**PODER EXECUTIVO**
**MINISTÉRIO DA EDUCAÇÃO**
**UNIVERSIDADE FEDERAL DO AMAZONAS**
**INSTITUTO DE COMPUTAÇÃO**
**PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA**


JARDELANE DE BRITO COSTA

# *TESTCHECK* – UMA ABORDAGEM BASEADA EM *CHECKLIST* PARA INSPECIONAR ARTEFATOS DE TESTE DE SOFTWARE

Manaus
2012

JARDELANE DE BRITO COSTA

# *TESTCHECK* – UMA ABORDAGEM BASEADA EM *CHECKLIST* PARA INSPECIONAR ARTEFATOS DE TESTE DE SOFTWARE

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas, como parte dos requisitos necessários para a obtenção do título de Mestre em Informática.

Orientador: Prof. Dr. Arilo Cláudio Dias Neto

Manaus
2012

JARDELANE DE BRITO COSTA

# *TESTCHECK* – UMA ABORDAGEM BASEADA EM *CHECKLIST* PARA INSPECIONAR ARTEFATOS DE TESTE DE SOFTWARE

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas, como parte dos requisitos necessários para a obtenção do título de Mestre em Informática.

Banca Examinadora

_____

Prof. Dr. Arilo Cláudio Dias Neto (Orientador)
Universidade Federal do Amazonas (UFAM)

_____

Prof. Dr. Guilherme Horta Travassos
Universidade Federal do Rio de Janeiro (COPPE/UFRJ)

_____

Prof. Dr. Rogério Patrício Chagas do Nascimento
Universidade Federal de Sergipe (UFS)

Manaus
2012

**CIP – CATALOGACÇÃO NA PUBLICAÇÃO**

Costa, Jardelane Brito.

TestCheck – Uma Abordagem Baseada em *Checklist* para Inspecionar Artefatos de Teste de Software / Jardelane de Brito Costa. Manaus: IComp da UFAM, 2012.

XVI, 98 f.: il.; 29,7 cm.

Orientador: Arilo Cláudio Dias Neto

Dissertação (mestrado) – Universidade Federal do Amazonas. Instituto de Computação, Manaus, BR – AM, 2012.

Referências Bibliográficas: p. 172-177.

1. Qualidade de Software. 2. Inspeção de Software 3. Testes de Software. 4. Artefatos de Testes. I. Dias-Neto, Arilo Cláudio. II. Universidade Federal do Amazonas, Programa de Pós-graduação em Informática. III. TestCheck – Uma Abordagem Baseada em *Checklist* para Inspecionar Artefatos de Teste de Software.

A Deus, meu alicerce, aos meus pais pelo apoio em todos os momentos, aos meus irmãos de sangue e em Cristo e aos meus filhos queridos e ao meu noivo pelo apoio incondicional.

# Agradecimentos

Em primeiro lugar a Deus, por sua infinita bondade e misericórdia. Tu és a minha paz, o meu consolo, minha rocha, o meu socorro. Tu és a minha força, refúgio e fortaleza. Sempre te adorarei. Obrigada Senhor, tudo posso em Ti que me fortalece.

Aos meus queridos e amados pais, que sempre acreditaram em mim, investiram em minha educação, quando nem eles tinham tido a oportunidade, mas tinham total convicção que era importante. Sempre me apoiando, aconselhando e incentivando para que a distância e a saudade não me fizessem desanimar ou desistir.

Ao professor Arilo Cláudio por me apresentar um mundo novo (Engenharia de Software/Teste de Software) por ter confiado e acreditado em minha capacidade de realizar esta pesquisa, pela excelente orientação e incentivo em todos os momentos. Muito obrigada professor pela oportunidade e por ter me ajudado a crescer como pesquisadora.

Aos meus amados irmãos de sangue e em Cristo por todo carinho e incentivo que sempre dispuseram a mim.

Ao meu amor, Douglas Miranda que aguentou firme e forte ao meu lado durante todo esse período. Obrigada pela paciência, carinho e amor. Seus conselhos estão bem guardados.

Aos professores da UFAM, em especial a professora Tayana Conte e o professor Ruiter Caldas, pelas palavras fortalecedoras que me fizeram pensar e continuar, afinal o que é uma batalha sem luta?

Aos alunos da disciplina Qualidade de Software e Introdução a Engenharia de Software pela participação nos experimentos realizados.

Aos profissionais do Instituo Nokia de Tecnologia e Desenvolvimento por aceitarem participar do estudo de observação realizado.

A minha avó que infelizmente não pôde participar deste momento, mas que acompanhou desde o inicio. Não pude ir me despedir da senhora, mas sei que sempre esteve ao meu lado. Obrigada por colocar no mundo minha mãezinha, a quem amo muito e é a fonte da minha inspiração.

Aos meus colegas do GRCM, ExperTS e IComp por terem me acolhido com todo carinho e amizade. Em especial a Viviane Gomes que foi colocado em meu caminho por Deus.

Aos meus amigos da UFOPA/UFPA, especialmente Rafael Brelaz e Rodolfo Wanderson pela amizade, carinho e ajuda. Mesmo distante contribuíram com minha pesquisa.

Ao CNPQ, pelo apoio financeiro.

Aos meus filhotes amados, Luma, Malu, Mel e Bob pelo carinho e amor dedicado a mim, sempre me fazendo tão feliz.

Aos meus sogros e cunhados pela oração, torcida e apoio. Vocês fazem parte da minha vida e da minha família.

As minhas incríveis amigas, Mary Lanne, Gabrielle Sá, Polianny Almeida e Láurian Melo, pelo carinho, longas conversas, apoio e amizade de todas vocês.

E a todos que me ajudaram direta ou indiretamente, todos vocês estão guardados com muito carinho.

Resumo da Dissertação apresentada à UFAM/AM como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática (M.Sc.)

*TESTCHECK* – UMA ABORDAGEM BASEADA EM *CHECKLIST* PARA INSPECIONAR ARTEFATOS DE TESTE DE SOFTWARE

Jardelane de Brito Costa

Março / 2012

Orientador: Arilo Claudio Dias Neto

A qualidade dos testes aplicados em um projeto de Software é um fator determinante para a qualidade do produto final. Isso se torna mais evidente em metodologias que se baseiam em testes para progressão do desenvolvimento do Software ou consideram os testes pontos centrais da avaliação da sua qualidade. Diversas estratégias podem ser aplicadas para avaliação da qualidade dos testes em um projeto de Software. Uma delas é garantir a qualidade dos artefatos de testes produzidos ao longo do processo de testes. Neste contexto, uma técnica que vem sendo bastante utilizada para garantir a qualidade de artefatos de Software é inspeção.

Existem diversos trabalhos que abordam inspeção em documentos produzidos ao longo do processo de desenvolvimento de software. No entanto, não foram encontrados na literatura técnica trabalhos que utilize inspeção em artefatos de Teste de Software. O que existe são trabalhos que mencionam a importância de se fazer inspeção nesses artefatos.

Assim, com o objetivo de avaliar a qualidade dos artefatos de teste, proporcionando que defeitos sejam encontrados antes da execução dos testes, esta dissertação apresenta uma abordagem baseada em *Checklist*, *TestCheck*, para inspeção de artefatos de teste de Software (planos, casos e procedimentos de teste). Para avaliar a abordagem proposta, foram realizados estudos experimentais (estudo de viabilidade, segundo estudo de viabilidade e estudo de observação na indústria). Eles analisaram e indicaram a eficácia, eficiência e amadurecimento da abordagem proposta em relação à detecção de defeitos em artefatos de teste.

**Palavra Chave:** Inspeção de Software, Teste de Software, Artefatos de Teste.

Abstract of Thesis presented to UFAM/AM as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

*TESTCHECK* – A *CHECKLIST*-BASED APPROACH FOR INSPECTION OF SOFTWARE TESTING ARTIFACTS

Jardelane de Brito Costa

March / 2012

Advisor: Arilo Claudio Dias Neto

The quality of testing applied to a software project is an important factor to reach the quality of the final product. This becomes more evident in methodologies based on testing to measure the software development progress or that consider testing the central element in the evaluation of its quality. Several strategies can be applied to evaluate the quality of testing in a software project. A possible strategy is to ensure the quality of artifacts produced during testing process. In this context, a technique that has been largely used to ensure the quality of software artifacts is inspection.

There are many works dealing with inspection of documents produced during the software development process. However, we have not identified in the technical literature works that apply inspection in software testing artifacts. We identified just studies indicating the importance of applying inspection in these artifacts.

Thus, with the purpose of evaluating the quality of testing artifacts, contributing to defects detection before tests execution, this thesis presents a checklist-based approach, *TestCheck*, for inspection of software testing artifacts (test plans, cases, and procedures). In order to evaluate the proposed approach, experimental studies (feasibility, observation and case studies) were conducted. They analyzed and indicated the efficacy, efficiency, and maturity of the proposed approach in relation to detection of defects in testing artifacts.

# ÍNDICE

# LISTA DE ABREVIATURAS E SIGLAS

V&V - Verificação e Validação

TDD – *Test Driven Development*

XP – *eXtreme Programming*

VV&T – Verificação, Validação e Teste

RUP – *Rational Unified Process*

LTR – *Level Test Report*

Mobile-D™ – Método de Desenvolvimento de Software para Dispositivos Móveis

CMMI – Capability Maturity Model Integration

MPS.BR – Melhoria de Processo do Software Brasileiro

ADS – *Advertisements*

*OUTLIER* - Em estatística, um outlier é uma observação que está numericamente distante quando comparada aos demais dados de um conjunto observado.

*BASELINE*- Diretrizes e resultados que formam uma base de dados para futuros estudos.

# LISTA DE FIGURA

# LISTA DE TABELAS

# CAPÍTULO 1: INTRODUÇÃO

*Neste capítulo serão apresentados o contexto do trabalho, o que motivou esta pesquisa e a questão de investigação. São também apresentados os seus objetivos, a metodologia de pesquisa adotada, o histórico deste trabalho e a organização deste texto.*

## 1.1. Contextualização e Motivação

O desenvolvimento de Software quando acontece de forma eficaz garante qualidade no produto construído. Isso pode ser evidenciado pela ausência de falhas por um longo período de tempo, proporcionando ao usuário ou cliente satisfação ao utilizar tal Software. Com isso, o objetivo a ser almejado durante o processo de desenvolvimento é construir Software com qualidade, mas para isso é necessário à aplicação de abordagens de Engenharia (PRESSMAN, 2006).

Obter qualidade nos processos e produtos de Engenharia de Software não é uma tarefa trivial, porém necessária. Nesta era tecnológica, em muitos produtos comercializados já vem embutido Software, exigindo dessa forma produtos com qualidade (ROCHA et.al., 2001). A Engenharia de Software utiliza-se de métodos e atividades visando apoiar a qualidade de Software. Dentre essas atividades, podem ser citadas as atividades de Verificação e Validação (V&V), que tem por objetivo analisar se o software está de acordo com suas especificações e assegurar que este atenda às expectativas do cliente (SOMMERVILLE, 2007). Dentre as técnicas mais tradicionais e eficientes relacionadas à V&V, podem ser citadas as técnicas de Inspeção e Teste de Software.

O contexto deste trabalho está relacionado principalmente ao cenário da aplicação de testes em projetos de Software. Teste de Software consiste em uma investigação experimental conduzida para prover informações aos usuários e envolvidos no processo sobre a qualidade do Software em teste no contexto no qual este será operado. Isso inclui, mas não consiste apenas, um processo de executar um programa com a intenção de revelar falhas, apresentando uma análise dinâmica do produto, com atividades planejadas previamente e de modo sistemático (KANER, 2006).

A busca por qualidade de Software fez com que diversas técnicas, ferramentas e estratégias de automação fossem desenvolvidas para tornar os Testes de Software mais eficientes. Apesar da grande variedade de soluções propostas, o desafio fundamental do Teste de Software é revelar falhas em um software a ser desenvolvido, e na prática ainda é

largamente dependente do desempenho de testadores a partir da execução de testes manuais (ITOKEN et al., 2007). Ainda assim, os testes apresentam bons resultados quanto à qualidade, porém poderiam ser otimizados com o uso de ferramentas para apoiar os testadores.

Tendo em vista a importância dos testes como atividade de garantia da qualidade do software (evitando que software sejam lançados no mercado inadequadamente) e os problemas enfrentados por conta de erros humanos, surge a necessidade de novos mecanismos que auxiliem os testadores neste processo visando à qualidade dos testes em um projeto de software. Diversas estratégias podem ser aplicadas para apoiar a obtenção de qualidade durante as atividades de teste de software, sendo uma delas as atividades de inspeção de software (FAGAN, 1976). Esta técnica é um método de revisão de software que se destaca por ser rigoroso, formal e executado com o objetivo de examinar artefatos de software, para encontrar defeitos antecipadamente (FAGAN. 1976).

O uso de técnicas de Inspeção de Software pode ajudar na melhoria da qualidade dos artefatos produzidos ao longo do processo de Testes de Software que, por conseguinte, trará melhores resultados para o software final. Devido à sua relevância nos processos de desenvolvimento de software, postulou-se que a qualidade dos testes pode ser assegurada através de inspeções de software como um complemento para a atividade de revisão informal (LANUBILE, 2007). A principal premissa ao buscar pela identificação de defeitos de forma antecipada é a de que quanto mais cedo no processo de desenvolvimento os defeitos forem encontrados, menor será o custo de correção e maior a probabilidade de solucioná-los, podendo ser até 10 vezes menos custoso que a identificação de defeitos com o software já em produção (BOEHM e BASILI, 2011).

As técnicas de inspeção se dividem, basicamente, em três categorias com características distintas: (1) *Ad Hoc* que se baseia no conhecimento e experiência do testador por não possuir nenhum mecanismo de apoio, (2) *Checklist* que consiste no processo de revisão apoiado por um questionário de perguntas e (3) Técnica de Leitura, que além de auxiliar na identificação de defeitos, auxilia também na forma como os artefatos devem ser revisados (lidos) através de diretrizes formais para que os defeitos sejam identificados (BARCELOS, 2006). No decorrer deste trabalho, será detalhado e justificado o tipo de técnica de inspeção que será adotado para o desenvolvimento desta pesquisa.

## 1.2. Descrição do Problema

Apesar dos avanços em Engenharia de Software na utilização de métodos e técnicas para obtenção de softwares com qualidade, um software ainda precisa ser testado antes de ser entregue ao cliente. Neste contexto, teste de software se apresenta como um dos meios mais eficazes para assegurar a qualidade de um software, bem como uma das áreas mais complexas e estudadas em engenharia de software (LUO, 2010). O objetivo de testar é revelar falhas antecipadamente, evitando que o cliente as encontre quando o software já estiver em uso, proporcionando, assim, maior qualidade do produto.

Antes de prosseguir, é importante apresentar a definição de alguns termos que serão utilizados constantemente neste trabalho (DELAMARO et al., 2007):

- Defeito (*fault*) é um passo, processo ou definição de dados incorretos;
- Engano (*mistake*) é ação humana que produz um defeito;
- Erro (*error*) é o resultado gerado pela ocorrência de um defeito;
- Falha (*failure*) é a consequência do erro, ou seja, durante a execução do programa, ocorre um erro podendo levar a uma falha.

Os testes quando conduzidos sem objetivo, planejamento e técnicas adequadas podem apresentar consequências desagradáveis, pois a descoberta de defeitos em uma fase tardia aumenta os custos para correção, provoca atrasos de cronograma, insatisfação do cliente, dentre outros aspectos negativos. Um estudo publicado em (BOEHM et al., 2001) indica que o custo para corrigir uma falha revelada após a codificação seria pelo menos 10 vezes mais caro que sua correção no início do processo, e o custo para corrigir falhas detectadas com o software já em produção seria pelo menos 100 vezes mais caro.

Para se atingir os objetivos traçados pelas atividades de teste de software, é preciso seguir um processo que gerencie as tarefas desde o seu planejamento até a análise dos seus resultados. No entanto, apenas seguir um processo não garante a sua qualidade. A qualidade dos testes está diretamente relacionada à possibilidade de revelar falhas em um software. Se os artefatos produzidos ao longo do processo de testes (ex.: plano, casos e procedimentos de teste) apresentam defeitos, o objetivo de prover testes com qualidade dificilmente será atingido e resultado final dos testes não será satisfatório (ITOKEN et al., 2007).

Teste de software é considerado um processo baseado na concepção, produção, seleção e otimização de um conjunto de casos de teste para testar um determinado sistema ou funcionalidade (ITOKEN et al., 2007). Ao revelar falhas (objetivo principal), acarreta melhorias de qualidade ao produto desenvolvido. Portanto, é necessário testar e

realizar bons testes para que as metas definidas ao longo do processo de desenvolvimento sejam alcançadas e o produto seja entregue de acordo com suas especificações.

Neste contexto, se não for garantida a qualidade dos testes, prevenindo eventuais enganos cometidos pelos profissionais durante o processo de testes, como garantir a qualidade do produto final? Então é necessário se precaver de alguma forma. Vários mecanismos podem ser empregados, dentre os quais podemos citar aplicação de técnicas de testes adequadas ao cenário do projeto ou fornecendo treinamento aos profissionais constantemente a fim de melhorar sua produtividade durante os testes. Porém, ainda não há a garantia de que aplicando tais tarefas a qualidade esperada dos testes seja atingida. Nesse cenário, aplicar técnicas de inspeção nos artefatos produzidos ao longo do processo de testes pode resultar em melhor qualidade dos testes produzidos ao longo de um projeto de software.

