderam a abordagem de Myers e Sandler (2004) primeiro, com 9% respondendo que a utilizarão no futuro. 39% aprenderam esta nova abordagem primeiro e 91% relataram que a utilizarão no futuro.

B.14 Estudo Primário 14 (EP14)

1. **Título e referência**: Generation of Test Cases from Functional Requirements. A Survey, (GUTIERREZ et al., 2006).
2. **Descrição sucinta**: É apresentada uma análise minuciosa de treze abordagens para a geração dos casos de testes a partir dos requisitos funcionais. São destacados os aspectos que estão e os que ainda não estão resolvidos em relação à geração de casos de teste a partir dos requisitos funcionais do software.
3. **Critério(s) de teste explorado(s)**: Teste Funcional, sem a especificação de um critério particular.
4. **Abordagem para o teste**: Geração de casos de testes.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: É descrito um survey, cuja análise dividiu os estudos selecionados em três grupos: (i) grupo 1 engloba os trabalhos que derivam os casos de teste diretamente da especificação de requisitos, (ii) grupo 2 engloba os trabalhos que geram um modelo comportamental a partir da especificação de requisitos e a partir deste modelo gera os casos de testes e (iii) gGrupo 3 engloba os trabalhos baseados no Método de Particionamento por Categoria, proposto por Ostrand e Balcer (1988).
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: Não se aplica.
9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: Não se aplica.

11. **Síntese dos resultados e contribuições**: Como contribuições podem ser destacados os aspectos que estão e os que ainda não estão resolvidos relativos a esta questão. Em relação aos resolvidos, pode-se afirmar que (a) os modelos comportamentais permitem a sistematização e automatização do processo (teste baseado em modelo). (b) a existência de ferramenta de suporte é um indicador de maturidade e melhora sua automatização. (c) Um aspecto interessante destacado pelos autores é a necessidade da derivação de casos de testes em 2 níveis (verificação do comportamento de um caso de uso isolada e individualmente e verificação se os casos de uso funcionam em conjunto). (d) A forma de definição de um caso de uso, para a derivação dos testes é outro aspecto importante.

    Em relação aos não resolvidos: (a)Falta de documentação - falta de referência a aplicações práticas ou estudos de casos realísticos. (b) nenhuma abordagem propõe algum tipo de métrica ou ferramenta para a avaliação da qualidade dos casos de testes gerados. (c) Nenhum dos autores mostrou que sua abordagem é melhor que os testes aleatórios ou teste que usam o senso comum (teste ad hoc). (d) o principal critério de cobertura utilizado nestas abordagens é a exploração combinatória de todos os cenários possíveis para os casos de uso. (e) a implementação dos casos de uso, pouco ou sequer foi citado pelos autores;
12. **Observações complementares**: Existem abordagens suficientes para adquirir uma ideia precisa de como derivar casos de testes, mas não existem ainda uma abordagem completa e integrada que descreve o processo como um todo. Uma abordagem desta natureza deve contemplar os seguintes itens: (i) construir um modelo comportamental, (ii) derivar cenários de teste a partir de um caso de uso, (iii) derivar cenários de teste a partir de vários casos de uso, (iv) gerar os casos de teste, (v) obter os cenários de teste, (vi) reduzir o número dos casos de teste sem perda em termos de cobertura, (vii) medir a cobertura, (viii) gerar os resultados esperados, (ix) ordenar os casos de teste para maximizar o critério de seleção (priorização de cenários de teste) e (x) construir os *scripts* de teste ou código de teste executável.

B.15 Estudo Primário 15 (EP15)

1. **Título e referência**: Avoiding Coincidental Correctness in Boundary Value Analysis, (HIERONS, 2006).
2. **Descrição sucinta**: Desenvolve uma análise sobre a utilização do critério de teste funcional Análise do Valor Limite, observando que devem ser usados

valores de entradas que possuem a capacidade de reduzir as chances de ocorrer a corretude coincidente, isto é, quando o sistema produz a saída esperada, mesmo não estando implementado corretamente. Isto pode ocorrer com certa frequencia quando do teste dos limites de um determinado subdomínio.

3. **Critério(s) de teste explorado(s)**: Análise do Valor Limite e Particionamento em Classes de Equivalência.
4. **Abordagem para o teste**: Diretrizes para a geração de casos de testes.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: É apresentada uma análise teórica mostrando como superar uma limitação do critério de teste. Esta limitação ocorre, por exemplo, para a entrada x se $x \in S_i$ e $x \in A_j$ para algum $i \neq J$, mas $f_i(x) = f_j(x)$. Onde S_i é um subdomínio definido na especificação, A_j sua implementação de forma incorreta e $f_i(x)$ e $f_j(x)$ são funcionalidades do sistema.
7. **Comparação entre critérios de teste**: A técnica Particionamento de Domínio é analisada em relação ao tipo de defeitos que é capaz de capturar. Estes defeitos podem ser: (i) **defeitos de computação** que ocorrem quando uma função errada é aplicada a algum subdomínio S_i na implementação. Este tipo de defeito é capturado pelas técnicas de testes de análise de particionamento de domínio; (ii) **defeitos de domínio** que ocorrem quando o limite entre dois subdomínios está implementado incorretamente. Este tipo de defeito é capturado pela técnica de teste Análise do Valor Limite.
8. **Cenário de aplicação de cada critério**: São apresentados alguns exemplos de cenários onde o problema pode ocorrer e consequentemente a abordagem é aplicável, dentre os quais: (i) cálculo do custo de unidades de água e energia para o consumidor num cenário de aquisição antecipada (pré-pago); (ii) Cálculo de desconto para venda de energia (antecipadamente) ao consumidor, de acordo com a quantidade desejada e (iii) cáculo de alguma função, cujos parâmetros de entrada e saída são valores não inteiros (ponto flutuante).
9. **Automação do teste**: Fornece diretrizes gerais para geração automática de casos de testes.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: A principal contribuição é em relação a geração automática de casos de testes, onde não é possível contar com a experiência do testador.