Este trabalho teve como objetivo a integração de técnicas de inspeção no processo de testes de software a fim de avaliar a qualidade dos artefatos produzidos ao longo deste processo, como será descrito nas próximas seções.

## 1.3. Hipótese

Se usarmos uma técnica de inspeção para revisar os artefatos de testes, poderemos aumentar a qualidade dos testes, pois será possível encontrar defeitos nos artefatos produzidos ao longo deste processo antes da sua execução. Isto reduziria o número de testes inadequados que seriam executados em um software que está sendo desenvolvido, e com isso aumentaria a efetividade e qualidade dos testes, que consequentemente irá impactar positivamente na qualidade do produto final.

## 1.4. Justificativa

O teste é considerado uma das atividades mais importante em desenvolvimento de software. A complexidade crescente dos softwares torna difícil a sua execução, o que pode afetar diretamente a qualidade do produto. Segundo PRESSMAN (2006), a qualidade é avaliada pela aplicação de uma série de revisões técnicas que examinam os elementos do modelo de projeto e pela aplicação de um processo de teste. Portanto, para execução de testes efetivos, é necessário que a qualidade dos casos de teste possa ser assegurada. Uma alternativa para a obtenção da qualidade nas atividades de teste é avaliar a qualidade dos artefatos produzidos ao longo do seu processo através de técnicas de inspeções de software (RODRÍGUEZ et al., 2010).

Dentre os cenários particulares onde a avaliação da qualidade dos artefatos de teste se torna essencial antes da execução propriamente dita dos testes, podem ser

citados o paradigma de desenvolvimento de software chamado Desenvolvimento Dirigido por Testes (TDD) (do inglês *Test-Driven Development*) (BECK, 2003) e algumas metodologias ágeis, tais como XP (*eXtreme Programming*) (BECK et al., 2005) e Scrum (SCHWABER, 1995).

Em TDD, o desenvolvimento do software é direcionado pelos testes especificados de forma antecipada. Desta forma, os testes são considerados o "oráculo" do processo de desenvolvimento, indicando os requisitos a serem atendidos no software a ser desenvolvido (BECK, 2003). Por isso, inspecionar estes artefatos do processo de testes pode garantir maior segurança e confiabilidade no software, encontrando defeitos antes de sua implementação. Em metodologia ágeis (ex: XP e Scrum), o conceito de *tests first* (testes primeiro) (AGILE MANIFESTO, 2001) normalmente é empregado. Nesta abordagem, desenvolvedores constroem os testes para uma funcionalidade antes de sua implementação, e após sua construção executam tais testes visando avaliar se o código construído é aprovado em tais testes previamente escritos. Dessa forma, garantir a qualidade dos testes escritos antes da implementação da funcionalidade pode ser uma tarefa importante visando o aumento da qualidade do produto desenvolvido.

No entanto, esta pesquisa não se restringe apenas a esses cenários, mas em qualquer cenário de desenvolvimento/manutenção de software onde se deseja obter melhorias na qualidade dos testes de software a partir da avaliação dos artefatos produzidos ao longo do processo de testes.

## 1.5. Objetivos

Nesta seção serão apresentados os objetivos desta pesquisa.

### 1.5.1. Objetivo Geral

Demonstrar que a utilização de uma técnica para inspeção de artefatos de teste produzidos ao longo do processo de testes de software pode influenciar positivamente na qualidade dos testes aplicados em um software (o que consequentemente impactaria a qualidade do produto final) a partir da detecção de um maior número de defeitos existentes nestes artefatos quando comparado a outras técnicas de inspeção.

### 1.5.2. Objetivos Específicos

- Identificação e disponibilização de um conjunto de regras que indicam como elaborar artefatos de teste de software com qualidade.
- Desenvolvimento de uma estratégia, baseada em *Checklist*, para apoiar inspeção em artefatos produzidos ao longo do processo de teste avaliado a partir

de estudos experimentais, demonstrando que este aumenta a taxa de detecção de defeitos quando comparado a outras técnicas de inspeção (ex: *Ad Hoc*).

- Elaboração de um pacote de experimentos a serem utilizados para avaliação da abordagem desenvolvida, visando o amadurecimento da abordagem e minimizando os riscos de sua transferência para ambiente industrial.

## 1.6. Metodologia

Para o desenvolvimento deste trabalho de pesquisa optou-se por utilizar uma metodologia baseada em experimentação, cujo objetivo é a analisar a viabilidade da abordagem proposta, o amadurecimento e adequação ao contexto industrial (Figura 1).

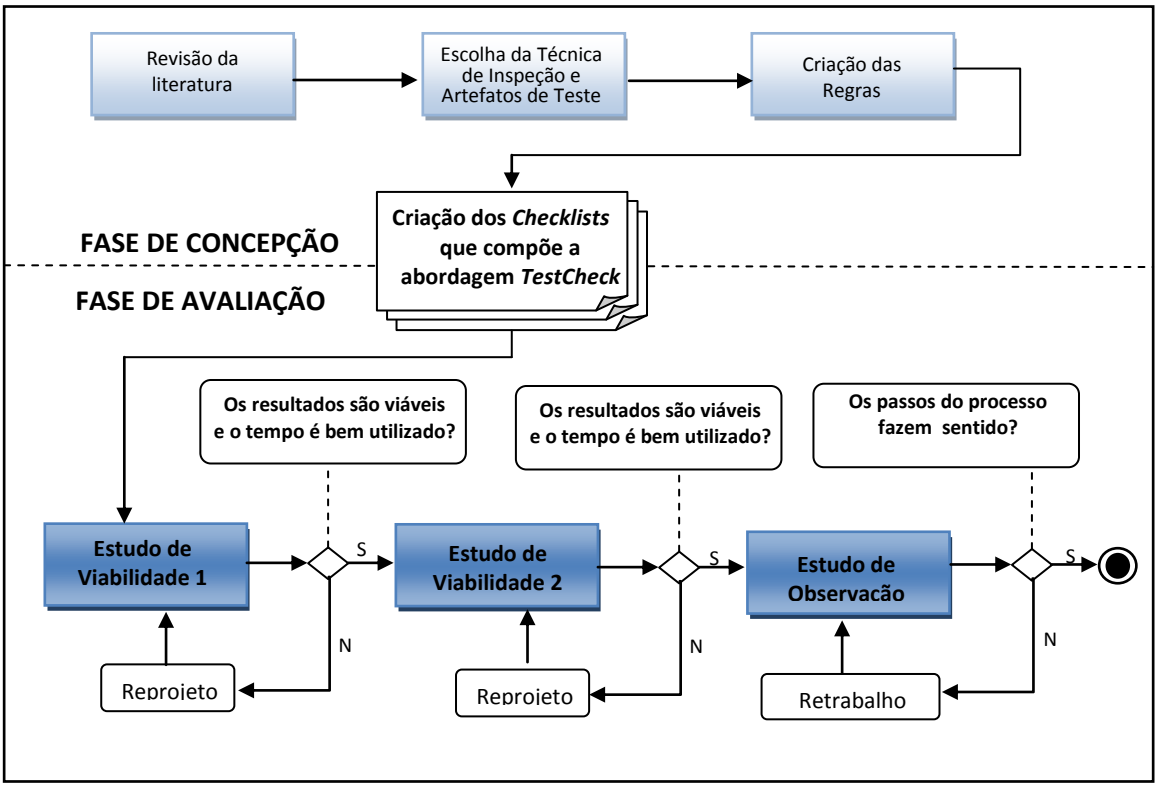



**Figura 1. Metodologia adotada para realização do trabalho de pesquisa.**

### 1.6.1. Fase de Concepção

- **C1)** Revisão da literatura sobre técnicas de inspeção em artefatos de teste de software seguindo os princípios de uma revisão sistemática da literatura (KITCHENHAM et al., 2009).

- **C2)** Escolha formal dos artefatos de teste que serão inspecionados e sob quais formatos.

- **C3)** Criação das regras para avaliação dos artefatos de teste selecionados no passo anterior.

- **C4)** Desenvolver o(s) *checklist*(s) que compõe a abordagem proposta para inspecionar artefato(s) de teste selecionado(s).

### 1.6.2. Fase de Avaliação (SHULL et al., 2001):

Para avaliação desta pesquisa, fez-se necessário a realização de estudos experimentais visando caracterizar a abordagem proposta em relação ao seu objetivo, identificar defeitos, visando permitir a transferência dessa tecnologia para o contexto industrial. Para isso, a metodologia definida por SHULL et al., (2001) foi utilizada. Ela é composta por quatro etapas que buscam avaliar a tecnologia proposta desde sua definição, analisando sua viabilidade, até sua transferência para indústria. Nessas quatro etapas, são usados quatro tipos diferentes de estudos experimentais (Figura 2).

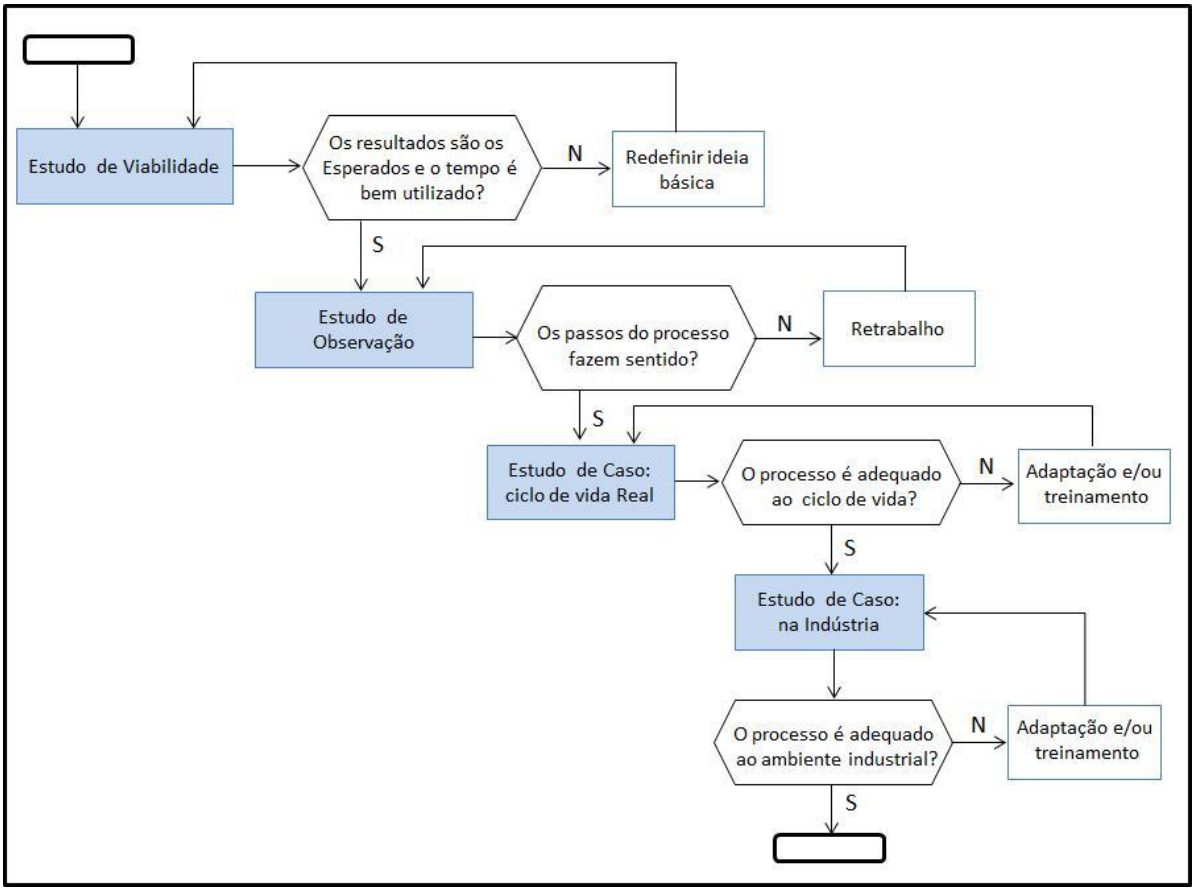



**Figura 2. Metodologia proposta em (SHULL et al., 2001)**

Nesta pesquisa, os três primeiros passos da metodologia de (SHULL et al., 2001) foram realizados, compondo a metodologia adotada na fase de avaliação desta pesquisa:

- **A1) Primeiro Estudo de Viabilidade:** é utilizado para avaliar uma abordagem ou técnica recém-criada. Consiste em avaliar se de fato a técnica construída é viável, chegando a um resultado satisfatório, ou seja, ela realmente faz o que se propôs a fazer;
- **A2) Segundo Estudo de Viabilidade:** visa o amadurecimento da técnica construída para aprimorar o entendimento dos pesquisadores em relação à aplicação da tecnologia e possibilitando seu refinamento;
- **A3) Estudo Observação na Indústria:** analisa se a técnica é realmente útil e caracteriza a aplicação da tecnologia no contexto de um ciclo de vida de desenvolvimento.

## 1.7. Organização do trabalho

Este trabalho está organizado em sete capítulos, incluído este primeiro capítulo de introdução, que apresentou a motivação, problema, objetivos e metodologia e o contexto da pesquisa. A organização do texto deste trabalho segue a estrutura abaixo:

- **Capítulo 2: "Fundamentação Teórica":** Descreve os principais conceitos relacionados à pesquisa, objetivando apresentar a importância de se construir testes com qualidade e por fim apresenta os trabalhos relacionados.
- **Capítulo 3: "Abordagem de Apoio à Inspeção de Artefatos de Teste – *TestCheck*":** Descreve o processo de criação da abordagem *TestCheck* baseada em *Checklist* objetivando auxiliar os inspetores na detecção de defeitos em Artefatos de Testes de Software.
- **Capítulo 4: "Estudo de Viabilidade para Avaliação de *TestCheck*":** Descreve o estudo de viabilidade realizado em ambiente acadêmico seguindo uma metodologia científica para avaliação de tecnologias de software. Consiste em uma seção de inspeção utilizando duas técnicas de inspeção que tem como objetivo detectar de defeitos em artefatos de teste de software e comparar os resultados de ambas as técnicas e avaliar suas eficiência e eficácia.
- **Capítulo 5: "Segundo Estudo de Viabilidade para Avaliação de *TestCheck*":** Descreve o segundo estudo realizado que equivale à primeira etapa da metodologia adotada em (SHULL et al., 2001). Consiste em um estudo de viabilidade com o objetivo de amadurecer a abordagem proposta focando na diminuição da taxa de falso-positivos quando utilizando *TestCheck* em relação ao primeiro estudo realizado.

- **Capítulo 6: “Estudo de Observação na Indústria”:** Descreve a terceira etapa da metodologia adotada, e a última realizada neste trabalho de pesquisa. Esta etapa consiste em um estudo de observação na indústria que objetiva analisar se a abordagem proposta se adequa ao ciclo de desenvolvimento de software em um contexto industrial.
- **Capítulo 7: Conclusões e Trabalhos Futuros:** Apresenta as conclusões desta dissertação, suas contribuições e trabalhos futuros que fornecem a direção para que seja dada continuidade a esta pesquisa.

# CAPÍTULO 2: FUNDAMENTAÇÃO TEÓRICA

*Neste capítulo serão apresentados os conceitos relacionados ao tema abordado nesta pesquisa e os trabalhos relacionados a este que está sendo proposto.*

## 2.1. Introdução

O processo de desenvolvimento de software envolve uma série de atividades nas quais, apesar das técnicas, métodos e ferramentas empregados, defeitos no produto ainda podem ocorrer. Atividades agregadas sob o nome de Garantia de Qualidade de Software têm sido introduzidas ao longo de todo o processo de desenvolvimento, entre elas atividades de Verificação, Validação e Teste (VV&T), que possuem como objetivo minimizar a ocorrência de defeitos em software e os riscos associados a estes, contribuindo desta forma para o aumento da qualidade do produto de software desenvolvido (MALDONADO, 1991). Dentre as técnicas de verificação e validação, a atividade de teste é uma das mais utilizadas, constituindo-se em um dos elementos para fornecer evidências da confiabilidade do software em complemento a outras atividades, como por exemplo, o uso de revisões e de técnicas formais e rigorosas de especificação e de verificação (MALDONADO, 1991).

De acordo com PRESSMAN (2006), garantir o funcionamento correto de um software em todos os possíveis cenários (de acordo com as especificações do cliente) e sem a presença de defeitos é algo impraticável em diversas situações. No entanto, existem atividades preventivas a fim de evitar esses tipos de problemas, dentre elas pode ser destacada a atividade de teste de software.

A atividade de teste de software consiste em uma análise dinâmica do produto e é uma atividade relevante para a identificação e eliminação de defeitos que persistem (PRESSMAN, 2006). No entanto, para que esta atividade desempenhe seu papel dentro do processo de garantia da qualidade é necessário que os testes que serão executados sobre o produto sejam de qualidade. A qualidade dos testes está diretamente relacionada à possibilidade de revelar falhas em um software. Quanto maior for esta possibilidade, mais qualidade possui uma atividade de teste. No entanto, se os artefatos produzidos ao longo do processo de testes apresentam defeitos, o objetivo de prover testes com qualidade dificilmente será atingido e resultado final dos testes não será satisfatório (ITOKEN et al., 2007). O processo de teste determina se o desenvolvimento do produto

está em conformidade com os requisitos e se o software satisfaz seu uso pretendido e as necessidades de seu cliente. O processo determina a amplitude apropriada e profundidade da documentação de teste – artefatos de testes (IEEE, 2008). Ou seja, é necessário garantir a qualidade dos artefatos gerados a partir do processo de teste.

Uma técnica que possibilita a avaliação dos artefatos de software e possui resultados significativos descritos na literatura é a técnica de inspeção (FAGAN, 1976). Esta técnica poderia ser empregada para a avaliação de artefatos de teste de software e com expectativas bastante positivas devido ao histórico de sucesso apresentado na literatura (MAFRA et al., 2005).

Assim, este capítulo aborda os principais conceitos referente a esta pesquisa, que possibilitará o entendimento para os próximos capítulos.

## 2.2. Testes de Software

Teste de Software consiste em uma investigação experimental conduzida para prover informações aos usuários e envolvidos no processo sobre a qualidade do software em teste, no contexto no qual este será operado. Isso inclui, mas não consiste apenas em um processo de executar um programa com a intenção de revelar falhas, apresentando uma análise dinâmica do produto, com atividades planejadas previamente e de modo sistemático (KANER, 2006). Toda estratégia de teste deve incorporar planejamento, projeto de casos de teste, execução de teste e a resultante coleta e avaliação de dados (PRESSMAN, 2006). É necessário e importante o compromisso do testador nesta atividade para manter sua qualidade, pois se realizada sem a devida preocupação e importância, será desperdiçado tempo e esforço. Os defeitos se infiltraram sem serem descobertos. Assim, é considerado um erro tratar as atividades de teste de software sem organização e planejamento.

A atividade de teste é uma atividade complexa. São diversos os fatores que podem colaborar para ocorrência de erros (DELAMARO et al., 2007). A inserção de defeitos pode ser facilitada devido à complexidade no desenvolvimento, e estes podem ser inseridos em diversos pontos do software, inclusive no decorrer do processo de desenvolvimento. Estima-se que quanto mais tarde um defeito for descoberto, maior será o custo decorrente de sua correção (BOEHM et al., 2001).

A importância do teste de software e sua relação com a qualidade devem ser enfatizadas, visto que existe um enorme crescimento do interesse por parte dos desenvolvedores com questões relacionadas à qualidade de software (DELAMARO et al., 2007). Com isso, a indústria tem despertado para a importância da atividade de teste de software, que por um lado pode contribuir para a melhoria da qualidade de um determinado

produto e em contrapartida pode representar um custo significativo dentro dos orçamentos empresariais. Para tanto, é indispensável no processo de desenvolvimento de software a utilização de métodos, técnicas e ferramentas que permitam a realização da atividade de teste de maneira que auxiliem no aumento da produtividade, da qualidade e da diminuição de custos desta atividade.

### 2.2.1. A Importância dos Testes em Metodologias de Desenvolvimento

Há uma infinidade de modelos diferentes para apoiar o desenvolvimento de software, cada um com seus pontos fortes e fracos, e eles tratam das atividades de teste de software de maneira diferente, conforme descrito a seguir:

- **Modelo Cascata**

O modelo em cascata (*waterfall*), publicado originalmente em (ROYCE, 1970), é extremamente estruturado, sequenciado e direcionado para manter relatórios de todas as decisões tomadas e atividades realizadas durante o desenvolvimento. As etapas no modelo em cascata podem ser vistas na Figura 3.

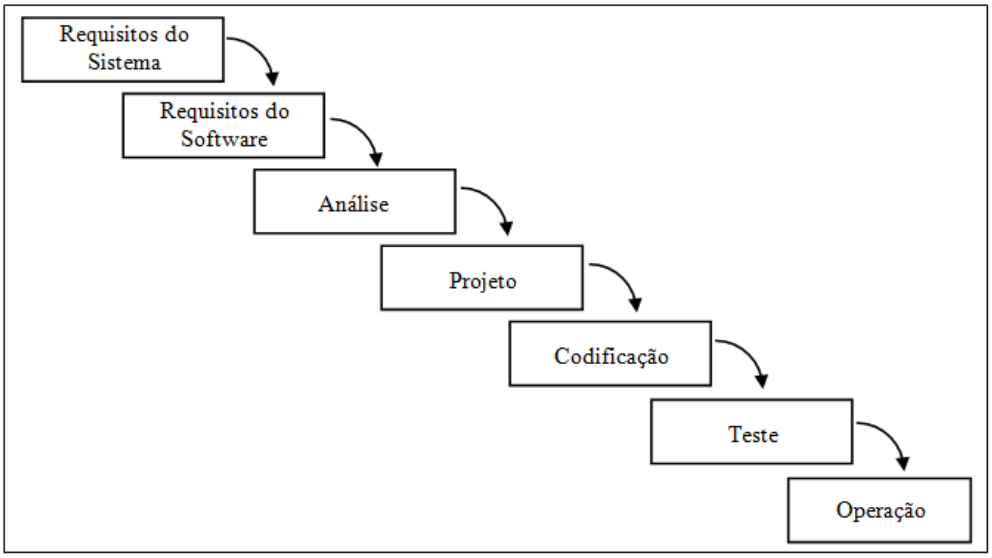



**Figura 3. Modelo Cascata (waterfall) original (ROYCE, 1970)**

Existem várias adaptações para este modelo. Este modelo propõe que todos os passos da etapa anterior sejam concluídos antes que a próxima etapa inicie. Segundo (RYBER, 2007), existe um argumento contra o modelo: “ele só funciona na teoria”, pois nunca se poderia terminar uma etapa completamente antes de ir para o próximo. O autor ainda afirma que um problema fundamental na atividade de testes do modelo em cascata é o fato de que esta entra no projeto em uma fase extremamente tarde e, por conseguinte, não se pode assegurar qualidade do software, além de introduzir um custo maior à correcao dos eventuais defeitos, devido ao momento tardio em que sao identificados.

- **Modelo V**

O Modelo V (Figura 4) é uma variação do modelo cascata que demonstra como as atividades de testes estão relacionadas com atividades de análise e projeto. Este modelo propõe que os testes de unidade, integração, sistema e aceitação também podem ser utilizados para verificar o projeto de software. Isto é, durante os testes de unidade e de integração os programadores e a equipe de testes devem garantir que todos os aspectos do projeto foram implementados corretamente no código, enquanto que nos testes de sistema e aceitação esta verificação pode ser realizada pelos usuários finais da aplicação. Em geral, ele é dividido em três camadas, onde a camada mais baixa está conectada para o desenvolvedor e seus testes de componentes, a camada central está conectada aos testadores e a equipe de projeto, que realizam os testes de sistema, e a camada superior, conectada ao cliente ou usuário final, que utiliza o teste de aceitação como forma de aprovar a entrega do que eles encomendaram (RYBER, 2007).

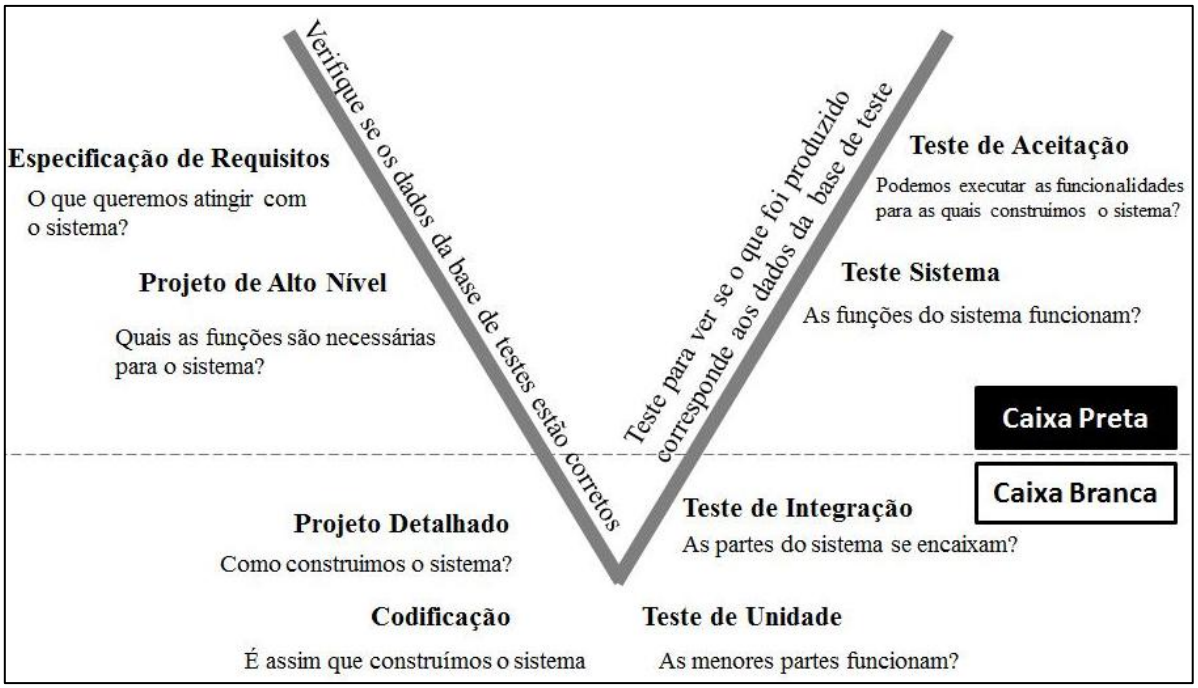



**Figura 4. Modelo adaptado de (RYBER, 2007)**

A conexão entre os lados esquerdo e direito do modelo V implica que, caso sejam encontrados problemas durante a verificação e a validação, o lado esquerdo do “V” pode ser executado novamente para corrigir e melhorar os requisitos, o projeto e a codificação antes da execução das etapas de testes que estão no lado direito. Em outras palavras, o modelo V torna mais explícitas algumas iterações e repetições do trabalho, ocultas no modelo cascata. Enquanto o enfoque do modelo cascata está nos documentos e nos artefatos, o enfoque do V está na atividade e na correção.

- **Modelo W**

Este modelo descreve que para cada atividade no desenvolvimento há uma atividade de teste correspondente (Figura 5) (RYBER, 2007). Ele procura suprir falhas existentes no modelo V, sendo uma delas não incluir nada sobre teste estático. A ênfase está na a execução de testes (lado direito), mas nada é dito sobre a revisão da base de teste. Ao contrário da forma como o modelo V funciona, com seu foco em diferentes níveis de teste, o modelo W foca nos produtos reais de desenvolvimento que são compilados. Toda atividade de desenvolvimento tem uma atividade de teste correspondente. O objetivo das atividades de teste é determinar se o produto entregue cumpre os seus requisitos.

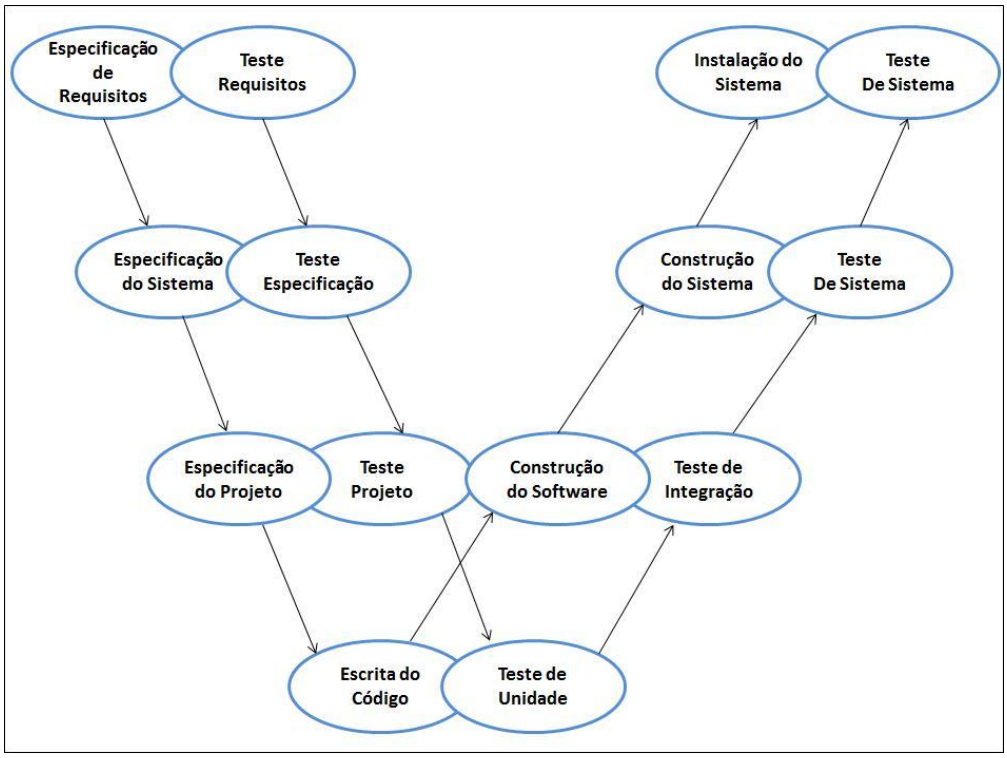



**Figura 5. Modelo W para teste (RYBER, 2007)**

O modelo W promove a ideia de que para toda atividade que gera um artefato que pode ser entregue, cada um destes artefatos deve ter uma atividade de teste associada. Por ser mais flexível que o modelo V (quantidade de níveis de teste igual à quantidade de níveis de desenvolvimento), é considerado o modelo que atende aos mais recentes modelos de desenvolvimento, como por exemplo o RUP – *Rational Unified Process* (RYBER, 2007).

- **RUP – *Rational Unified Process***

Inicialmente proposto por JACOBSEN et al. (1999), esta metodologia de desenvolvimento foi construída em torno de seis práticas que devem ser utilizadas ao

máximo pelas equipes. Essas práticas podem resolver os problemas que normalmente se enfrenta durante o desenvolvimento de software (RYBER, 2007):

1. Desenvolver iterativamente;
2. Gerenciar os requisitos;
3. Utilizar arquitetura baseada em componentes;
4. Modelo de software visual – modelos gráficos;
5. Verificar a qualidade do software continuamente – os primeiros testes;
6. Controle de mudanças para o software, um controle sobre as mudanças.

O principal objetivo do RUP é atender as necessidades dos usuários, garantindo uma produção de software de alta qualidade que cumpra um cronograma e um orçamento. Assim, o RUP mostra como o software será construído na fase de implementação, gerando o modelo do projeto. O RUP define perfeitamente quem é responsável por qual tarefa, como as coisas deverão ser feitas e quando devem ser realizadas, descrevendo todas as metas de desenvolvimento especificamente para que sejam alcançadas (RYBER, 2007).

A atividade de teste segue, e assume formas diferentes, dependendo da fase em progresso. A fase de Iniciação (ênfase no escopo do software), na maioria dos casos, não inclui qualquer execução de casos de teste. Testes durante a fase de Elaboração tem o objetivo de avaliar que a arquitetura definida para o software está coesa. Na fase de Construção (ênfase no desenvolvimento) são aplicados testes de unidade e de sistema, mas divididos em iterações. Na fase de Transição (ênfase na implantação), testes de aceitação ocorrem. Todas as iterações nas fases de Construção, Elaboração e Transição tem como objetivo criar algo executável e, portanto, testável (RYBER, 2007).

Esta abordagem pode ser comparada ao modelo W, que tem uma atividade de teste para cada atividade de desenvolvimento. Gerenciamento de requisitos, análise e projeto são os principais objetivos enfatizados no processo, e isso dá aos testadores uma excelente base na forma de modelos diferentes para a construção de casos de teste.

Um efeito dessa abordagem é que ela gera mais trabalho em testes, pois o número de testes de regressão aumenta quando a entrega de cada nova iteração tem de ser testada em conjunto com todas as partes anteriores.

A partir de tantos problemas enfrentados no desenvolvimento de software tradicional, pode ser percebida a importância dos testes para aumentar a qualidade no produto utilizando diferentes metodologias de desenvolvimento de software.

- **Metodologia Ágil**

Entre os processos ágeis mais conhecidos, podem ser citados: *eXtreme Programming* (XP) (BECK, 2005), *Scrum (*SCHWABER, 1995), *Lean Software*

*Development* (POPPENDICK, 2009), dentre outros. Os processos ágeis de desenvolvimento de software seguem, em geral, os princípios no manifesto ágil (AGILE MANIFESTO, 2001). No entanto, eles possuem características diferentes entre si, de forma que podem ser aplicados juntos de forma a se complementarem. Por exemplo, os métodos Scrum e XP podem ser aplicados juntos em um projeto de software, pois o primeiro enfatiza o gerenciamento de projetos, enquanto o segundo enfatiza a fase de codificação. A diferença entre os processos tradicionais e a metodologia ágil é que este acolhe mudanças, mas isso não significa que ele defende a desordem e o caos. Pelo contrário, a metodologia ágil é muitas vezes mais exigente do que nos processos tradicionais de desenvolvimento do software, uma vez que existe necessidade de uma qualidade forte e metodologia de teste a fim de ser capaz de acolher a mudança (RYBER, 2007).