B.16 Estudo Primário 16 (EP16)

1. **Título e referência**: Automatic Test Generation: A Use Case Driven Approach, (NEBUT et al., 2006).
2. **Descrição sucinta**: Apresenta, descreve e exemplifica a utilização de uma nova abordagem proposta para automatizar a geração de casos de testes a partir de requisitos modelados como casos de uso. A abordagem utiliza a metodologia de Projeto por Contrato, aprimorada por Meyer (1992). O processo de automatização segue uma abordagem incremental, dividida em duas partes, na primeira descreve como formalizar os requisitos, estendendo os casos de uso com contratos que são definidos em uma linguagem lógica proposicional, que é utilizada para definir pré e pós-condições e parâmetros de entrada para os casos de uso. O resultado desta primeira parte são chamados de objetivos de teste. A segunda parte descreve a forma de gerar os testes a partir dos casos de uso estendidos de uma forma automática. Neste estudo primário, a abordagem de Nebut et al. (2003), constante da Seção B.9, é estendida para gerar casos de testes baseados no comportamento de cada requisito funcional.
3. **Critério(s) de teste explorado(s)**: Teste Baseado em Casos de Uso.
4. **Abordagem para o teste**: Geração de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: A abordagem é validada através de um estudo de caso descrevendo a aplicação da abordagem na automatização da geração dos casos de testes para um sistema de aviação embarcado. Todos os passos da adordagem: a formalização, a derivação dos objetivos de teste, a simulação dos casos de uso estendidos, a avaliação em relação aos critérios de cobertura, a geração dos casos de tese, etc. são descritos e analisados detalhadamente.
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: Os testes foram aplicados ao sistema *TAS - Thalès Airbone Systems*, embarcado em Avião Militar Francês denominado *Rafale*.
9. **Automação do teste**: Este é o objetivo principal do estudo. Todo o processo é suportado pelas ferramentas descritas em: Nebut e Fleurey (2003). Dentre estas ferramentas está o sistema de transição de casos de uso (*Use Case Transition System - UCTS*. Este sistema define um conjunto de cinco critério utilizados para medir o nível de cobertura do teste: (i) Todos as arestas, (ii) Todos os vértices, (iii) Todos os casos de uso instanciados, (iv) Todos os vértices e todos os casos de uso instancidos e (v) Todos os termos de préc-condições.

10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: As principais contribuições deste estudo primário são: (i) apresentação de uma abordagem que transfere o foco do esforço dispensado à geração de casos de teste para as atividades de especificação de requisitos e (ii) emprego desta abordagem na automatização da geração de casos de testes a partir da formalização dos requisitos de um sistema embarcado no contexto de softwares orientados a objetos.
12. **Observações complementares**: Os autores ponderam que ao invés de se utilizar diretamente um método formal, preferiram partir de práticas estabelecidas e a partir daí caminhar rumo a formalização. Este método é menos sofisticado que muitos métodos formais, no entando contém práticas e necessidades típicas da indústria não contempladas nos métodos formais.

B.17 Estudo Primário 17 (EP17)

1. **Título e referência**: Use Case-Based Acceptance Testing of a Large Industrial System: Approach and Experience Report, (ROUBTSOV; HECK, 2006).
2. **Descrição sucinta**: Apresenta, descreve e aplica uma abordagem, baseada em casos de uso, para teste de aceitação. Esta abordagem é composta de três níveis, sendo que no mais alto, os casos de uso são validados através da análise de cenários e nos mais baixos, *scripts* e casos de teste são aplicados. A abordagem é baseada na Norma IEEE 829 IEEE (1998) e possui os seguintes passos: preparação do teste, revisão da especificação de requisitos, avaliação do plano de teste, revisão dos *scripts* e casos de teste, execução do teste. Em adição, utiliza diagramas de casos de uso para a definição dos cenários de teste e diagramas de classes para mapear a rastreabilidade entre os requisitos e os seus respectivos testes.
3. **Critério(s) de teste explorado(s)**: Teste Baseado em Casos de Uso.
4. **Abordagem para o teste**: Geração de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: é conduzido um estudo de caso com a participação dos usuários do sistema, juntamente com parte da equipe de desenvolvimento, demonstrando a aplicação da abordagem ao teste de aceitação de um complexo sistema para a empresa de transporte público holandesa, descrito mais adiante nesta subseção.
7. **Comparação entre critérios de teste**: Não se aplica.

8. **Cenário de aplicação de cada critério**: O estudo de caso foi aplicado ao um sistema *web* denominado *E-Ticket system* para a venda *online* de passagens no sistema de transportes públicos na Holanda. Os consumidores compram as passagens através de cartões recarregáveis, emitidos especialmente para esta finalidade.
9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: O principal resultado é refletido na cobertura dos requisitos alcançada pelos testes, chegando a 94%.
12. **Observações complementares**: Abordagem baseada em casos de uso é uma técnica aplicável a testes de aceitação.

B.18 Estudo Primário 18 (EP18)

1. **Título e referência**: Comparison of Five Black-box Testing Methods for Object-Oriented Software, (SEO; CHOI, 2006).
2. **Descrição sucinta**: Apresenta um estudo comparativo entre cinco critérios de teste funcional, aplicáveis ao teste de sistemas desenvolvidos sob o paradigma da orientação a objetos. Teste Baseado em Casos de Uso deriva os casos de teste a partir dos cenários de casos de uso, com a adição de restrições. Teste a partir do Diagrama de Colaboração deriva os casos de teste a partir da definição da sequência de chamadas de operações executadas pelos objetos, definindo valores de entrada e saída. Testes utilizando Object-Z, primeiramente precisa obtê-los a partir da conversão dos mesmos utilizandos a linguagem de especificação Z e os casos de testes gerados a partir destes objetos. O domínio das funções é particionado e restrições são expressas utilizando OCL e os testes são gerados a partir dos atributos dos objetos constantes do domínio. Teste Baseado em Casos de Uso Estendidos são derivados para verificar os eventos de entrada e saída a partir de um (caminho "mensagem/método") de um cenário do caso de uso.
3. **Critério(s) de teste explorado(s)**: Teste Baseado em Casos de Uso, Teste a partir do Diagrama de Colaboração, Teste Baseado em Casos de Uso Estendidos, Teste a partir de requisitos convertiros em Objetc-Z e Teste a partir de requisitos formalizados com a linguagem de restrições *OCL - object constraint language*.
4. **Abordagem para o teste**: Geração de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.

6. **Classificação e descrição sucinta da análise realizada**: É apresentado um experimento onde cada um dos critérios foi aplicado ao teste de dois sistemas e os resultados reportados na forma de estatística em relação aos aspectos de comparação, basicamente a cobertura alcançada por cada um destes critérios.
7. **Comparação entre critérios de teste**: Os critérios foram comparados para verificar quais são os mais efetivos em termos de cobertura, onde a cobertura é obtida pela quantidade de itens executados em cada um dos cenários de teste.
8. **Cenário de aplicação de cada critério**: Estes critérios foram comparados durante o teste de dois sistemas: (i) um sistema de caixa eletrônico, onde se testa uma operação de retirada de dinheiro; (ii) um sistema de matrículas escolares, onde se testa a funcionalidade de inscrição de um aluno em um determinado curso.
9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: Os autores recomendam o uso combinado de: Teste Baseado em Casos de Uso Estendido e Teste a partir de requisitos formalizados com OCL, pelo fato de que estes critérios apresentaram os melhores níveis de cobertura em relação aos demais.
11. **Síntese dos resultados e contribuições**: Os resultados apresentaram os seguintes percentuais de cobertura:

    (a)para o primeiro cenário: (i) caso de uso simples - 24%, (ii) diagrama de colaboração - 44%, (iii) object-Z - 44%, (iv)OCL - 74% e (v) caso de uso estendido - 84%;

    (b)para o segundo cenário: (i) caso de uso simples - 41%, (ii) diagrama de colaboração - 46%, (iii) object-Z - 48%, (iv) OCL - 66% e (v) caso de uso estendido - 81%.

B.19 Estudo Primário 19 (EP19)

1. **Título e referência**: Traceability from Use Cases to Test Cases, (ZIELCZYNSKI, 2006).
2. **Descrição sucinta**: Apresenta e descreve um método, proposto pela IBM, para a derivação de casos de teste a partir dos requisitos funcionais, especificados na forma de casos de uso. Sua aplicação consiste da identificação dos cenários de execução do caso de uso (básico, alternativos, exceções); mapeamento em um diagrama de atividades; identificação dos cenários (que é uma instância do caso de uso); mapeameneto de cada um dos cenários em um grafo

representando cada caminho específico dentro do fluxo de execução; derivação dos casos de testes para cada cenário, de forma que seu grafo representativo seja todo percorrido; criação de uma matriz de rastreabilidade conectando cada cenário com seu respectivo caso de teste.

3. **Critério(s) de teste explorado(s)**: Teste Baseado em Casos de Uso.
4. **Abordagem para o teste**: Geração de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: Valida o método através de uma simulação em que exemplifica cada passo da abordagem aplicado ao cenário de venda de livros pela *internet*, desde a identificação do caso de uso, sua descrição, identificação dos cenários, criação do diagrama de sequência, grafos de cada cenários, derivação dos casos de teste e criação da matriz de rastreabilidade. No final as vantagens da utilização é apresentada através de uma lista de benefícios na seção de conclusão do trabalho.
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: Venda de livros *online* pela *internet*.
9. **Automação do teste**: Utiliza a ferramenta IBM® *Rational*® *RequisitePro* como suporte à execução da abordagem proposta.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: Os casos de testes sendo derivados de forma mais automática evita duplicações, melhora a cobertura, torna fácil o monitoramento do processo de execução, facilita o trabalho dos testadores, facilita o teste de regressão (quando necessário) e por último diminui o tempo do projeto.

B.20 Estudo Primário 20 (EP20)

1. **Título e referência**: A Case Study for Generating Test Cases from Use Cases, (GUTIERREZ et al., 2008).
2. **Descrição sucinta**: Aborda a geração de casos de testes a partir dos requisitos funcionais, utilizando a técnica de análise de cenários, num processo que consiste de gerar um diagrama de atividades a partir da descrição dos casos de uso, utilizando o algoritmo, proposto por Gutierrez (2006) e Gutierrez et al. (2007). A efetividade da técnica é verificada utilizando a análise de mutantes aplicada à especificação de casos de uso, utilizando 11 operadores de mutação e o padrão de defeitos em casos de uso, propostos por Binder em Binder (2000).

3. **Critério(s) de teste explorado(s)**: Teste Baseado em Casos de Uso.
4. **Abordagem para o teste**: Geração de dados de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: É apresentado um estudo de caso que aborda os passos para a geração de casos de teste, que passa pela descrição do caso de uso em arquivo XML, geração de um diagrama de atividades e geração dos casos de testes a partir das sequências de ações identificadas no diagrama. Mostra ainda a avaliação da cobertura alcançada pelos critérios de cobertura Todos os Cenários, Todos os Nós e Todas as Transições, conforme descrito mais adiante.
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: O estudo de caso apresenta a avaliação dos testes efetuados em dois sistemas: Um sistema de cadastro de um catálogo de *links online* na *internet*, identificado como sistema *WEB* e um sistema de cadastramento de anotações, utilizando a linha de comando como interface com o usuário, identificado como sistema CML.
9. **Automação do teste**: Utiliza a ferramenta **TestGen** desenvolvida por Gutierrez (2007) e que implementa os seguintes critérios de cobertura utilizados no estudo de caso: Todos os Nós, Todas as Transições e Todos os Cenários.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: Os resultados foram avaliados em relação a taxa de mortalidade de mutantes verificadas em cada cada um dos programas testados, em relação aos critérios de cobertura definidos.

    - Para o sistema *WEB*, do total de 92 mutantes gerados, o critério Todos os Cenários eliminou 76 mutantes, representando um taxa de mortalidade de 82,6%. O critério Todos os Nós eliminou 60 mutantes, representando uma taxa de mortalidade de 65,2% e o critério Todas as Transições eliminou 72 mutantes, representando uma taxa de mortalidade de 78,3%.
    - Para o sistema CML, do total de mutantes 53 gerados, os três critérios eliminaram a mesma quantidade de 45 mutantes, representando um taxa de mortalidade de 84,9%;.

B.21 Estudo Primário 21 (EP21)

1. **Título e referência**: Web Service Test Case Generation Based on Decision Table (Short Paper), (NOIKAJANA; SUWANNASART, 2008).

2. **Descrição sucinta**: Apresenta uma metodologia para a geração de casos de testes para *web services* a partir dos seus requisitos e descrição definida em WSDL-S, uma Linguagem de descrição semântica de *web services* e SWRL, uma Linguagem de regras semânticas para *web services*. A metodologia utilizada é composta pelos passos: a) Preprocessamento: etapa onde a especificação (lógica e semântica) do *web service* é analisada através dos seus documentos descritivos e as regras derivadas e mapeadas para cada operação constante da especificação; b) Análise da Tabela de Decisão: que contempla a geração das condições e ações e a definição das regras e c) Geração dos casos de teste: onde um documento XML contendo os casos de teste derivados da Tabela de Decisão é criado.
3. **Critério(s) de teste explorado(s)**: Tabela de Decisão.
4. **Abordagem para o teste**: Geração de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: Um exemplo de aplicação é dado através de uma simulação descrevendo seu uso na geração de casos de testes para um *Rectangle web service*, que recebe quatro parâmetros, inteiros representanto os lados, e retorna o tipo de retângulo formado por estes parâmetros.
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: Teste de *web services*.
9. **Automação do teste**: Um protótipo de ferramenta: *TAD - Testing by Automatically generate Decison Table* foi utilizado para auxiliar no processo de automatização da geração dos casos de teste.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: A utilização de Tabela de Decisão para este tipo de teste é uma contribuição, porém o maior ganho é econômico, uma vez que a quantidade de casos de testes gerados diminui ao passo que a cobertura, em relação aos requisitos não é afetada.