Segundo (PRANGE, 2007), existem muitos problemas com a abordagem tradicional de teste:

1. Os testes devem ser feitos depois que todo o código foi escrito. Quando alguém não está acostumado com um software, é preciso algum tempo e esforço para poder tratar dos problemas existentes no código. Mesmo se o próprio programador que fez o código tiver que testá-lo, caso ele faça isso depois de ter implementado há muito tempo, ele poderá ter dificuldades.
2. Os testes devem ser feitos por outros desenvolvedores. Muitas vezes, quando os testes são feitos depois da implementação, é uma equipe especializada que os faz. Como eles podem não entender algum detalhe do código, é possível que esqueçam testes importantes.
3. Os testes podem não se basear no código. Os responsáveis por escrever os testes podem se basear em uma documentação ou em algum outro artefato que não seja o código. Se algum desses artefatos estiver desatualizado, os testes podem ser escritos de maneira totalmente equivocada.
4. Os testes podem não ser automatizados. Se os testes forem executados manualmente, certamente não serão realizados frequentemente e exatamente da mesma maneira a cada vez.
5. Alterações podem criar problemas que não são detectados pelos testes. É perfeitamente possível consertar um problema usando uma abordagem tradicional de modo a criar falhas em outros lugares. Se a cobertura de testes do código não for completa (ou próxima disso), a infraestrutura de testes existente pode não encontrar esses novos problemas.

As metodologias ágeis se tornaram populares na área de desenvolvimento de software e introduzem diversas práticas que contribuem para a qualidade do produto. Segundo PRANGE (2007), qualidade não pode ser alcançada apenas através da avaliação de um produto já pronto, mas antes e durante o desenvolvimento. Por exemplo, a metodologia XP faz uso de desenvolvimento orientado a teste, do inglês *Test Driven Development* (TDD) que escreve primeiramente os testes para depois escrever o código. PRANGE (2007) afirma que este tipo de desenvolvimento pode contribuir para resolver os problemas deixados pela abordagem tradicional (testes são feitos depois que todo o código foi escrito, testes são feitos por outros desenvolvedores, testes podem não se basear no código e alterações podem criar problemas que não são detectados pelos testes). O principio deste desenvolvimento está na construção primeiramente de teste, ou seja, primeiro é criado um caso de teste e em seguida faz-se à implementação do código necessário para que este seja executado com sucesso.

Usando o conceito de design simples (*simple design*), princípio do XP, o método é implementado de forma a fazer com que o teste de unidade passe. Uma vez que isso aconteça, o programador “limpa” o código e o mantêm sob as regras de design (refatoração). Esses passos são feitos sucessivamente, até que cada funcionalidade esteja pronta (PRANGE 2007).

### 2.2.2. Níveis de Testes

Os testes devem ocorrer em diferentes níveis, de acordo com as fases do processo de desenvolvimento. Os principais níveis de teste de software, segundo (ROCHA, 2001), são:

- **Teste de Unidade:** também conhecido como testes unitários. Tem por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente. O universo alvo desse tipo de teste são os métodos dos objetos ou mesmo pequenos trechos de código.
- **Teste de Integração:** visa revelar falhas associadas às interfaces entre os módulos quando esses são integrados para construir a estrutura do software que foi estabelecida na fase de projeto.
- **Teste de Sistema:** avalia o software em busca de falhas por meio da utilização do mesmo como se fosse um usuário final. Dessa maneira, os testes são executados nos mesmos ambientes, com as mesmas condições e com os mesmos dados de entrada que um usuário final utilizaria no seu dia-a-dia de manipulação do software. Verifica se o produto satisfaz seus requisitos.

- **Teste de Aceitação:** são realizados geralmente por um restrito grupo de usuários finais do sistema. Esses simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado.

A proposta apresentada neste trabalho visa apoiar a qualidade dos testes independentemente do nível de teste aplicado em um projeto de software.

### 2.2.3. Processo de Teste

Segundo (PFLEEGER, 2004), um processo no contexto de Engenharia de Software consiste em uma série de passos necessários ao planejamento, execução e controle das atividades, restrições e recursos que produzem um resultado esperado. Contextualizando este cenário para o processo de teste de software, o objetivo do processo de teste é revelar falhas no software, para que estas sejam solucionadas e seja entregue um produto com qualidade (CRESPO et al., 2004). A execução do processo de testes deve ocorrer ao longo do processo de desenvolvimento de um software, e possui diferentes tarefas de acordo com a atividade atual do processo de desenvolvimento, produzindo artefatos de teste (PERRY, 1995).

Segundo VILELA (2004), um processo é composto pelos seguintes elementos: sub-processos, atividades, artefatos (produzidos e consumidos), pré-condição, pós-condição, recursos e ferramentas de apoio. Figura 6**Erro! Auto-referência de indicador não válida.**, Figura 7 e Figura 8 apresentam a representação gráfica seguindo a notação definida em (VILELA, 2004) para um possível processo de testes proposto por (DIAS NETO, 2006), que utiliza os documentos propostos no padrão IEEE-STD 829 (1998) como artefatos produzidos ao longo do processo. Este processo de testes é dividido em dois sub-processos (Figura 6).

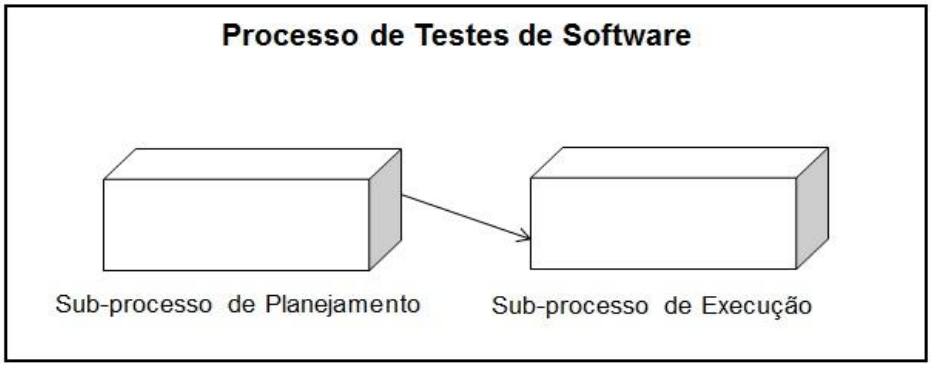



**Figura 6. Processo de Testes (DIAS-NETO, 2006)**

- **Sub-processo de planejamento de teste:** cujo objetivo é planejar e definir o que está sendo feito ao longo dos testes, antes que comece sua realização efetiva, ou

seja, definir objetivos, cronograma, pessoas, riscos, casos e procedimento de teste. As atividades desse processo produzem ao final os seguintes documentos, propostos pelo padrão IEEE-STD 829 (1998): Plano de Teste, Especificação de Projeto de Teste, Especificação de Casos de Teste e Especificação de Procedimento de Teste. As macro-atividades desse sub-processo são: Planejar Teste, Projetar Teste, Especificar Casos de Teste e Definir Procedimentos de Teste (DIAS-NETO, 2006). Este sub-processo está detalhado graficamente na Figura 7.

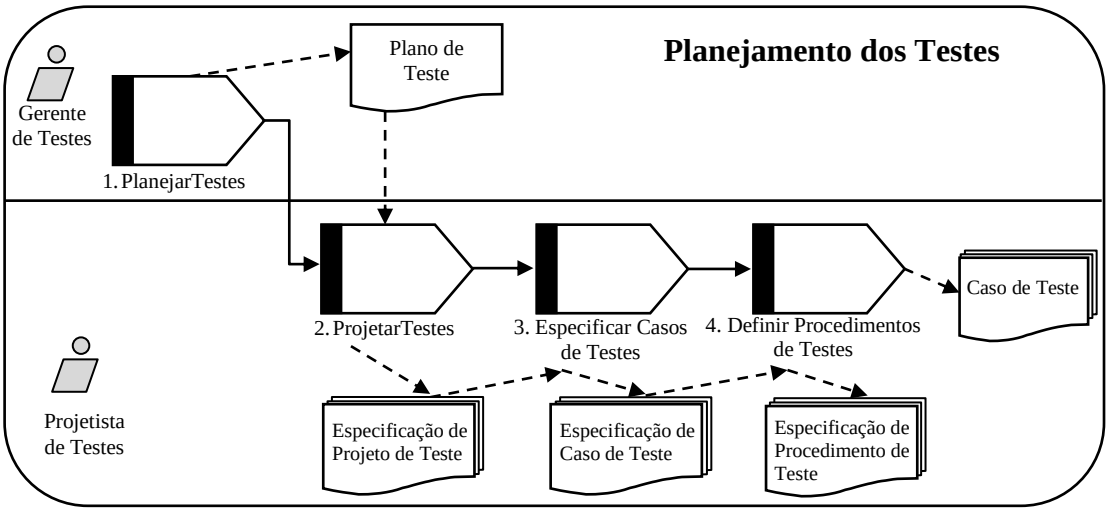



**Figura 7. Sub-Processo de Planejamento dos Testes (DIAS-NETO, 2006)**

- **Sub-processo de execução dos testes:** cujo objetivo é a execução de teste a partir do que foi estabelecido durante o planejamento, monitorando as atividades realizadas gerando os seguintes documentos, propostos pelo padrão (IEEE-STD 829, 1998): Histórico dos Testes, Relatório de Incidente de Testes e Relatório de Resumo de Teste, composto por das macro-atividades: Executar teste e Analisar Resultados. Este sub-processo está detalhado graficamente na Figura 8.

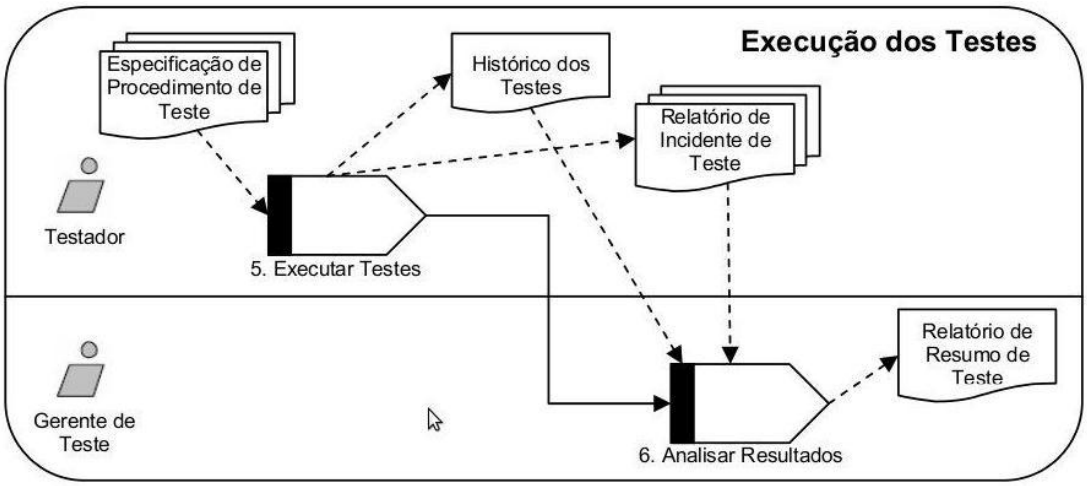



**Figura 8. Sub Processo de Execução dos Testes (DIAS-NETO 2006)**

Os documentos produzidos no processo de teste de software foram disponibilizados pelo padrão IEEE Standard 829 (1998). Os usuários deste padrão podem optar por adicionar, combinar ou eliminar documentos inteiros e/ou tópicos da documentação com base nas necessidades (e nível de integridade) dos seus software individuais. Este é um padrão cujo objetivo é descrever um conjunto de documentos produzidos no processo de teste de software. Tais documentos são: Plano de Teste, Especificação de Projeto de Teste, Especificação de Caso de Teste, Especificação de Procedimento de Teste, Histórico dos Testes, Relatório de Incidente de Teste, Relatório de Resumo de Teste e Relatório de Encaminhamento de Item de Teste. Em 2008, o padrão IEEE-Std 829 (IEEE-STD 829, 2008) passou por alterações, a quantidade de artefatos aumentou e as informações ficaram mais distribuídas e detalhadas. O Plano de Teste foi dividido, sendo acrescido um Plano de Nível de Teste (*Level Test Plan*). O objetivo do Plano de Teste passou a ser apenas fornecer um planejamento global de teste e de gestão de documentos de teste para vários níveis de teste (dentro de um projeto ou vários projetos), enquanto que o Plano de Nível de Teste especifica o escopo, a abordagem, recursos e cronograma das atividades de teste para cada nível de testes especificado, identifica os itens a serem testados, os recursos a serem testados, as tarefas de teste a serem realizadas, o pessoal responsável para cada tarefa, e o risco associado (s). Também foram incluidos o Relatório de Teste de Nível (*Level Test Report – LTR*) e Relatório de Teste Principal (*Master Test Report*). O objetivo do LTR é resumir os resultados das atividades de teste e com base nesses resultados fornece avaliações e recomendações. Costuma-se substituir a palavra "Nível" no título do documento para o nível de teste executado, por exemplo, Relatório de Teste de Aceitação. Existe um LTR para cada nível de teste definido pela organização ou projeto. A finalidade do Relatório de Teste Principal é resumir os resultados das atividades por nível de teste para fornecer avaliações baseadas nestes resultados. O artefato Relatório de Encaminhamento de Item de Teste foi removido desta nova versão.

Apesar de novos artefatos terem sido incluídos, não significa que é necessário utilizar todos eles. O padrão dispõe de vários artefatos com objetivo de proporcionar às organizações a escolha dos documentos que se adequem a realidade da empresa e que englobem todas as informações necessárias para a realização da atividade de teste.

## 2.3. Inspeção de Software

A técnica de inspeção foi originalmente desenvolvida na IBM em meados de 1970 por Michael Fagan (FAGAN, 1976) e, inicialmente, proposta para avaliação do código-fonte de programa. O objetivo específico é a identificação e remoção de defeitos de forma antecipada, ou seja, antes de o produto ser entregue sem, contudo, envolver-se na procura

de soluções para esses defeitos. Inspeções são formais, eficientes e um método econômico de encontrar defeitos no projeto e código (FAGAN, 1976).

É uma abordagem complementar dentro do processo de Verificação e Validação (V&V), para verificação e análise de software (PRESSMAN, 2006). É um processo de revisão, podendo ser realizada durante todas as fases do processo, como pode ser visto na Figura 9.

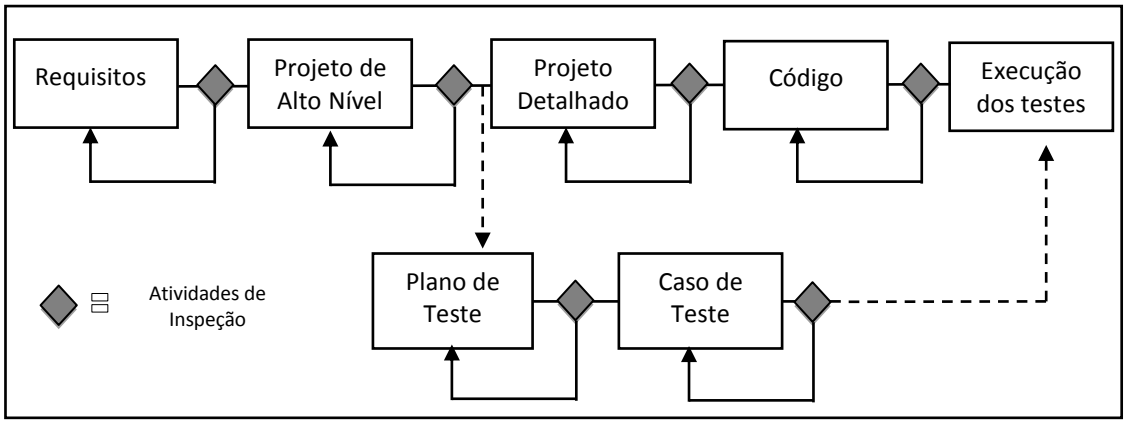



**Figura 9. Inspeções de Software nos diferentes artefatos (KALINOWSKI, 2004)**

### 2.3.1. Processo de Inspeção

Inspeção é considerada uma das atividades mais importantes do gerenciamento de qualidade e um dos melhores métodos para detectar defeitos nos diferentes subprodutos do processo de desenvolvimento de software (LANUBILE, 2007). Ela difere das outras técnicas de revisão por ser realizada de forma sistemática necessitando de reuniões com a equipe, cada participante envolvido tem responsabilidades específicas e desempenha funções bem definidas de acordo com os papeis a eles atribuídos (FAGAN, 1976). Esses papéis são descritos por FAGAN e apresentados a seguir:

- **Moderador:** é a pessoa chave para o sucesso da inspeção. Ele deve ser um programador competente, mas não precisa ser um perito no documento que está sendo inspecionado. Para preservar a objetividade e para aumentar a integridade da inspeção, é geralmente vantajoso utilizar um moderador de outro projeto que não esteja relacionado. Deve gerenciar e liderar a equipe de inspeção.
- **Autor:** é o profissional responsável pela criação do documento a ser inspecionado
- **Relator:** durante as reuniões o relator redige todos os possíveis defeitos encontrados reportados pelo inspetor do projeto.
- **Inspetor:** é o responsável por executar a inspeção.

O processo tradicional de Inspeção proposto por (FAGAN, 1976) inclui seis etapas, conforme a Figura 10:

- **Planejamento:** é analisada a documentação do projeto a ser inspecionado, é definido a equipe participante, a atividade que cada integrante vai exercer e o local das reuniões.
- **Visão Geral:** nesta etapa o autor apresenta e descreve com detalhes o documento a ser inspecionado a todos os membros da equipe, entrega a documentação a ser inspecionada e a divisão das funções a cada participante.
- **Preparação:** Os participantes, usando a documentação do projeto, procuram entender o projeto e iniciam uma leitura individual do documento a ser inspecionado, possivelmente os defeitos são flagrados durante esta operação, mas, em geral, o número de defeitos encontrados não é tão alto como na operação de inspeção.
- **Realização da Inspeção/ Detecção de Defeitos**: os participantes (inspetores) analisam o artefato de software, com o objetivo de detectar os defeitos existentes nesses produtos.
- **Retrabalho**: todos os defeitos detectados, observado no relatório de inspeção, são encaminhados ao autor, para que sejam resolvidos e removidos estes defeitos.
- **Revisão:** é da responsabilidade do moderador verificar que todas as questões, problemas e preocupações descobertos na inspecção foram resolvidos e que nenhum novo defeito foi introduzido.