B.22 Estudo Primário 22 (EP22)

1. **Título e referência**: Boundary Value Analysis Using Divide-and-Rule Approach, (VIJ; FENG, 2008).
2. **Descrição sucinta**: Apresenta um novo algoritmo para a geração de casos de teste utilizando o critério de teste Análise do Valor Limite em cenário onde os valores de algumas variáveis são dependentes de valores ou de relacionamen-

tos com outras variáveis. Este algoritmo proposto segue os seguintes passos: a) identifica e classifica as variáveis em três conjuntos de variáveis independentes, dependentes e determinantes dos limites da variáveis dependentes; b) converte o conjunto de variáveis dependentes (pela combinação com as variáveis determinantes de limite) em um conjunto de variáveis independentes e aplica o critério de teste na sua forma tradicional.

3. **Critério(s) de teste explorado(s)**: Análise do Valor Limite.
4. **Abordagem para o teste**: Geração de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: Exemplifica a aplicação através de uma simulação utilizando o algoritmo na conversão das variáveis dependentes em independentes e a sua aplicação utilizando uma ferramenta para automação da geração do teste, em um problema de gerados de datas.
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: Os testes foram gerados para o problema de geração de datas, o , o *Next Date Problem*, abordado em Jorgensen (2001). Neste problema, a quantidade de dias é dependente do mês e do ano, quando este for bissexto.
9. **Automação do teste**: Descreve a implementação e exemplifica a utilização de uma ferramenta, destinada ao uso em ambiente *Web*. Esta ferramenta foi aplicada à geração de casos de teste para o problema descrito.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: Esta abordagem contribui para a superação de uma limitação do critério de Análise do Valor Limite, tornando possível sua utilização na geração de casos de teste para variáveis em cenários onde exista relacionamentos de dependência entre estas variáveis.

B.23 Estudo Primário 23 (EP23)

1. **Título e referência**: Cause Effect Graph to Decision Table Generation, Srivastava et al. (2009).
2. **Descrição sucinta**: Apresenta um novo algoritmo para a geração da Tabela de Decisão a partir do Grafo de Causa e Efeito, de forma que possa solucionar os problemas constantes de abordagens anteriores, como em Myers e Sandler (2004), cuja melhoria é trabalhada em Nursimulu e Probert (1995), Tai et al. (1993) e Paradkar et al. (1997), porém estas soluções propostas não são

eficientes, pois não geram todos os possíveis casos de teste. O novo algoritmo apresenta como diferencial o fato de trabalhar com combinações de causas e efeitos, atentando para o fato que um efeito pode ser igualmente uma causa para outro efeito. Desta forma desenvolve o algoritmo dando especial atenção às combinações de causas que conduzem a um determinado efeito e às combinações de efeitos que conduzem a outros efeitos.

3. **Critério(s) de teste explorado(s)**: Tabela de Decisão e Grafo de Causa e Efeito.
4. **Abordagem para o teste**: Geração de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: Exemplifica a aplicação do algoritmo através de uma análise teórica analisando os problemas constantes das abordagens anteriores e descrevendo os passos do algoritmo na geração da Tabela de Decisão que contempla todos os casos de testes possíveis.
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: A teoria do Grafo de Causa e Efeito, bem como a geração da tabela de decisão foi exemplificada através da especificação de um hipotético sistema de emissão de mensagens, onde cada mensagem tem uma ou mais condicionais para ser emitida.
9. **Automação do teste**: A implementação de uma ferramenta é citada, mas não é utilizada no exemplo utilizado para demonstrar a eficácia do algoritmo proposto.
10. **Utilização conjunta de critérios**: Não se aplica
11. **Síntese dos resultados e contribuições**: O novo algoritmo com complexidade ($O(n^2)$) apresenta-se como uma alternativa viável à abordagem de Mathur (2008), cuja complexidade é ($O(n^3)$).
12. **Observações complementares**: O objetivo deste novo algoritmo é validar tanto a especificação, quanto a implementação.

B.24 Estudo Primário 24 (EP24)

1. **Título e referência**: Effectiveness and Cost of Verification Techniques: Preliminary Conclusions on Five Techniques, (VALLESPIR; HERBERT, 2009).
2. **Descrição sucinta**: Apresenta um estudo comparativo entre cinco critérios e técnicas de teste funcional aplicadas ao teste de unidade de dois programas contendo treze defeitos classificados em relação à possibilidade da geração ou não de uma falha a partir destes defeitos. Esta classificação é: (i) PF -

potencial falha, o defeito pode gerar uma falha e (ii) NF - o defeito não var gerar uma falha. Dentre os cinco critérios e técnicas comparadas, Inspeção de Área de Trabalho (*Desktop Inspection*) é estática, desta forma, os defeitos são detectados a partir de inspeção no código fonte do programa. As demais são dinâmicas, exigindo a geração e a execução dos casos de teste. Este processo é auxiliado pela ferramenta JUnit, abordada em Massol e Husted (2003).

3. **Critério(s) de teste explorado(s)**: (i) Inspeção de Área de Trabalho *(Desktop Inspection) - identificada por **DI***, (ii) Particionamento em Classes de Equivalência e Análise do Valor Limite (abordadas conjuntamente como uma única estratégia de teste) e identificada por **EP**, (iii) Tabela de Decisão, identificada por **DT**, (iv) Caminho Linear Independente *(Linearly Independent Path), identificada por **LIP*** e (v) Cobertura de Múltiplas Condições *(Multiple Condition Coverage), identificada por **MCC***.
4. **Abordagem para o teste**: Geração de casos de teste. Cada testador é responsável pela geração, execução e reportagem dos resultados.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: Descreve um experimento executado por 17 estudantes do 4º período do curso de engenharia da computação, divididos em dois grupos com 3 componentes e dois grupos com quatro componentes. Os grupos com 3 componentes aplicaram DI, LIP e DT e os grupos com quatro MCC e EP. Os grupos com três componentes deveriam detectar 39 defeitos cada e os grupos com quatro componentes, 52 defeitos.
7. **Comparação entre critérios de teste**: Os critérios e técnicas foram comparados em relação a: (i) eficácia na detecção de defeitos, medida pela quantidade de defeitos detectada; (ii) custo de detecção, medido pelo tempo de execução de cada técnica/critério e (iii) eficiência, que é medida pela divisão da eficácia pelo tempo.
8. **Cenário de aplicação de cada critério**: Os testes foram aplicados a dois programas escritos em Java para o ordenamento de vetores, com ou sem elementos repetidos. Se existitem elementos repetidos, os programas devem informar a quantidade de repetições encontradas.
9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: Estão sintetizados na Tabela B.1

Tabela B.1: *Resultados da comparação*

técnica	quantidade de defeitos	tempo em minutos	eficiência defeitos por hora
DI	12	206,66	1,16
MCC	9	170	1,08
LIP	1	246,66	0,24
EP	14	198	1,06
DT	12	313	0,77

B.25 Estudo Primário 25 (EP25)

1. **Título e referência**: Automatic Generation of Test Suites from Decision Table - Theory and Implementation, (SHARMA; CHANDRA, 2010).
2. **Descrição sucinta**: Apresenta e exemplifica a aplicação de um novo método para a automática geração de casos de testes a partir da Tabela de Decisão. Este novo método é baseado em um *framework* genérico. A geração dos dados de testes utiliza a combinação entre critérios Particionamento em Classes de Equivalência, Análise do Valor Limite e Tabela de Decisão. As classes de equivalência são obtidas a partir do domínio de entrada do programa, uma Tabela de Decisão é gerada combinando as classes de equivalência de forma a identificar e eliminar potenciais redundâncias. A partir daí os testes são derivados normalmente a partir da Tabela de Decisão.
3. **Critério(s) de teste explorado(s)**: Tabela de Decisão, Grafo de Causa e Efeito, Particionamento em Classes de Equivalência e Análise do Valor Limite.
4. **Abordagem para o teste**: Geração de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: Valida o método através de uma simulação apresentando e exemplificando a utilização do *framework* genérico em um sistema de livraria que concede desconto aos clientes dependendo do tipo do cliente e da quantidade de itens adquiridos.
7. **Comparação entre critérios de teste**: Os critérios explorados são comparados em relação ao custo (quantidade de casos de teste gerados) e eficiência na detecção de defeitos.
8. **Cenário de aplicação de cada critério**: Aplica-se o Particionamento em Classes de Equivalência quando os dados de entrada são definidos em termos de faixa e conjunto de valores discretos. Aplica-se Análise do Valor Limite quando o programa é uma função de várias variáveis independentes. Aplica-se o Grafo de Causa e Efeito para a decomposição da especificação de requisitos em unidades lógicas para a validação destes requisitos e, por fim aplica a Tabela de Decisão para testar variáveis dependentes, isto é, o valor de uma é dependente do valor de outra.

9. **Automação do teste**: O *framework* proposto automatiza o processo de geração de casos de testes, utilizando as ferramentas: Junit Massol e Husted (2003) e Jess Hill (2003) que implementa a conversão DOM/SAX Deitel et al. (2001) para a linguagem de programação java.
10. **Utilização conjunta de critérios**: Utiliza conjuntamente os três critérios, agrupando comportamento das entradas e saídas em classes de equivalência que representam as regras e a tabela de decisão é gerada para contemplar a combinação destas classes.
11. **Síntese dos resultados e contribuições**: Em relação ao **custo** Tabela de Decisão apresenta melhor custo benefício pelo fato de gerar menos casos de teste. Em relação a **eficiência** Tabela de Decisão é mais eficiente que os outros dois, pois elimina casos de teste redundantes e acrescenta poderoso rigor lógico aos casos de teste gerados.
12. **Observações complementares**: o *framework* reduz a quantidade de casos de testes, eliminando redundâncias e garantindo um conjunto mínimo de testes necessários.

B.26 Estudo Primário 26 (EP26)

1. **Título e referência**: An Overview on Test Generation from Functional Requirements, (ESCALONA et al., 2011).
2. **Descrição sucinta**: Este estudo é extensão ao trabalho de Gutierrez et al. (2006) com a finalidade de responder à seguinte questão: *é possível gerar casos de testes a partir dos requisitos funcionais descritos de* ***maneira informal***? Os autores adicionam as seguintes questões complementares: *É possível obter um conjunto completo de casos de testes a partir dos requisitos funcionais? Quão fácil é obtê-lo? Quão automatizável pode ser este processo?*
3. **Critério(s) de teste explorado(s)**: Teste Funcional, sem a especificação de um critério em particular.
4. **Abordagem para o teste**: Geração de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Não se aplica.
6. **Classificação e descrição sucinta da análise realizada**: É descrito um survey em que 24 abordagens foram investigadas, tendo como ponto de partida as conclusões de três trabalhos anteriores: Ryser e Glinz (2000), Denger e Mora (2003) e Gutierrez et al. (2006).
7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: Não se aplica.

9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: Não se aplica
11. **Síntese dos resultados e contribuições**:

    - As conclusões apresentadas em Denger e Mora (2003) e Gutierrez et al. (2006) ainda permanecem válidas, no entanto, cada uma das abordagens analisadas, neste trabalho, apresentam seu próprio formato e template;
    - Não existe uma abordagem definitiva para resolver o problema da geração de casos de testes a partir dos requisitos funcionais, de forma satisfatória, o que implica na falta de evolução destas abordagens analisadas;
    - Os seguintes aspectos precisam ainda precisam ser melhorados: uso de padronização para entradas e saídas, aplicação de padrões e métodos mais formais para descrever o processo propriamente dito, a necessidade de resultados empíricos, a medição da possibilidade de automatização do processo e uma ferramenta de suporte eficaz.

B.27 Estudo Primário 27 (EP27)

1. **Título e referência**: Teste Funcional Sistemático Estendido: Uma Contribuição na Aplicação de Critérios de Teste Caixa-Preta, (VIDAL, 2011).
2. **Descrição sucinta**: Introduz um novo critério de teste funcional, o Teste Funcional Sistemático Estendido, que é uma extensão ao critério proposto por Linkman et al. (2003), analisado na Seção B.8, para contemplar os tipos de dados Data e Hora. Formaliza as diretrizes apresentadas por Linkman et al. (2003) para a geração de casos de testes, através de um conjunto de algoritmos, sendo definido um algoritmo para cada tipo de dado específico, facilitando tanto o entendimento quanto a aplicação do Teste Funcional Sistemático e do Teste Funcional Sistemático Estendido.
3. **Critério(s) de teste explorado(s)**: Teste Funcional Sistemtemático e Teste Funcional Sistemtemático Estendido.
4. **Abordagem para o teste**: Geração de casos de teste.
5. **Proposição de novo(s) critério(s) de teste**: Teste Funcional Sistemático Estendido.
6. **Classificação e descrição sucinta da análise realizada**: É uma dissertação de mestrado em que a extensão é apresentada e validada através de dois estudos de caso, que contemplam a geração de casos de testes para dois sistemas: um sistema *Web* voltado para apoiar a de Gestão Estratégica Si-

meon (2010) e outro para a geração de casos de teste para alguns requisitos do roteiro de testes do PAF-ECF Confaz (2010).