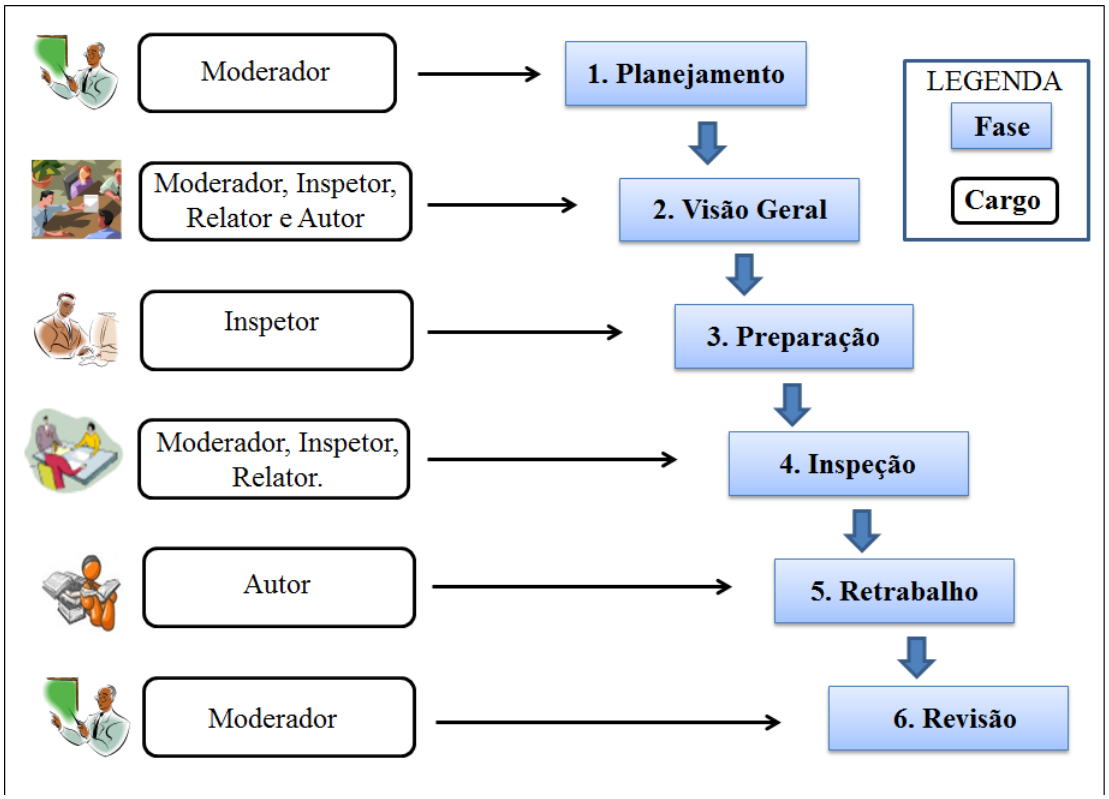



**Figura 10. Processo de Inspeção (BERTINI, 2006)**

Em 1989 é apresentada uma modificação no processo de inspeção proposto por Fagan, publicada em (HUMPHREY, 1989), onde alterou-se o foco da atividade de Preparação individual. Ao invés de tentar entender o artefato a ser inspecionado, passa-se a inspecioná-lo em busca de encontrar seus defeitos. Ou seja, cada inspetor entrega uma lista de discrepância para o moderador da inspeção antes que a reunião de inspeção se inicie. O objetivo da reunião passa a ser uma discussão em relação às discrepâncias encontradas e sua classificação em falso positivo (discrepância classificada como defeito por um inspetor de forma equivocada) ou defeito.

Em 2000, foi proposta uma reorganização do processo de inspeção tradicional, publicada em (SAUER et al., 2000). A reorganização visa introduzir reuniões assíncronas e equipes geograficamente distribuídas. Essas mudanças objetivam redução do custo e do tempo total para realizar inspeção. Esta reorganização do processo de inspeções de software mantém a estrutura rígida e os aspectos colaborativos do processo tradicional. Basicamente, consiste em substituir as atividades de Preparação e Reunião do processo tradicional por três novas atividades sequenciais: detecção de defeitos, coleção de defeitos e discriminação de defeitos (Figura 11).

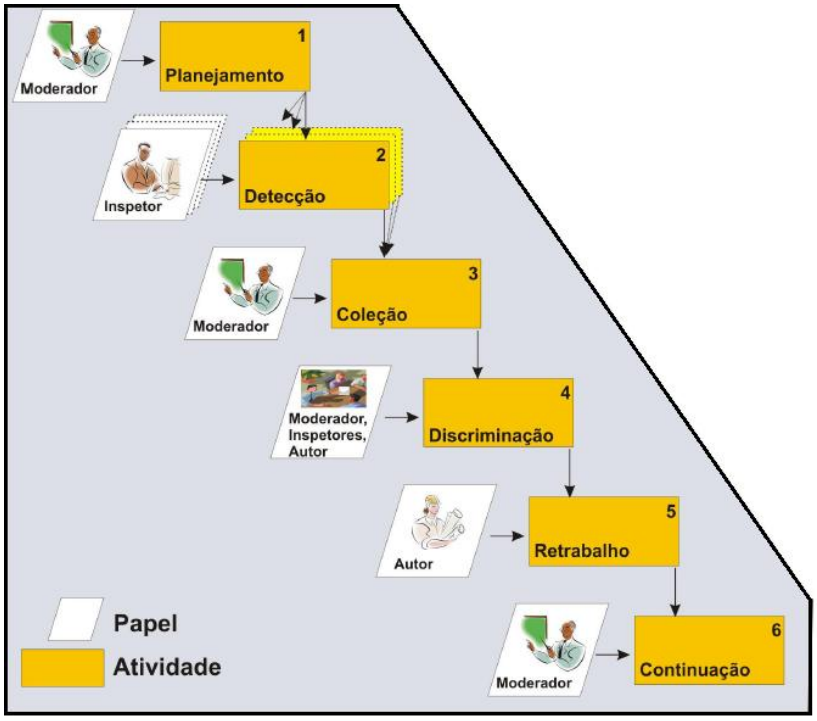



**Figura 11. Processo de Inspeção adaptado, (KALINOWSKI, 2004)**

- **Detecção de Defeitos:** cada inspetor selecionado pelo moderador durante o planejamento realizará uma atividade de detecção de defeitos. O objetivo desta atividade consiste em procurar defeitos no documento a ser inspecionado produzindo uma lista de discrepância

- **Coleção de Defeitos**: entregue as listas de discrepâncias ao moderador, este agrupa todas as listas. Esta atividade envolve eliminar discrepâncias repetidas (encontradas por mais de um inspetor), mantendo só um registro para cada discrepância.
- **Discriminação de Defeitos:** A equipe (moderador, autor e os inspetores) discute as discrepâncias listadas de forma assíncrona. Em seguida as discrepâncias serão classificadas em falso positivo ou defeito. Os falsos positivos serão eliminados e os defeitos serão agrupados em uma lista. De posse dessa lista o autor fará as correções necessárias.

### 2.3.2. Taxonomia de Defeitos

A inspeção busca detectar defeitos em artefatos de software. Existem diferentes tipos de defeitos e é importante que estes sejam classificados de forma a facilitar sua detecção. A taxonomia que foi adotada nesta pesquisa foi definida por SHULL (1998), composta pelos seguintes tipos: omissão, informação estranha, inconsistência, fato incorreto e ambiguidade.

Essa taxonomia é conhecida como sendo uma taxonomia não ortogonal, ou seja, é possível que um determinado defeito pertença a mais de um tipo (SHULL, 1998). A Tabela 1 apresenta a descrição de cada tipo de defeito utilizado na pesquisa e que foi apresentado a cada inspetor, devidamente interpretada para o contexto de teste de software.

**Tabela 1. Descrição dos Tipos de Defeito (SHULL, 1998)**

| Tipo Defeito | Descrição |
|---|---|
| Omissão | 1. Quando um elemento (ex: item de teste, tarefa, dado de entrada/saída, etc.) do planejamento de teste necessário para viabilizar sua execução não foi definido; |
| Ambiguidade | 2. Quando a forma como os elementos do planejamento de teste ou suas responsabilidades foram definidos dificulta seu entendimento, prejudicando a execução dos testes (ex: Homônimos). |
| Inconsistência | 3. Quando elementos descritos no planejamento dos testes possuem mesma descrição, mas nomes/identificadores distintos (Sinônimo);<br>4. Quando um elemento do planejamento dos testes presente em um artefato (caso ou procedimento de teste) não foi definido no plano de teste;<br>5. Quando a representação de um elemento não condiz com a semântica estabelecida pela abordagem de documentação dos testes. |
| Fato Incorreto | 6. Quando um elemento não foi descrito ou representado de forma correta (ex: adoção de um tipo incorreto para uma entrada de dados).<br>7. Quando não é possível mapear um elemento do planejamento de teste de um artefato para outro (ex: do caso de teste para o procedimento de teste). |
| Informação Estranha | 8. Quando não é possível determinar o papel de um elemento do planejamento de teste no atendimento aos requisitos especificados. |

### 2.3.3. Técnicas de Detecção de Defeitos

Com objetivo de auxiliar na detecção de defeitos, surgiram técnicas que podem ser aplicadas nesta fase, com o propósito de melhorar os resultados da inspeção. Tais técnicas são classificadas em três categorias principais: *Ad Hoc*, *Checklist* e Técnicas de Leitura.

- ***Ad Hoc:***

É uma técnica muito utilizada por organizações de software, porém não oferece nenhum apoio ferramental à detecção de defeitos. O inspetor depende exclusivamente do seu conhecimento e grau de experiência, portanto não existe a necessidade de treinamento. É classificada como não sistemática (os inspetores não recebem nenhuma orientação), geral (os inspetores devem conferir todos os aspectos do documento) e idêntica (todos os inspetores usam a mesma técnica) (BERTINI, 2006).

- ***Checklist:***

Representa uma boa melhoria, se comparada com a técnica *Ad Hoc*. Além disso, o *Checklist* pode ser melhorado continuamente baseado nos resultados de experimento utilizando a técnica. O *Checklist* consiste em uma lista de questões que os inspetores devem responder, sendo auxiliados a identificar em que parte do artefato avaliado devem procurar os defeitos (SHULL et al., 2000). O objetivo principal do Checklist é apresentar aos revisores dicas e recomendações para apoiar a detecção de defeitos. Segundo (BRYKCZYNSKI, 1999), heurísticas que são comumente sugeridas para a criação de UM *Checklist* eficaz incluem:

1. *Checklists* devem ser regularmente atualizados com base na análise de defeitos. Ao atualizá-lo regularmente, os inspetores podem ser mais propensos a ler e usá-los. Se os questionamentos do *Checklist* são atualizados em resposta a defeitos que ocorrem com frequência, então é mais provável que eles vão ajudar o revisor em encontrar defeitos adicionais.
2. *Checklists* não deve ser superior a uma página. Um revisor é menos inclinado a folhear as páginas de um *Checklist* ao examinar, por exemplo, uma listagem de código. Sendo de uma única página, este pode ser colocado sobre uma mesa e ler em estreita proximidade com o produto a ser analisado.
3. Os itens do *Checklist* devem ser redigidos na forma de uma pergunta. Esta heurística é bastante duvidosa, porque todos os itens do *Checklist* são baseados em questão que poderia ser reformuladas como frases imperativas.

4. Os itens do *Checklist* não devem ser muito gerais (por exemplo, “Os requisitos do sistema está completo, consistente e não ambíguo?”).

Segundo (BRYKCZYNSKI, 1999), a equipe do projeto deve investir o esforço necessário para analisar os tipos de defeitos que estão sendo introduzidos nos documentos e desenvolver *Checklists* personalizados para ajudar revisores aumentar a sua eficácia a detecção de defeitos.

- **Técnicas de Leitura:**

São procedimentos que visam guiar individualmente os inspetores no entendimento de um artefato de software e, por consequência, na identificação de defeitos (SHULL et al., 2000). As técnicas de leitura surgiram como forma de melhorar o desempenho da inspeção de software, no que se refere à atividade de detecção de defeitos. O termo leitura foi escolhido de forma a enfatizar as similaridades com o processo mental que as pessoas utilizam quando tentam entender o significado de algum texto (MAFRA et al., 2005). Segundo (BASILI et al., 1996), uma técnica de leitura pode ser caracterizada como uma série de passos para a análise individual de um produto de software textual para atingir a compreensão necessária para uma determinada tarefa. (SHULL et al., 2000) afirma que esta definição tem três componentes importantes:

1. **Uma série de passos:** Deve prover orientações ao usuário da técnica. O nível de orientação deve variar dependendo do nível mais adequado para o objetivo da tarefa e pode variar de um procedimento passo-a-passo a um conjunto de perguntas que visam concentração na leitura.
2. **A análise individual:** Técnicas de Leitura devem se preocupar com o processo de compreensão que ocorre com o indivíduo. Embora o método em que esta técnica seja incorporada possa requerer trabalho em equipe, por exemplo, depois da inspeção seja necessária uma reunião para discutir a compreensão de algum aspecto do documento, é importante que os defeitos sejam encontrados individualmente, ou seja, é uma tarefa individual.
3. **O entendimento é necessário para executar uma tarefa específica:** Técnicas de Leitura têm um objetivo particular: visam produzir certo nível de compreensão de alguns aspectos (ou todos) de um documento. Focando sempre no entendimento do desenvolvedor de artefatos.

Abordagens de avaliação baseadas em técnicas de leitura normalmente são mais eficientes na detecção de defeitos quando comparadas a outras técnicas de inspeção, como *Checklist*, por exemplo. No entanto, para que esse tipo de técnica venha ser utilizado,

o artefato deve ser apresentado de forma padronizada (BARCELOS, 2006). A Tabela 2 apresenta uma descrição comparativa das técnicas de detecção de defeitos citadas.

**Tabela 2. Comparação entre os tipos de técnicas de detecção de defeitos (SILVA, 2004)**

| Característica | *Ad Hoc* | *Checklist* | Técnica de Leitura |
|---|---|---|---|
| **Notação:** Qual linguagem ou notação os documentos devem ser escritos? | Qualquer | Qualquer | Específica (ex: IEEE-Std 829) |
| **Sistemático:** Os passos específicos do processo de revisão individual são definidos? | Não | Parcialmente | Sim |
| **Focado:** Cada revisor deve focar em diferentes aspectos do documento? | Não | Não | Sim |
| **Melhoria Controlada:** A partir da realimentação dos revisores, aspectos da técnica podem ser melhorados? | Não | Parcialmente | Sim |
| **Adaptável:** A técnica pode ser adaptada para projetos ou organizações específicas? | Não | Sim | Sim |
| **Treinamento:** A técnica permite ou necessita de treinamento? | Não | Parcialmente | Sim |

## 2.4. Trabalhos Relacionados

Foi realizada uma investigação na literatura em busca de pesquisas que envolvessem aplicação de inspeções ao longo do processo de desenvolvimento de software. Existem vários trabalhos que abordam inspeção em diversas fases desse processo, como por exemplo, inspeção aplicada a requisitos de software (POTTER et al., 1994; CHENG et al., 1996; KINER et al., 1997; SHULL, 2000; KALINOWSKI et al., 2007;), inspeção em documento arquitetural (BARCELOS, 2006; CLEMENTS *et al.*, 2002; DOBRICA et al., 2002; BAHSOON et al., 2003), inspeção de código fonte (PORTO, 2009; LAITENBERGER et al., 1997). No entanto, ao buscar trabalhos relevantes que abordem técnicas de inspeção em processo de teste, identificaram-se poucos trabalhos reportados na literatura técnica, o que motivou o desenvolvimento desta pesquisa.

Em (LANUBILE et al., 2007) é apresentada uma pesquisa sobre inspeção em casos de testes automatizados. De acordo com os autores, sempre que os casos de teste são defeituosos, o tempo dos desenvolvedores pode ser desperdiçado para resolver problemas que na verdade não são originários do código de produção. Devido à sua relevância no processo de desenvolvimento, postulou-se que a qualidade dos casos de teste pode ser assegurada através de inspeções de software como um complemento para a atividade de revisão informal. Nesta pesquisa, foi analisado o efeito da realização de inspeções de software em código de testes automatizados. Neste trabalho, na etapa de inspeção de

casos de teste, testes de unidade automatizados foram submetidos à mesma equipe de inspeção. O objetivo da inspeção foi verificar a qualidade do código de teste. Para este propósito, os inspetores utilizaram um *Checklist* para análise de código de teste. Os resultados indicam que inspeção de software pode melhorar a qualidade do código de teste, principalmente em relação à possibilidade de repetição dos casos testes, que é uma das qualidades mais importantes para a automação de teste. Os autores descobriram também que o benefício de inspeções de software pode ser observado quando os testes de unidade automatizados são criados por desenvolvedores individuais, bem como por dupla de desenvolvedores. Esta pesquisa foi realizada utilizando testes automatizados, que não é o foco da nossa pesquisa, mas os resultados positivos que foram obtidos indicam a viabilidade em aplicar técnicas de inspeções de software em artefatos do processo de testes como mecanismo para obtenção de melhorias na qualidade dos testes de software.