7. **Comparação entre critérios de teste**: Não se aplica.
8. **Cenário de aplicação de cada critério**: Sistema de Gestão Estratégia e Sistema Emissor de Cupom Fiscal (PAF-ECF).
9. **Automação do teste**: Não se aplica.
10. **Utilização conjunta de critérios**: Não se aplica.
11. **Síntese dos resultados e contribuições**: Em ambos os estudos de caso foram destacados a maior potencialidade para a detecção de defeitos a partir da aplicação do critério proposto.
12. **Observações complementares**: Maiores detalhes sobre o TFSE podem ser obtidos na Seção 2.4, na Página 32.

APÊNDICE C

Condução da Revisão Sistemática

C.1 Condução

A revisão sistemática foi conduzida por um período de 5 meses (10/2011 a 02/2012), de acordo com o planejamento apresentado nas seções anteriores. Ao todo, foram recuperados 4.440 trabalhos, que foram submetidos para as etapas de seleção preliminar, seleção final e extração de resultados. Nas próximas seções são apresentados mais detalhes das atividades realizadas, incluindo a estratégia adotada para construção das strings de busca e os resultados das buscas para cada uma das fontes selecionadas.

C.1.1 Seleção Preliminar

A seleção preliminar foi conduzida em três etapas:

1.construção das Strings de busca;
2.realização das buscas; e
3.seleção preliminar de trabalhos.

Essas três etapas são detalhadas nas próximas seções.

C.1.1.1 Construção das Strings de Busca

Para se definir as strings de busca, foram utilizadas as palavras-chaves e sinônimos identificados na Seção 3.1.3.1 do Capítulo 3 localizado na Página 51. Utilizando o operador lógico "ou" (OR) para integrar os termos-chave e seus respectivos sinônimos, e o operador "e" (AND) para integrar termos-chave diferentes, conforme é apresentado a seguir:

(black-box OR functional OR requirements-based OR specification-based) AND (software test*) AND (techniques OR criteria OR approaches OR methods)

C.1.1.2 Buscas Realizadas

As buscas foram realizadas utilizando máquinas de busca da IEEExplore e ACM Digital Library e eventualmente no Google Acadêmico (http:www.scholar.google.com.br/schhp?hl=pt-BR&tab=ws), quando um determinado estudo faz referência um outro considerado relevante e que não esteja disponível em nenhuma destas duas bibliotecas digitais. Desta forma, alguns estudos foram obtidos dos seguintes sites:

http://www.sciencedirect.com;
http://www.elsevier.com;
http://www.citeseer.ist.psu.edu.

As tabelas abaixo descrevem as strings utilizadas nas buscas. Estas *strings* estão descritas com seus respectivos quantitativos (retorno, exclusão, seleção), classificados por fonte de pesquisa, onde:

fonte = base de dados idexada onde a pesquisa foi efetuada;
retorno = quantidade de estudos retornados;
interseção = quantidade de estudos que já constam do retorno de pesquisa com outra *string* de busca;
subtotal = (retorno - interseção);
ce1 = quantidade de artigos eliminados pela aplicação do critério de exclusão 1;
ce2 = quantidade de artigos eliminados pela aplicação do critério de exclusão 2;
ce3 = quantidade de artigos eliminados pela aplicação do critério de exclusão 3;
ce4 = quantidade de artigos eliminados pela aplicação do critério de exclusão 4;
ce5 = quantidade de artigos eliminados pela aplicação do critério de exclusão 5;
ce6 = quantidade de artigos eliminados pela aplicação do critério de exclusão 6;
ce7 = quantidade de artigos eliminados pela aplicação do critério de exclusão 7;
ce8 = quantidade de artigos eliminados pela aplicação do critério de exclusão 8;
selecionados = (subtotal - (c1+c2+c3+c4+c5+c6+c7+c8).

C.1.1.3 Busca no IEEE

A *string* base foi integralmente processada pela máquina de busca da *IEEExplore*, não necessitando, assim, de qualquer ajuste adicional para adaptação a esta

máquina de busca. Contudo, em razão do baixo índice de estudos primários selecionados e em relação à particularidades de cada uma das questões de pesquisa, esta *string* básica foi desmembrada em outras *strings* capazes de contemplar estas particularidades. As Tabelas C.1, C.2 e C.3 apresentam as *strings* desmembradas para a busca em relação à Questão Primária.

C.1.1.4 Questão Primária

A Tabela C.1 apresenta os dados relativos à busca utilizando a primeira *string* definida para a questão primária.

Tabela C.1: *Primeira string de busca utilizada na fonte IEEE relativa à Questão Primária.*

(functional OR black-box OR specification-based OR requirements-based) AND software AND test* AND (techniques OR methods OR criteria OR approaches OR strategies) AND (compar* OR evaluat* OR asses*)												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	507	450	57	7	12	8	13	3	1	5	7	1

A seleção de apenas um estudo primário a partir da busca realizada com a utilização da primeira *string* definida para a questão primária não foi considerado suficiente para responder a esta questão de pesquisa. Desta forma, foi definido um novo refinamento para esta *string*, o qual consta da Tabela C.2.

Tabela C.2: *Segunda string de busca utilizada na fonte IEEE relativa à Questão Primária.*

"software test*" and compar*												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	469	64	405	49	85	61	97	20	8	36	49	0

O resultado da busca utilizando a *string* constante da Tabela C.2 não se mostrou satisfatório, pois dentre os estudos primários retornados, nenhum foi selecionado. Desta forma foi necessário um novo refinamento para a *string* relativa à questão primária. Este novo refinamento se mostrou mais eficaz, possibilitando a seleção de 4 estudos primários, conforme dados constantes da Tabela C.3.

Tabela C.3: *Terceira string de busca utilizada na fonte IEEE relativa à Questão Primária.*

compar* AND "software testing" AND (techniques OR methods OR criteria OR strategies OR approaches)												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	958	271	687	82	143	102	164	34	14	61	82	4

A busca na fonte IEEE em relação à Questão Primária retornou um total de 1934 estudos primários, dos quais 5 foram selecionados.

C.1.1.5 Questão Secundária 1

Os mesmos passos seguidos na busca relativa a Questão Primária foram adotados na busca relativas as questões secundárias, ou seja, quando os resultados das primeiras *strings* não são considerados satisfatórios, define-se outras mais abrangentes para o aumento da sensibilidade. Sendo assim, foram utilizadas três *strings* para a busca relativa à Questão Secundária 1, conforme apresentado nas Tabelas C.4, C.5 e C.6.

Tabela C.4: *Primeira string de busca utilizada na fonte IEEE relativa à Questão Secundária 1.*

(functional OR black-box OR specification-based OR requirements-based) AND (software AND (test OR testing)) AND (techniques OR methods OR criteria) AND (applying OR using) AND (scenarios OR situation OR condition)											
Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
45	0	45	5	10	7	11	2	1	4	5	0

Tabela C.5: *Segunda string de busca utilizada na fonte IEEE relativa à Questão Secundária 1.*

(functional OR black-box OR specification-based OR requirements-based) AND (software AND (test OR testing)) AND (techniques OR methods OR criteria) AND (applying OR using)												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	508	231	277	33	58	41	66	14	5	25	34	1

Tabela C.6: *Terceira string de busca utilizada na fonte IEEE relativa à Questão Secundária 1.*

(functional OR black-box OR specification-based OR requirements-based) AND software AND test* AND (techniques OR methods OR criteria AND approaches OR strategies)												
Fonte	Retorno	Interseção	subtototal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	604	1	603	72	127	91	145	30	12	54	72	0

A busca na fonte IEEE em relação à Questão Secundária 1 retornou um total de 1157 estudos primários, dos quais 1 foi selecionado.

C.1.1.6 Questão Secundária 2

Foram definidas duas *strings* de busca para a Questão Secundária 2, uma abordando genericamente a questão e a outra particularizando "especificação de

teste", os resultados são apresentados nas Tabelas C.7 e C.8 para a fonte **IEEE** e nas C.19 e C.20, para a fonte **ACM**:

Tabela C.7: *Primeira string de busca utilizada na fonte IEEE relativa à Questão Secundária 2.*

("software specification testing" or "software test specification" or "functional software test specification") AND (asses* OR evaluat*) AND (functional OR black-box OR specification-based OR requirements-based) AND (techniques OR methods OR criteria)												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	33	33	0	0	0	0	0	0	0	0	0	0

Tabela C.8: *Segunda string de busca utilizada na fonte IEEE relativa à Questão Secundária 2.*

"software test" AND "specification"												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	38	6	32	4	7	5	7	2	0	3	4	0

A busca na fonte IEEE em relação à Questão Secundária 2 retornou um total de 71 estudos primários, nenhum dos quais foi selecionado.

Totalizando a busca na fonte IEEE em relativamente às três questões de pesquisa, 8 *strings* de busca foram utilizadas. 3162 estudos primários foram retornadas e 6 foram selecionados.

Diante deste fraco desempenho na seleção de estudos primários, foi decidida a definição de uma *string* de busca para cada um dos critérios de teste funcional que são do interesse da revisão sistemática (definidos na Subseção 3.1.3.2, na página 52 desta dissertação). Desta forma, cinco novas *strings* de busca foram definidas. As *strings* e os resultados relativos à busca na fonte IEEE estão descritos nas tabelas C.9, C.10, C.11, C.12 e C.13.

C.1.1.7 *Strings* auxiliares - IEEE

Tabela C.9: *String de busca utilizada na fonte IEEE relativa ao critério de teste funcional Boundary Value Analysis.*

"boundary value analysis" and "software testing"												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	15	0	15	2	3	2	3	1	0	1	2	2

Tabela C.10: *String de busca utilizada na fonte IEEE relativa ao critério de teste funcional Cause-Effect Graph.*

"Cause-Effect Graph"												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	5	0	5	1	1	1	1	0	0	0	1	0

Tabela C.11: *String de busca utilizada na fonte IEEE relativa ao critério de teste funcional Decision Table*

"decision table" and "software testing"												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	11	2	9	1	2	1	2	0	0	0	1	2

Tabela C.12: *String de busca utilizada na fonte IEEE relativa aos critérios de particionamento de domínio.*

"partition testing" and "software testing"												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	44	13	31	4	6	5	7	2	1	3	4	1

Tabela C.13: *String de busca utilizada na fonte IEEE relativa ao Teste Baseado em Casos de Uso*

"use case" AND testing												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
IEEE	54	18	36	4	7	5	8	2	1	3	4	3

A adição destas 5 *strings* auxiliares possibilitou a seleção de 8 novos estudos primários, elevando para 14 o total selecionado para a fonte IEEE. Nas próximas subseções estão descritas as *strings* utilizadas na busca na fonte ACM.

C.1.1.8 Busca na ACM

As *strings* originais não foram processadas pela máquina de busca da ACM, sendo necessário o desmembramento, além daquele citado para a pesquisa individual de cada critério de teste funcional. Estas *strings* desmembradas estão identificadas nas próximas subseções.

C.1.1.9 Questão Primária

Tabela C.14: *Primeira string de busca utilizada na fonte ACM relativa à Questão Primária.*

(functional OR black-box OR specification-based OR requirements-based) AND (software AND (test OR testing)) AND (techniques OR methods OR criteria) AND (applying OR using) AND (scenarios OR situation OR condition)												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	0	0	0	0	0	0	0	0	0	0	0	0

Esta *string* não foi processada pela máquina de busca, sendo assim, foram definidas quatro strings de busca para a pesquisa relativa a esta questão, uma para cada sinônimo do termo critério de teste funcional, além daquelas definidas para a pesquisa de forma individualizada para cada critério. As próximas tabelas apresentam os resultados obtidos com estas novas *strings*:

Tabela C.15: *Segunda string de busca utilizada na fonte ACM relativa à Questão Primária.*

functional and "software test*" and (techniques or methods or criteria)												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	388	58	330	39	69	49	79	16	7	29	39	3

Tabela C.16: *Terceira string de busca utilizada na fonte ACM relativa à Questão Primária.*

black-box and "software test*" and (techniques or methods or criteria)												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	162	24	138	16	29	21	33	7	3	12	16	1

Tabela C.17: *Quarta string de busca utilizada na fonte ACM relativa à Questão Primária.*

specification-based and "software test*" and (techniques or methods or criteria)												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	87	13	74	9	15	11	18	4	2	7	9	0

Tabela C.18: *Quinta string de busca utilizada na fonte ACM relativa à questão primária.*

requirements-based and "software test*" and (techniques or methods or criteria)												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	30	4	26	3	6	4	6	1	1	2	3	0

As *strings* constantes das Tabelas C.14, C.15, C.16, C.17 e C.18 foram suficientes para a busca em relação à Questão Primária e Secundária 1. A busca efetuada com a utilização destas *string* recuperou um total de 667 estudos primários, dos quais 4 foram selecionados.

C.1.1.10 Questão Secundária 2

Tabela C.19: *Primeira string de busca utilizada na fonte ACM relativa à Questão Secundária 2.*

Abstract:"test* specification" or Abstract:"specification test*"												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	19	17	2	1	1	0	0	0	0	0	0	0

Tabela C.20: *Segunda string de busca utilizada na fonte ACM relativa à questão secundária 2.*

(Abstract:"software test") and (Abstract:"specification")												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	20	20	0	0	0	0	0	0	0	0	0	0

A busca na fonte ACM em relação à Questão secundária 2 retornou um total de 39 estudos primários, nenhum dos quais foi selecionado.