Em (HEDBERG et al., 2006) foi realizada uma pesquisa focando no papel das revisões técnicas em Mobile-D™ (método de desenvolvimento de software para dispositivos móveis). Este processo consiste em típicas técnicas ágeis, tais como programação em pares, que foram reivindicadas para melhorar a qualidade do produto. No entanto, revisões tradicionais eram vistos como dignos de experimentação. Seguindo as práticas do XP, testes são escritos por um desenvolvedor e não possuem interação com o cliente. O teste de aceitação contém detalhes técnicos que são vitais para o desenvolvimento. A metodologia de desenvolvimento orientada a testes utilizada no projeto aumentou a importância dos testes ainda mais, já que o código de teste foi escrito primeiro, e com base neles o código fonte foi escrito. Portanto, (HEDBERG et al., 2006) propõem a realização de revisões nos testes de aceitação com o objetivo de verificar se os testes realmente correspondem às estórias de usuários e encontrar testes adicionais, que podem ser vitais para a avaliação da estória. No projeto descrito no artigo, a equipe implementou este procedimento de revisão e os resultados foram bem sucedidos. Segundo os autores, o teste de aceitação revisado ficou com mais qualidade e continha menos defeitos.

Em (POON et al., 2010) foi realizada uma avaliação dos testes em um experimento para avaliar se a descrição e escolha de categorias (utilizando o critério de geração de dados de teste chamado partição por categoria, similar ao critério de classe de equivalência) para representação dos dados de entrada em testes funcionais em um projeto são bem realizadas. Após a definição das categorias, o artefato caso de teste foi gerado (somente categorias relevantes são úteis para geração de casos de teste) e submetido a dois estudos. O primeiro estudo realizado teve como principal objetivo investigar como os tipos e a quantidades de erros cometidos em uma abordagem de

identificação *Ad Hoc* variam entre testadores inexperientes e experientes. Para o segundo estudo, foi entregue um *Checklist* com objetivo de auxiliar os revisores. O objetivo principal foi avaliar a eficácia do *Checklist* proposto, em termos de sua capacidade para ajudar os testadores a reduzir a ocorrência de categorias ausentes e problemáticas. Após os estudos, os resultados foram analisados e para o primeiro estudo houve evidências de que a experiência no desenvolvimento e teste de software não ajuda a melhorar a qualidade do conjunto de categorias identificadas utilizando um abordagem de identificação *Ad Hoc*. No segundo estudo, observou-se que a utilização do *Checklist* ajudou os testadores de software a reduzir a ocorrência de categorias ausentes e categorias problemáticas de todos os tipos. Apesar disso, este *Checklist* proposto possui propósito diferente do apresentado nesta pesquisa, não sendo assim possível realizar comparações entre as abordagens.

Como dito anteriormente, poucos trabalhos que abordam técnicas de inspeção em processo de teste foram identificados na literatura técnica. Não foi encontrado nenhum trabalho que aplicasse e avaliasse a utilização da técnica de detecção de defeitos (baseada em *Checklist* ou Técnicas de Leitura) em artefatos de testes de software com o objetivo de melhorar a qualidade desses artefatos.

## 2.5. Considerações Finais

Ao longo deste capítulo foram apresentados os principais conceitos necessários ao entendimento e desenvolvimento da pesquisa, bem como trabalhos relacionados. Estes conceitos são importantes para mostrar a relevância deste trabalho e nos motivar com os resultados já apresentados na literatura. Em (ACKERMAN et al., 1989), o autor afirma que inspeções melhoram a qualidade do produto, principalmente se utilizar inspeções em várias fases do desenvolvimento, como a fase de definição de requisitos e implementação do software.

Usando inspeções em vários estágios do processo de desenvolvimento, é provável que sejam detectados defeitos em fases posteriores que não foram encontrados em inspeções anteriores. Em (KALINOWSKI, 2004) é mencionada a importância de inspeções de software para melhorar a qualidade de artefatos de software antes que o artefato seja passado para a próxima fase do processo de desenvolvimento de software. Segundo o autor, uma das formas de aumentar a qualidade é a aplicação de inspeções nos diferentes artefatos desenvolvidos ao longo do processo de desenvolvimento.

Além disso, a importância da aplicação de inspeções na garantia da qualidade é destacada por modelos que avaliam qualidade de processos de software, como CMMI

(2006) e MPS.BR (2011), que identificam revisões como uma das áreas chave para obtenção da qualidade em processos de software.

Assim, com o objetivo de melhorar a qualidade dos testes em projetos de software, foi desenvolvida uma abordagem baseada em *Checklist*, denominada *TestCheck*, que visa inspecionar os artefatos produzidos ao longo do processo de testes. O objetivo com a abordagem é atuar na qualidade dos artefatos produzidos ao longo do processo de teste como mecanismo para se obter a qualidade deste processo. No próximo capítulo será apresentada com detalhes a abordagem criada.

# CAPÍTULO 3: ABORDAGEM DE APOIO À INSPEÇÃO DE ARTEFATOS DE TESTE – *TESTCHECK*

*Este capítulo descreve o processo de criação da abordagem TestCheck, que consiste em uma abordagem para inspeção baseada em Checklist que tem como objetivo detectar de defeitos em artefatos de teste de software.*

## 3.1 Introdução

A utilização de técnicas de inspeção para garantir a qualidade do software desenvolvido está sendo bastante discutido na literatura técnica. Por isso existe todo um cuidado durante todo o processo de desenvolvimento, pois se é possível garantir um processo de qualidade, consequentemente será produzido um produto final com qualidade. Além disso, o custo de corrigir defeitos em artefatos de software aumenta exponencialmente com o decorrer do processo de desenvolvimento (BOEHM et al., 2001). Assim, novas iniciativas para avaliar a qualidade desses artefatos devem ser tomadas no sentido de identificar os defeitos para que estes possam ser corrigidos tão logo sejam introduzidos.

A abordagem *TestCheck* foi criada a partir de padrões existentes na literatura técnica e a experiência do grupo de pesquisa em Experimentação e Teste de Software (ExperTS[1]) da Universidade Federal do Amazonas (UFAM) em projetos de software na indústria. A sua concepção foi baseada nas características técnicas identificadas por (LAITENBERGER e DEBAUD, 1997) para compor uma abordagem de inspeção. O intuito foi encontrar uma solução adequada para o objetivo proposto neste trabalho, que é a criação de uma abordagem que permitisse avaliar a qualidade de artefatos produzidos ao longo de um processo de teste de software. Essas características levaram à execução das seguintes tarefas:

- Definir a técnica utilizada para detectar os defeitos dos artefatos;
- Determinar o artefato de software que deve ser avaliado pela abordagem;
- Definir os tipos de defeitos que a abordagem busca identificar;
- Especificar os itens que devem compor o(s) *Checklist*(s) proposto(s).

[1] http://www.icomp.ufam.edu.br/experts

Nas próximas seções, todas as escolhas tomadas para o desenvolvimento de *TestCheck* são justificadas, e então a abordagem é descrita com seus detalhes.

## 3.2 Escolha da Técnica de Detecção de Defeitos

Para a definição de uma abordagem para inspeção de artefatos de teste, algumas decisões precisaram ser tomadas, sendo a primeira delas a técnica de detecção de defeitos a ser utilizada (*Ad Hoc*, *Checklist* ou Técnica de Leitura).

Todas as técnicas foram analisadas criteriosamente para que aquela que melhor se adequasse à proposta do trabalho pudesse ser escolhida. Primeiramente, foi avaliada a técnica *Ad Hoc*. Aplicar técnicas *Ad Hoc* significa realizar a inspeção usando o conhecimento e experiência atual do inspetor, não sendo necessário qualquer instrumento de apoio a detecção de defeitos (BARCELOS e TRAVASSOS, 2006). Logo, não seria possível prover uma abordagem de inspeção baseada nesta técnica de detecção.

Outra técnica de detecção candidata é Técnica de Leitura. Abordagens de avaliação baseadas nessa técnica (SHULL et al., 2000; CONRADI et al., 2003; MAFRA et al., 2005) são mais eficientes na detecção de defeitos quando comparadas a outras técnicas de inspeção, como *Checklists*, por exemplo. No entanto, técnicas de leitura requerem que o artefato a ser inspecionado esteja descrito em uma forma específica e padronizada, para então guiar o processo de leitura dos documentos (BARCELOS e TRAVASSOS, 2006). Tal característica ainda não pode ser atingida na área de Teste de software por não existir uma padronização para se representar artefatos de teste. Apesar da existência do padrão IEEE-Std 829 (IEEE, 2008) como roteiro para documentação dos artefatos de teste, as empresas procuram adaptá-lo a sua realidade, como pode ser observado em uma pesquisa de opinião aplicada na indústria de software publicada em (DIAS-NETO, 2006). Além disso, este padrão descreve apenas roteiros que podem (devem) ainda ser customizados para serem aplicados em projetos de software.

Entre as técnicas de detecção de defeitos existentes, *Checklist* é uma das que podem ser utilizadas para identificar defeitos em artefatos de teste. Ela representa uma evolução às técnicas *Ad Hoc*, mas que não requer a padronização da representação dos documentos como ocorre nas técnicas de leitura. Ao se utilizar *Checklist* para revisar artefatos de software, é fornecido ao inspetor um conjunto de questionamentos que o auxiliam a identificar em que parte do artefato avaliado ele deve procurar por defeitos (LANUBILE et al., 2007). Utilizar esse tipo de técnica em inspeções de artefatos de teste consiste em definir questionamentos que, por exemplo, indicariam ao inspetor como identificar os elementos de teste

(planos, casos e procedimentos de teste) que devem ser analisados, visando avaliar o atendimento aos requisitos de teste do software.

Portanto, devido às características da área de Teste de Software, a técnica de detecção baseada em *Checklist* foi selecionada como a técnica de detecção de defeito a ser aplicada neste trabalho.

## 3.3 Escolha dos Artefatos de Teste a serem Inspecionados

Para elaboração da abordagem *TestCheck*, usou-se como fonte para seleção dos artefatos do processo de testes de software candidatos a serem inspecionados a lista de artefatos sugeridos pelo IEEE-Std-829 (IEEE, 2008), artefatos estes que se aplicam a diferentes metodologias de desenvolvimento (ex.: baseadas em processos ou metodologias ágeis), aliada à experiência do grupo ExperTS da UFAM em sua participação em projetos de software na industria (Figura 12). Os artefatos candidatos são: plano de teste, especificação de projeto de teste, especificação de caso de teste, especificação de procedimento de teste, log de teste, relatório de incidente de teste e relatório de resumo de teste.

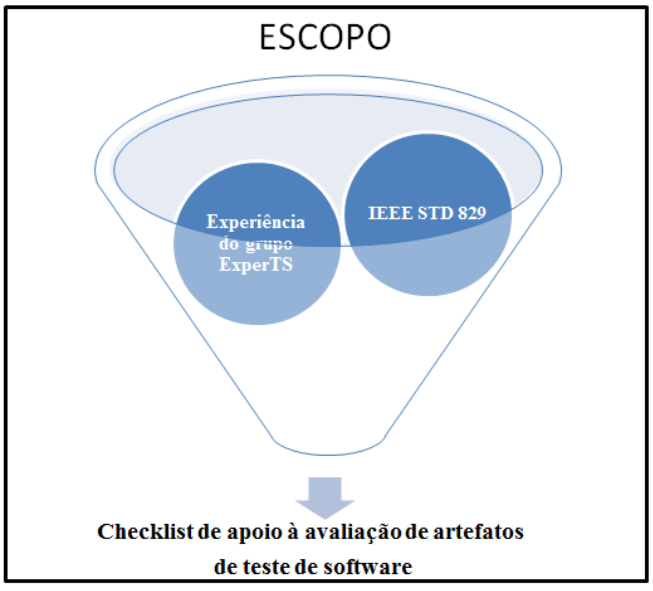



**Figura 12. Fontes de Apoio para Criação dos *Checklists***

A seleção dos artefatos de teste a serem inspecionados se baseou, inicialmente, na fase do processo de teste, onde optou-se neste momento por avaliar artefatos produzidos durante o planejamento dos testes, como forma de minimizar e prevenir a incidência de defeitos durante a execução dos testes, além de antecipar o momento de detecção de defeitos dentro do processo de testes, o que poderia inviabilizar esta atividade.

A partir da escolha da fase do processo de testes, a seleção dos artefatos a serem inspecionados foi feita com base nos resultados de um *survey* publicado em (DIAS-NETO, 2006) que foi aplicado em organizações de software localizadas em um polo de desenvolvimento de software brasileiro. Neste estudo, percebeu-se a maior ênfase das organizações de software na produção dos artefatos **plano de teste**, **caso de teste** e **procedimento de teste** ao longo de seus processos de teste. Outro fator que motivou esta escolha surgiu após uma parceria do grupo de pesquisa ExperTS com um instituto de pesquisa e desenvolvimento sediado na região Norte do Brasil, onde alunos do grupo de pesquisa atuam como membros da equipe de testes em projetos reais, e dentre suas atribuições estão a elaboração e revisão de planos, casos e procedimentos de teste. Mais detalhes sobre os artefatos selecionados para inspeção segundo o padrão IEEE-Std 829 (1998):

- **Plano de Teste:** descreve os objetivos dos testes, níveis de teste a serem aplicados, divisão de tarefas, atividade, cronograma e recursos necessários para os testes, identificação dos riscos relacionados aos testes, abordagem de teste a ser aplicada e critérios para aceitação e rejeição do software.
- **Especificação de Caso de Teste:** define com um nível adequado de detalhe as informações necessárias no que se refere às entradas e saídas do software que está sendo testado. Isso inclui todos os casos de testes identificados.
- **Especificação de Procedimento de Teste:** descreve todas as restrições especiais sobre os procedimentos, nível de teste que está relacionado a um caso de teste, pré-condições e os passos necessários para execução de um procedimento de teste.

A Figura 13 apresenta graficamente uma evolução do subprocesso de planejamento de testes descrito em (DIAS-NETO, 2006), de forma a inserir atividades de inspeção para avaliação dos artefatos entre as atividades atuais do processo de testes. As atividades de inspeção, que representam os itens evoluídos no processo de testes, estão representadas por um losango. Nesta figura, é possível identificar quais os artefatos a serem inspecionados e em que momento do processo de testes isso acontece.

Os artefatos selecionados podem ser aplicados a testes em diferentes níveis (ex: unidade, integração, sistema, aceitação). Assim, os *Checklists* que compõem a abordagem *TestCheck* foram elaborados de forma a ser aplicado a estes diferentes níveis, sendo formado por itens que proveem questionamentos a respeito da estrutura e conteúdo dos artefatos em estar direcionados a um nível específico.

As próximas subseções irão apresentar informações sobre os *Checklists* criados para apoiar a inspeção de cada um dos artefatos de teste selecionados nesta pesquisa.

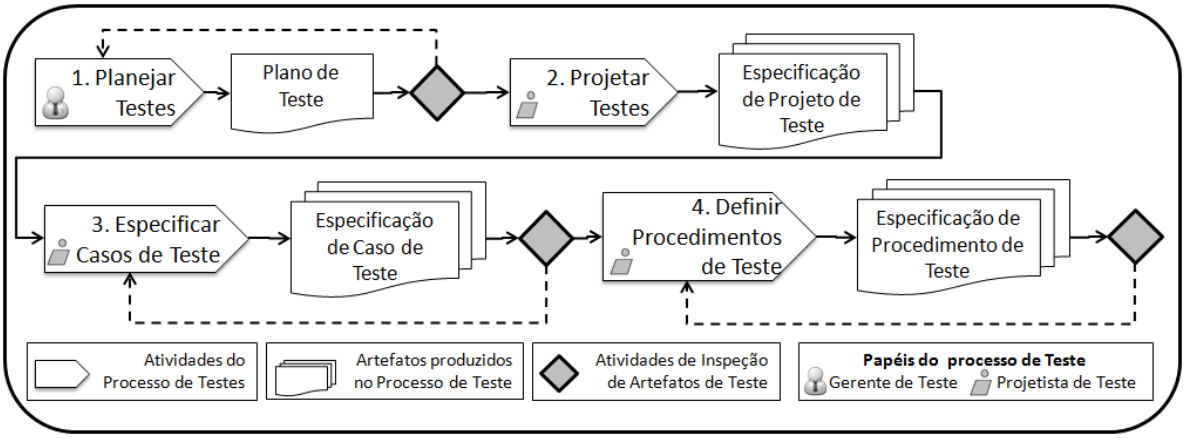



**Figura 13. Evolução do Subprocesso de Planejamento dos Testes (BRITO e DIAS-NETO, 2012)**

## 3.4 Itens de Avaliação dos *Checklists*

Defeitos em artefatos de software podem ser classificados através de diversas taxonomias. Essas taxonomias auxiliam os inspetores na identificação e categorização dos defeitos encontrados durante a atividade de detecção. A taxonomia utilizada como base neste trabalho é a definida por (SHULL, 1998) e foi apresentada na seção 2.3.2 (Tabela 1).

Para compor os *Checklists* que foram *TestCheck*, vários itens de avaliação foram definidos e agrupados de acordo com o artefato de teste a ser inspecionado. No entanto, esses itens foram detalhados em questões que visam atender a dois propósitos: (1) avaliar a consistência dos artefatos de teste entre si e (2) avaliar o atendimento dos testes aos requisitos (Figura 14).

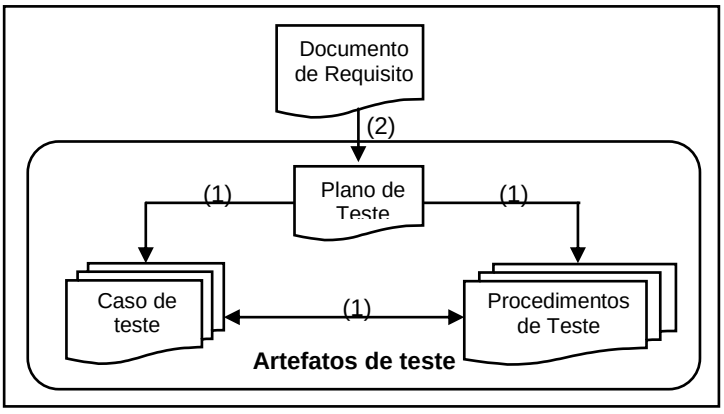



**Figura 14. Perspectivas de avaliação dos itens que compõem *TestCheck***

Cada questão dos *Checklists* propostos foi elaborada de forma que sua resposta positiva (SIM) representa a inexistência de defeito no artefato avaliado em relação ao conteúdo que ela se propõe a avaliar. Caso sua resposta seja negativa (NÃO), isso sugere que defeitos existem no artefato e precisam ser reportados ao moderador da inspeção. Além disso, os *Checklists* foram definidos com um conjunto abrangente de itens de

avaliação (questões) com o objetivo de permitir a adaptação às características dos artefatos de teste a serem avaliados em uma organização. Assim, foi definida uma opção "Não Aplicada" que pode ser respondida para uma questão caso aquele item não se adeque ao tipo de artefato adotado pela organização.

O processo de criação *Checklist* da abordagem ocorreu artefato a artefato, iniciando-se pelo desenvolvimento de um *Checklist* para o artefato Plano de Teste, em seguida passou-se para o artefato Caso de Teste e, por fim, foi definido o *Checklist* para o artefato Procedimento de Teste, conforme descrito nas próximas subseções.

Após as decisões terem sido tomadas, começou o processo de criação da abordagem baseada em *Checklist*. Primeiramente foram formulados os questionamentos para compor os *Checklist*, sendo que estes deveriam estar escritos de maneira a abranger qualquer tipo de software, podendo ser adaptável a um projeto de software. Por isso, procurou-se agregar todos os possíveis questionamentos que poderiam ser providos a respeito dos artefatos selecionados. Para que os *Checklists* atendessem ao objetivo proposto (detecção de defeitos), algumas características foram focadas:

- Os itens de avaliação devem avaliar características que devam conter em um artefato de teste de software.
- Os questionamentos devem estar agrupados por item de avaliação ao qual a pergunta está inserida. Por exemplo, em um item de avaliação "ITEM DE TESTE" só poderiam ser inseridas questões relacionadas a este item de avaliação, com o objetivo de facilitar a compreensão por parte de quem está utilizando a abordagem.
- Todos os questionamentos devem estar de forma clara, direta e concisa, evitando possíveis duplas interpretações.
- Todos os questionamentos devem estar colocados de maneira geral, não sendo específico a um tipo de software.

### 3.4.1. *Checklist* para Inspeção de Especificação de Plano de Teste

O plano de teste deve conter todas as informações necessárias e relevantes ao planejamento. Assim, defeitos inseridos neste documento podem comprometer o resultado final dos testes. O padrão IEEE-Std 829 disponibiliza 16 itens que devem fazer parte de um artefato plano de teste. Para compor o *Checklist* que avaliou o Plano de Teste, foi realizado uma análise pela pesquisadora juntamente com os membros do grupo ExperTS de todos os itens proposto pelo padrão IEEE, para identificar os possíveis itens de teste (que podem influenciar na qualidade do artefato e por consequência na qualidade dos testes executados) que deveriam ou não constar no *Checklist*. Neste caso, sua ausência foi justificada, conforme descrito na Tabela 3.

**Tabela 3. Justificativa para não inclusão de um item de teste**

| Itens que compõem o Plano de Teste | Justificativa para exclusão |
|---|---|
| Funcionalidades que não devem ser testadas | Sua ausência não afetará a execução dos testes, ao contrário das funcionalidades a serem testadas. |
| Introdução | Similarmente, não é uma informação que poderia causar algum problema na execução da atividade de teste. |
| Aprovações (Assinaturas) | O objetivo do *Checklist* é garantir a qualidade do documento, e não confirmar o comprometimento dos envolvidos ao plano. |
| Missão de Avaliação e Motivação dos Testes | Informação que não afeta a qualidade dos testes, além de poder ser incluída, por exemplo, na introdução. |

Assim, o *Checklist* para avaliação do plano de testes foi desenvolvido com 13 itens de avaliação (IEEE), a serem analisados através de 39 questões. Essas questões buscam auxiliar aos inspetores na busca por defeitos inseridos no documento. Os itens de avaliação que são avaliados pelo *Checklist* proposto estão listados na Tabela 4, juntamente com uma explicação a respeito do objetivo a ser atingido com sua avaliação.

**Tabela 4. Itens que compõem o *Checklist* para avaliação de plano de teste.**

| Itens de Avaliação | Objetivo com esta avaliação | Questões |
|---|---|---|
| Identificador do Plano de Teste | Evitar possíveis duplicações ou mesmo ausência, além da análise da versão do documento. | 3 |
| Item de teste | Avaliar a clareza e completude da descrição dos itens a serem testados em um projeto de Software. | 5 |
| Características de Qualidade | Avaliar a descrição de quais características de qualidade serão testadas ou não a partir deste plano de testes. | 3 |
| Tipos de testes | Avaliar se os tipos (níveis) de teste requeridos em um projeto estão descritos de forma adequada no plano. | 2 |
| Abordagem de teste | Avaliar a descrição das abordagens, técnicas e ferramentas a serem seguidas para a construção dos testes em um projeto. | 2 |
| Critério para aceitação/rejeição dos itens de teste | Avaliar a objetividade e clareza dos critérios de aceitação/rejeição definidos para os itens de teste. | 3 |
| Critérios de suspensão e retomada de testes | Avaliar a clareza e objetividade dos critérios definidos para suspender e reiniciar os testes. | 1 |
| Tarefas de testes | Avaliar a completude e corretude das tarefas listadas como necessárias para condução dos testes, evitando sobreposição de tarefas. | 5 |
| Responsabilidades/ Perfil da equipe | Avaliar a descrição e perfil dos grupos responsáveis por cada tarefa. | 3 |
| Necessidades do Ambiente de Teste | Avaliar a descrição das características físicas e Softwares/hardware necessários para execução dos testes, analisando sua adequação aos testes a serem realizados. | 2 |
| Necessidade de treinamento | Avaliar a descrição dos treinamentos necessários e compatibilidade aos testes a serem realizados. | 3 |
| Cronograma | Avalia a corretude dos prazos definidos para cada atividade/tarefa de acordo com sua complexidade. | 2 |
| Riscos e contingências | Avaliar a completude e adequabilidade dos possíveis riscos associados aos testes em um projeto, assim como a descrição de seus planos de mitigação e contingência. | 5 |

A partir dos itens de avaliação, foram criadas as questões visando uma maior abrangência na busca por defeitos. As questões não foram elaboradas com o objetivo de direcionar o inspetor na detecção de apenas uma categoria de defeitos. Procurou-se possibilitar ao inspetor que a partir de uma dada questão fosse possível identificar vários possíveis tipos de defeitos associados ao item de avaliação.

A Tabela 5 apresenta uma descrição da sua estrutura, com suas questões, seus objetivos, tipos de defeitos que podem ser revelados e um exemplo de um possível defeito que poderia ser reportado por um inspetor para o item de avaliação: ITEM DE TESTE.

**Tabela 5. Descrição das questões para o item de avaliação ITEM DE TESTE**

| **Descrição:** um item de teste consiste em um elemento (parte, módulo) a ser testado durante o projeto e que definirá a divisão do projeto dos testes (projetos para cada item de teste devem ser elaborados) | | |
|---|---|---|
| **ID** | **Questões para avaliar o item: ITEM DE TESTE** | **Tipo de defeitos esperados** |
| P04 | Os itens de teste definidos no plano de teste estão seguindo o mesmo padrão em relação às partes do Software a serem testadas? | Ambiguidade e Inconsistência |
| P05 | Os itens de teste listados no plano de testes realmente deveriam ser testados? Foi percebida a ausência de algum item importante? | Fato Incorreto, Informação Estranha e Omissão |
| P06 | Foram definidas as referências para os artefatos que descrevem os itens de testes? | Omissão e Inconsistência |
| P07 | Todos os itens de teste são únicos, não havendo repetição ou sobreposição? | Inconsistência |

| **Exemplos de Defeito** | **Questão que o detectou** | **Tipo do defeito** |
|---|---|---|
| A definição dos itens de teste não está padronizada, seguindo uma mesma unidade, pois um deles é um módulo do sistema e outro uma classe de dados. | P04 | Inconsistência |
| Um item de teste não identificado nos requisitos foi adicionado ao plano de teste. | P05 | Informação Estranha |

O mesmo procedimento foi adotado para os demais itens de avaliação e suas questões. A Figura 15 apresenta um extrato do *Checklist* para avaliação do Plano de Teste (sua versão mais recente). A primeira coluna contém a quantidade de questionamentos antecedidos pela letra “P” referenciando o artefato, os itens a serem avaliados e suas respectivas perguntas e as possibilidades de respostas. Este *Checklist* em sua versão completa está apresentado no Apêndice A deste trabalho.

ExperTS TESTCHECK – *Checklist* para Avaliação de Plano de Teste PPGI

| Nº | Item de avaliação: Identificador do Plano de Teste | Sim | Não | N/A |
|---|---|---|---|---|
| P01 | Foi definido um identificador para o plano de teste? | | | |
| P02 | O identificador utilizado determina unicamente o plano de testes? | | | |
| P03 | Está sendo identificada a versão atual do plano de teste? | | | |
| **Nº** | **Item de avaliação: Item de Teste** | | | |
| P04 | Os itens de teste definidos no plano de teste estão seguindo o mesmo padrão em relação às partes do software a serem testadas? | | | |
| P05 | Os itens de teste listados no plano de testes realmente deveriam ser testados? Foi percebida ausência de algum item importante? | | | |
| P06 | Foram definidas as referências para os artefatos que descrevem os itens de testes? | | | |
| P07 | Todos os itens de teste são únicos, não havendo repetição ou sobreposição? | | | |
| **Nº** | **Item de avaliação: Características de Qualidade** | | | |
| P08 | Todas as características de qualidade selecionadas para os testes estão presentes como requisitos funcionais/ não funcionais do software? | | | |
| P09 | Há necessidade de teste para todas as características de qualidade listadas? | | | |
| P10 | As características de qualidade que não serão testadas tiveram motivo adequadamente justificado? | | | |

**Figura 15. Extrato Parcial do *Checklist* para Avaliação de Plano de Teste.**

### 3.4.2. *Checklist* para Inspeção de Especificação de Caso de Teste

O caso de teste inclui dados de entrada, resultados esperados, ações e condições gerais para a sua execução. Para este artefato, o padrão da IEEE apresenta 7 itens que deveriam constar no documento. Novamente, foram analisados os itens de testes sugeridos e escolhido os itens para compor o *Checklist* proposto. Neste caso, nenhum item foi desconsiderado, o que aconteceu foi a unificação de dois itens (Especificação de Saída e Resultados esperados) em único item de avaliação.

Assim, o *Checklist* desenvolvido para avaliação de casos de teste possui 6 itens de avaliação detalhados em 20 questões. Estes itens estão listados na Tabela 6, juntamente com uma explicação a respeito do objetivo a ser atingido com sua avaliação e a fonte de origem de cada item. Os itens de avaliação provem uma análise a respeito do/a(s):

**Tabela 6. Itens que compõem o *Checklist* para avaliação de caso de teste.**

| Itens de Avaliação | Objetivo com esta avaliação | Questões |
|---|---|---|
| Identificador do Caso de Teste | Evitar possíveis duplicações ou mesmo ausência, além da análise da versão do documento. | 3 |
| Item de teste | Avaliar a consistência entre os itens de teste descritos no plano de teste e o item de teste associado ao caso de teste que está sendo descrito. | 3 |
| Especificações de Entrada | Avaliar a corretude e completude das entradas descritas para o caso de teste, seus valores, tipos e pré-condições. | 5 |
| Especificações de Saída/Resultados esperados | Avalia a corretude e completude dos resultados ou comportamentos esperados após os testes. | 4 |
| Necessidades do Ambiente de Teste | Avaliar a descrição de recursos físicos (Software/ hardware) necessários para executar um caso de teste. | 2 |
| Dependência entre casos de teste | Avaliar a corretude do relacionamento de dependência entre os casos de teste do projeto. | 3 |

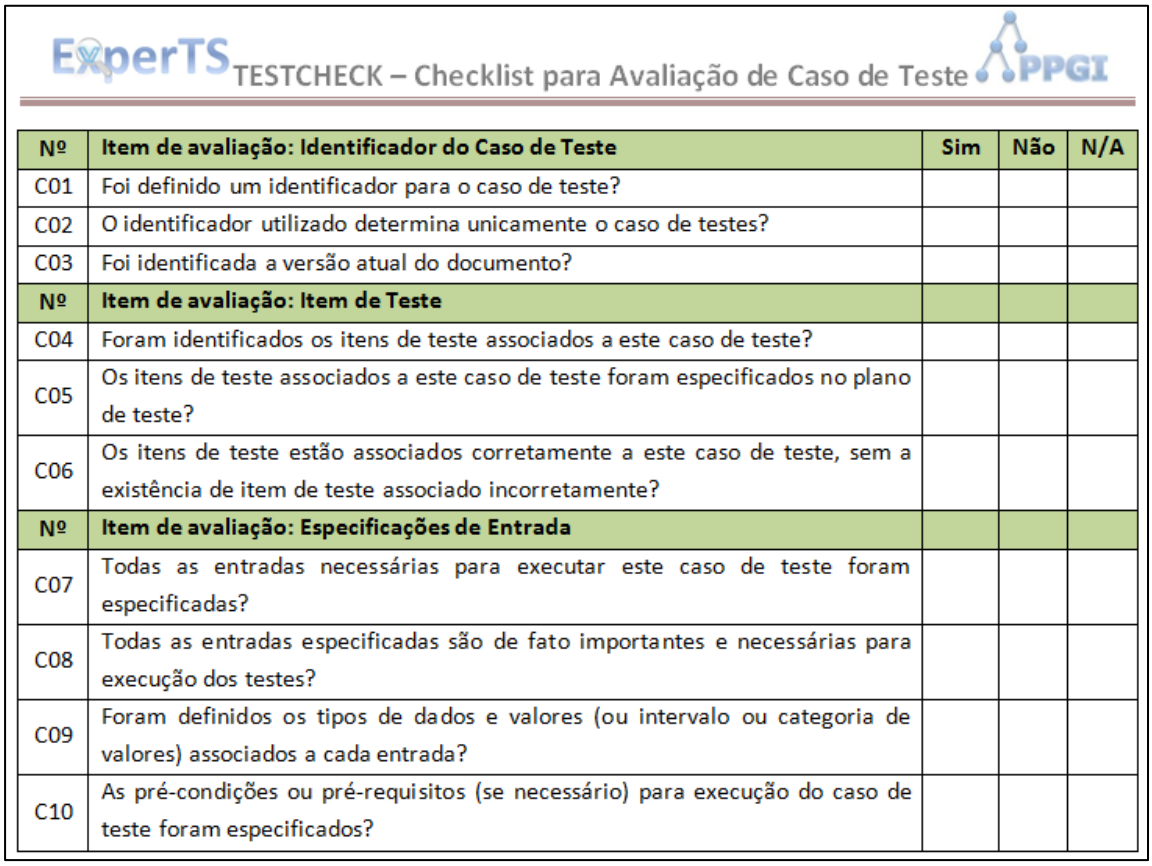
Após a escolha dos itens que compõe o *Checklist* Caso de Teste começou o processo de elaboração das questões. Novamente as questões foram criadas visando uma maior abrangência na busca por defeitos. A Tabela 7 apresenta uma descrição da sua estrutura, com suas questões, seus objetivos, tipos de defeitos que podem ser revelados e um exemplo de um possível defeito que poderia ser reportado por um inspetor para o item de avaliação ESPECIFICAÇÕES DE ENTRADA.

**Tabela 7. Descrição das questões para o item ESPECIFICAÇÕES DE ENTRADA**

| **Descrição:** caso de teste é formado por dados de entrada usados para exercitar o Software. | | |
|---|---|---|
| **ID** | **Questões para avaliar o item: ESPECIFICAÇÕES DE ENTRADA** | **Tipo de defeitos esperados** |
| C07 | Todas as entradas necessárias para executar o caso de teste foram especificadas? | Omissão |
| C08 | Todas as entradas especificadas são de fato importantes e necessárias para execução dos testes? | Informação Estranha, Inconsistência e Fato Incorreto |
| C09 | Foram definidos os tipos de dados e valores (ou intervalo ou categoria de valores) associados a cada entrada? | Omissão e Ambiguidade |
| C10 | As pré-condições ou pré-requisitos (se necessário) para execução do caso de teste foram especificados? | Omissão |
| **Exemplos de Defeito** | **Questão que o detectou** | **Tipo do defeito** |
| Para o teste da funcionalidade CADASTRAR VEÍCULO foi definido um campo TELEFONE, que parece não ser adequado à funcionalidade em questão. | C08 | Informação Estranha |
| Não foi definido o tipo do campo CPF, podendo ser numérico ou literal. | C09 | Omissão |

A Figura 16 apresenta um extrato do *Checklist* Caso de Teste (última versão). A primeira coluna contém a quantidade de questionamentos antecedidos pela letra “C”

referenciando o artefato, os itens a serem avaliados e suas respectivas perguntas e as possibilidades de respostas. Este *Checklist* em sua versão completa está apresentado no Apêndice A deste trabalho.