Igualmente à busca na fonte IEEE, a busca na fonte ACM necessitou da definição de um conjunto de *strings* auxiliares para sensibilizar a busca em relação aos critérios de teste funcional de interesse da revisão sistemática. As Tabelas C.21, C.22, C.23, C.24 e C.25, descrevem os resultados obtidos nas buscas com a utilização destas novas *strings*.

C.1.1.11 *Strings* auxiliares - ACM

Tabela C.21: *String de busca utilizada na fonte ACM relativa ao critério de teste funcional Boudary Value Analysis.*

"boundary value analysis" and "software testing"												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	37	5	32	4	7	5	8	1	0	3	4	0

Tabela C.22: *String de busca utilizada na fonte ACM relativa ao critério de teste funcional Cause-Effect Graph.*

"cause-effect graph" and "software testing"												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	9	1	8	1	1	1	1	0	0	0	1	2

Tabela C.23: *String de busca utilizada na fonte ACM relativa ao critério de teste funcional Decision Table.*

"decision table" and "software testing"												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	42	6	36	4	7	5	8	2	1	3	4	1

Tabela C.24: *String de busca utilizada na fonte ACM relativa aos critérios de teste de particionamento de domínio.*

("partition testing" or "equivalence partitioning") and "software testing")												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	130	20	110	13	23	17	26	6	2	10	13	0

Tabela C.25: *String de busca utilizada na fonte ACM relativa ao Teste Baseado em Casos de Uso*

(Abstract:"use case") and (Abstract:"software testing")												
Fonte	Retorno	Interseção	subtotal	ce1	ce2	ce3	ce4	ce5	ce6	ce7	ce8	Selecionados
ACM	12	4	8	1	2	1	2	0	0	1	1	0

A adição destas 5 *strings* auxiliares possibilitou a seleção de 3 novos estudos primários, elevando para 7 o total selecionado para a fonte IEEE.

As buscas realizadas nas fontes IEEE e ACM possibilitaram a seleção de 21 estudos primários. O conjunto completo de estudos primários selecionados totaliza 27. Os outros 6 restantes, 4 foram obtidos através de pesquisas diretas no *Google* acadêmico e outros 2 diretamente com professores da UFG.

C.1.2 Seleção Final

Nesta subseção são apresentadas os detalhes da seleção final dos estudos primários nas fontes IEEE e ACM.

C.1.2.1 Base eletrônica indexada IEEE

Na Figura C.1, a Fase 1 corresponde ao total de estudos primários retornados da base eletrônica ACM após a submissão das respectivas string de consulta (n=3.291). A Fase 2 corresponde ao total de estudos resultantes do processo de seleção preliminar (n=134), sendo n=3.157 excluídos pois o título ou resumo não atendiam o escopo das questões de pesquisa da RS. A Fase 3 corresponde ao total de estudos resultantes do processo de seleção final (n=32), sendo n=102 excluídos uma vez

que após a leitura completa dos referidos estudos, identificou-se que os mesmos não atendiam o escopo das questões de pesquisa da revisão sistemática. Finalmente, na Fase 4 foram eliminados ainda n=18 visto que após a avaliação dos estudos segundo os critérios de qualidade dos estudos primários definidos no planejamento da Revisão Sistemática, foram considerados de baixa qualidade e, desse modo, restaram n=14 estudos primários selecionados para extração e sumarização dos resultados.

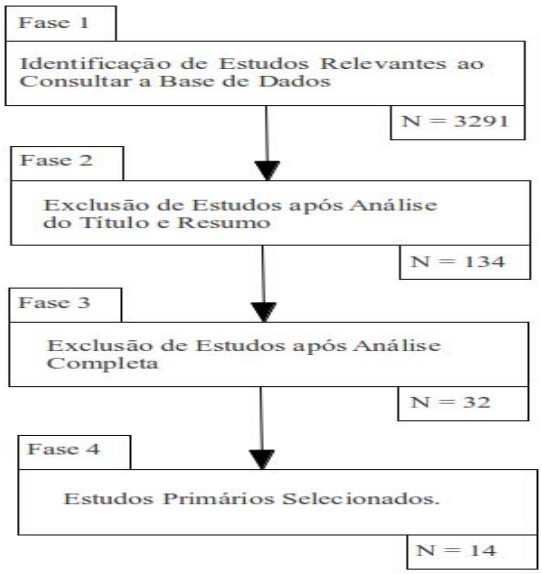

Figura C.1: *Seleção de Estudos Primários IEEE*

C.1.2.2 Base eletrônica indexada ACM

Na Figura C.2, a Fase 1 corresponde ao total de estudos primários retornados da base eletrônica ACM após a submissão das respectivas string de consulta (n=936). A Fase 2 corresponde ao total de estudos resultantes do processo de seleção preliminar (n=304), sendo n=632 excluídos pois o título ou resumo não atendiam o escopo das questões de pesquisa da Revisão Sistemática. A Fase 3 corresponde ao total de estudos resultantes do processo de seleção final (n=23), sendo n=281 excluídos uma vez que após a leitura completa dos referidos estudos, identificou-se que os mesmos não atendiam o escopo das questões de pesquisa da Revisão Sistemática . Finalmente, na Fase 4 foram eliminados ainda n=16 visto que após a avaliação dos estudos segundo os critérios de qualidade dos estudos primários definidos no planejamento da RS, foram considerados de baixa qualidade e, desse modo, restaram n=7 estudos primários selecionados para extração e sumarização dos resultados.

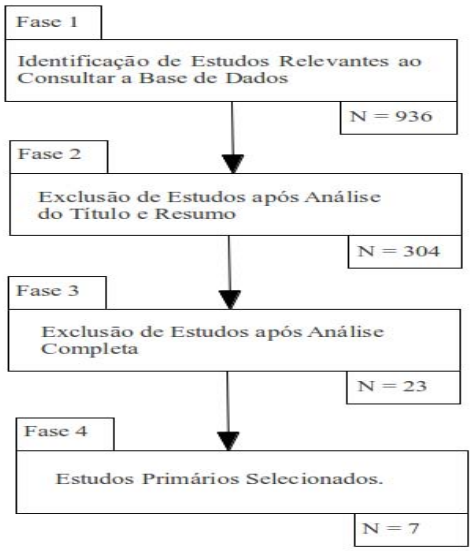

Figura C.2: *Seleção de Estudos Primários ACM*

Finalizando, 27 estudos primários foram selecionados, sendo:

- 14 na fonte IEEE;
- 7 na fonte ACM;
- 4 no *Google* Acadêmico e
- 2 diretamente da UFG.