ExperTS TESTCHECK – Checklist para Avaliação de Caso de Teste PPGI

| Nº | Item de avaliação: Identificador do Caso de Teste | Sim | Não | N/A |
|---|---|---|---|---|
| C01 | Foi definido um identificador para o caso de teste? | | | |
| C02 | O identificador utilizado determina unicamente o caso de testes? | | | |
| C03 | Foi identificada a versão atual do documento? | | | |
| **Nº** | **Item de avaliação: Item de Teste** | | | |
| C04 | Foram identificados os itens de teste associados a este caso de teste? | | | |
| C05 | Os itens de teste associados a este caso de teste foram especificados no plano de teste? | | | |
| C06 | Os itens de teste estão associados corretamente a este caso de teste, sem a existência de item de teste associado incorretamente? | | | |
| **Nº** | **Item de avaliação: Especificações de Entrada** | | | |
| C07 | Todas as entradas necessárias para executar este caso de teste foram especificadas? | | | |
| C08 | Todas as entradas especificadas são de fato importantes e necessárias para execução dos testes? | | | |
| C09 | Foram definidos os tipos de dados e valores (ou intervalo ou categoria de valores) associados a cada entrada? | | | |
| C10 | As pré-condições ou pré-requisitos (se necessário) para execução do caso de teste foram especificados? | | | |

**Figura 16. Extrato Parcial do *Checklist* para Avaliação de Caso de Teste.**

### 3.4.3. *Checklist* para Inspeção de Especificação de Procedimento de Teste

O procedimento de teste especifica os passos para executar um ou um conjunto de casos de teste. Para o *Checklist* referente a este artefato, todos os itens inclusos no padrão da IEEE foram utilizados. Assim, o *Checklist* desenvolvido possui 4 itens de avaliação e 16 questões. Estes itens estão descritos na Tabela 8.

**Tabela 8. Itens que compõem o *Checklist* para avaliação de procedimento de teste.**

| Itens de Avaliação | Objetivo com esta avaliação | Questões |
|---|---|---|
| Identificador do Procedimento de Teste | Evitar possíveis duplicações ou mesmo ausência, além da análise da versão do documento. | 3 |
| Propósito do Procedimento de Teste | Avaliar a descrição dos objetivos/propósitos do procedimento de teste em relação ao seu escopo. | 2 |
| Requisitos Especiais | Avaliar a descrição dos requisitos necessários para a execução correta do procedimento de teste. | 4 |
| Passos do Procedimento | Avaliar a completude/corretude e sequenciamento dos passos descritos para um procedimento de teste. | 7 |

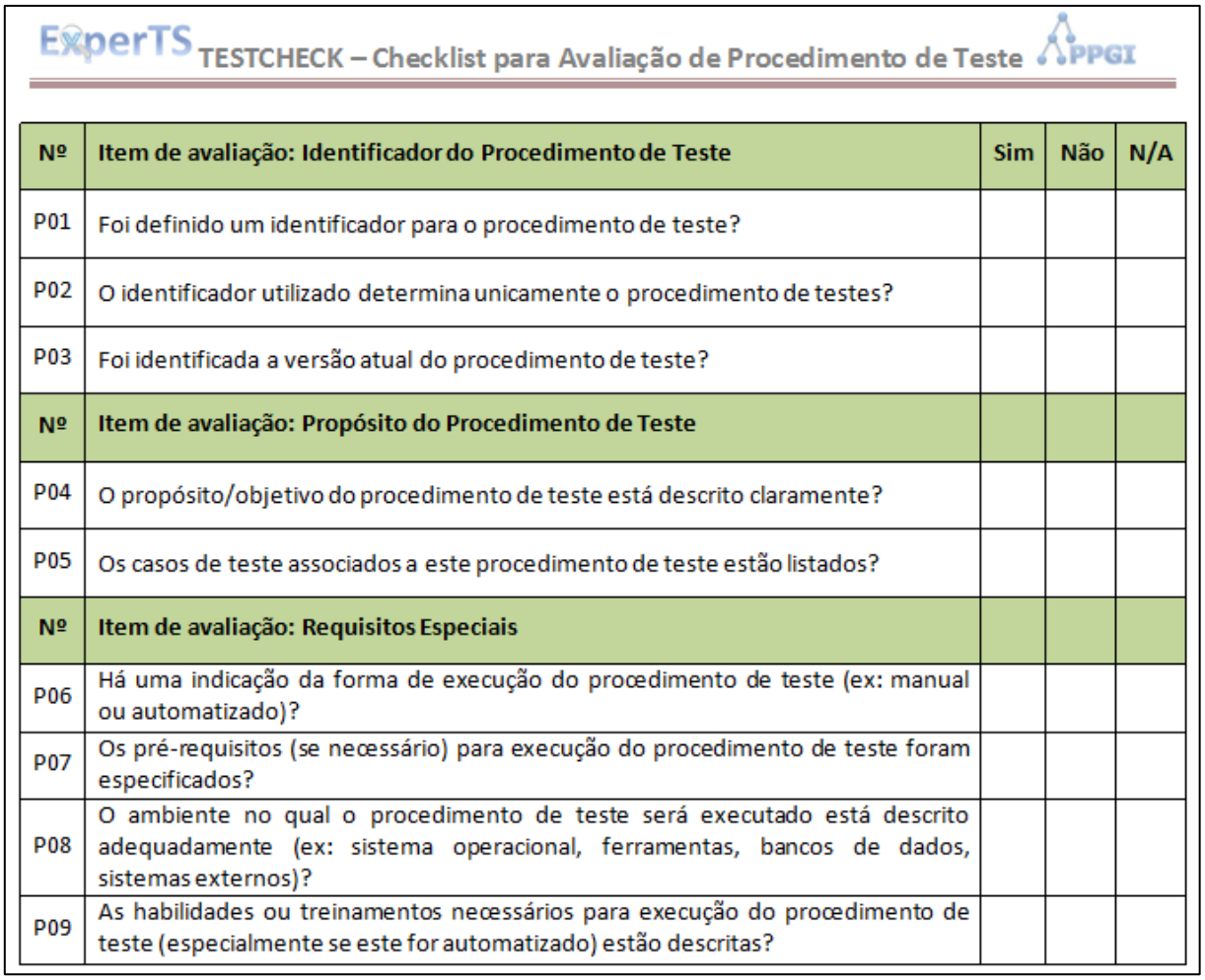
Identificados os itens que compõe o *Checklist* do Procedimento de Teste, foram elaboradas as perguntas seguindo os mesmos critérios utilizados na elaboração dos questionamentos para Plano e Caso de Teste. A Tabela 9 apresenta uma descrição da sua estrutura, com suas questões, seus objetivos, tipos de defeitos que podem ser revelados e um exemplo de um possível defeito que poderia ser reportado por um inspetor para o item de avaliação REQUISITOS ESPECIAIS.

**Tabela 9. Descrição das questões para o item REQUISITOS ESPECIAIS**

| **Descrição:** o procedimento de teste possui os passos para execução do teste. Entre outras informações contidas no artefato está requisitos especiais utilizado para execução do teste. | | |
|---|---|---|
| **ID** | **Questões para avaliar o item: REQUISITOS ESPECIAIS** | **Tipo de defeitos esperados** |
| P06 | Há uma indicação da forma de execução do procedimento de teste (ex: manual ou automatizado)? | Omissão e Inconsistência |
| P07 | Os pré-requisitos (se necessário) para execução do procedimento de teste foram especificados? | Omissão<br>Informação Estranha<br>Inconsistência e<br>Fato Incorreto |
| P08 | O ambiente no qual o procedimento de teste será executado está descrito adequadamente (ex: sistema operacional, ferramentas, bancos de dados, sistemas externos)? | Omissão |
| P09 | As habilidades ou treinamentos necessários para execução do procedimento de teste (especialmente se este for automatizado) estão descritas? | Omissão |

| **Exemplos de Defeito** | **Questão que o detectou** | **Tipo do defeito** |
|---|---|---|
| O ambiente no qual o teste será executado não foi definido | P08 | Omissão |
| Foi definido no documento de requisitos que o usuário só poderá utilizar opções do sistema se estiver autenticado no sistema. Ao analisar este procedimento, observou-se que esta restrição não foi definida como pré-requisito. | P07 | Omissão |

A Figura 17 apresenta um extrato do *checklist* para avaliação do Procedimento de Teste, referente à sua última versão. A primeira coluna contém a quantidade de questionamentos antecedidos pela letra “P” referenciando o artefato, os itens a serem avaliados e suas respectivas perguntas e as possibilidades de respostas. Este *Checklist* em sua versão completa também está apresentado no Apêndice A deste trabalho.



ExperTS TESTCHECK – Checklist para Avaliação de Procedimento de Teste PPGI

| Nº | Item de avaliação: Identificador do Procedimento de Teste | Sim | Não | N/A |
|---|---|---|---|---|
| P01 | Foi definido um identificador para o procedimento de teste? | | | |
| P02 | O identificador utilizado determina unicamente o procedimento de testes? | | | |
| P03 | Foi identificada a versão atual do procedimento de teste? | | | |
| **Nº** | **Item de avaliação: Propósito do Procedimento de Teste** | | | |
| P04 | O propósito/objetivo do procedimento de teste está descrito claramente? | | | |
| P05 | Os casos de teste associados a este procedimento de teste estão listados? | | | |
| **Nº** | **Item de avaliação: Requisitos Especiais** | | | |
| P06 | Há uma indicação da forma de execução do procedimento de teste (ex: manual ou automatizado)? | | | |
| P07 | Os pré-requisitos (se necessário) para execução do procedimento de teste foram especificados? | | | |
| P08 | O ambiente no qual o procedimento de teste será executado está descrito adequadamente (ex: sistema operacional, ferramentas, bancos de dados, sistemas externos)? | | | |
| P09 | As habilidades ou treinamentos necessários para execução do procedimento de teste (especialmente se este for automatizado) estão descritas? | | | |

**Figura 17. Extrato Parcial do *Checklist* para Avaliação de Procedimento de Teste.**

Durante o processo de concepção e avaliação da abordagem *TestCheck*, foram geradas 3 diferentes versões dos *Checklists* que a compõem, sendo que as modificações realizadas tiveram impacto somente nas questões definidas para avaliação dos itens descritos nesta subseção, ou seja os itens de avaliação permaneceram os mesmos. No Apêndice A pode ser obtida a versão mais atual dos *Checklists* que compõem *TestCheck*.

## 3.5 Considerações Finais

Este capítulo descreve com detalhes o processo de concepção da abordagem *TestCheck*. Essa abordagem foi desenvolvida com o objetivo de identificar defeitos em artefatos de teste de software. Os *Checklists* desenvolvidos que compõe a abordagem foram criados com base em padrões internacionais para documentação de artefatos de teste de software, aliados à experiência do grupo de pesquisa nesta área.

Apesar de usar fontes relevantes para concepção da abordagem, estas não podem garantir que a abordagem seja viável para o objetivo proposto. Para que se possa garantir

a viabilidade na detecção de defeitos, estudos experimentais são necessários visando avaliar a abordagem em relação a sua eficiência e eficácia em detectar defeitos em artefatos de teste de software, e confirmar que os itens de avaliação que compõe os *Checklists* são relevantes e que os questionamentos realizados são compreendidos.

Assim, concluída a fase de concepção de *TestCheck*, iniciou-se a sua fase de avaliação. Os próximos capítulos irão explicar como a versão atual da abordagem foi definida. Assim, no próximo capítulo será descrito um estudo de viabilidade da abordagem proposta e os resultados obtidos.

# CAPÍTULO 4: ESTUDO DE VIABILIDADE PARA AVALIAÇÃO DE *TESTCHECK*

*Este capítulo descreve o estudo de viabilidade realizado seguindo uma metodologia científica para avaliação de tecnologias de software, que consiste em uma seção de inspeção utilizando duas técnicas de inspeção que tem como objetivo detectar de defeitos em artefatos de teste de software e comparar os resultados de ambas as técnicas e avaliar suas eficiência e eficácia.*

## 4.1. Introdução

Devido à importância dos testes de software no processo de desenvolvimento e na qualidade do produto final, a avaliação dos documentos produzidos ao longo do processo de teste, verificando se eles atendem aos requisitos especificado pelo cliente e projetado pela equipe técnica do projeto, passa a ter extrema importância para os *stakeholders* (profissionais envolvidos diretamente e indiretamente no projeto). Neste contexto, foi proposta uma abordagem para inspeção baseada em *Checklist* (*TestCheck*) visando identificar defeitos existentes em artefatos de teste.

Após a construção de *TestCheck*, fez-se necessário a realização de estudos de avaliação visando caracterizar essa nova abordagem em relação à sua capacidade em identificar defeitos e visando permitir a transferência dessa tecnologia para o contexto industrial, reduzindo os riscos de implantação em um contexto industrial. Para isso, a metodologia definida por SHULL et al. (2001) foi utilizada. Ela é composta por quatro etapas, das quais três foram aplicadas nesta pesquisa.

Assim, o primeiro estudo realizado foi o de viabilidade e teve como objetivo caracterizar os itens de avaliação (*Checklist*) que formam a abordagem *TestCheck* em relação a sua eficiência e eficácia para identificação de possíveis defeitos em artefatos de teste. Seu planejamento, projeto e resultados estão descritos a seguir.

## 4.2. Definição do Estudo de Viabilidade

### 4.2.1. Propósito

O propósito do primeiro estudo foi responder a questão: “A aplicação da abordagem *TestCheck* para avaliação de artefatos de teste é viável analisando a sua eficiência e eficácia em relação ao número de defeitos detectados e ao tempo de aplicação?”.

### 4.2.2. Perspectiva

A perspectiva é do ponto de vista dos pesquisadores, que desejam analisar a viabilidade da aplicação da abordagem *TestCheck.*

### 4.2.3. Objetivos Específicos

- **Analisar** a aplicação da abordagem *TestCheck*.
- **Com o propósito de** caracterizá-la.
- **Em relação** à eficiência e eficácia em identificar defeitos em artefatos de teste.
- **Do ponto de vista** da Pesquisadora.
- **No contexto da** inspeção de artefatos de teste (Plano, Caso e Procedimento de teste) retirados repositórios da Internet.

### 4.2.4. Questões e Métricas

As questões abaixo se referem à aplicação da abordagem *TestCheck* em sessões de inspeção de documentos de teste. A saber:

- Q1: Qual é a eficácia da abordagem *TestCheck* no que diz respeito à detecção de defeitos?

  **Métricas:**

  - Quantidade de discrepâncias detectadas pela abordagem *TestCheck.*
  - Defeito = Quantidade de defeitos detectados pela abordagem *TestCheck*
  - Eficácia 2 = Quantidade de defeitos detectados / Quantidade de discrepâncias (defeitos + falso-positivos) identificadas.

- Q2: Qual é a eficiência da aplicação de *TestCheck* no que diz respeito à detecção de defeitos?

  **Métricas:**

  - Tempo = Somatório do tempo de cada participante quando utilizando a abordagem *TestCheck*.
  - Eficiência = Quantidade de defeitos detectados / Tempo de inspeção.

### 4.2.5. Questões em Aberto

- Qual a influência do nível de conhecimento do inspetor no domínio do problema ao utilizar a abordagem *TestCheck* para identificar defeitos?
- Qual a influência do nível de conhecimento do inspetor em teste de software ao utilizar a abordagem *TestCheck* para identificar defeitos?
- Qual o principal tipo de defeito encontrado por *TestCheck*?

## 4.3. Planejamento do Estudo

Esse estudo objetivou a inspeção de documentos de teste reais através dos *Checklists* que compõe a abordagem *TestCheck*. Com os resultados obtidos a partir da realização desse estudo é possível caracterizar a abordagem proposta em relação aos resultados obtidos com a aplicação de uma abordagem *Ad Hoc*. Entretanto, a comparação entre as duas abordagens não é o nosso objetivo principal, ou seja, não se pretende identificar qual a melhor abordagem de avaliação através dessa comparação. Até porque intuitivamente uma abordagem que auxilie o inspetor em como procurar e onde procurar um defeito terá melhores resultados quando comparada a uma avaliação *Ad Hoc* e os resultados obtidos com a aplicação dessas abordagens em somente dois casos não são suficientes para tirar esse tipo de conclusão. Tendo em vista as características da abordagem *Ad Hoc*, um possível questionamento é o porquê de utilizá-la para comparar com a abordagem proposta. Esta abordagem *Ad Hoc* foi aplicada por não ter sido identificado na literatura técnica outra abordagem similar à *TestCheck* que pudesse ser aplicada nos estudos.

Portanto, o objetivo dessa comparação é de caracterizar *TestCheck* em relação a um controle, ou *baseline*. Para esse estudo, o *baseline* escolhido foi uma abordagem *Ad Hoc*, pois é uma abordagem que tem sido utilizada em contexto industrial.

Com isso, neste estudo de viabilidade buscou-se adquirir conhecimento que permita avaliar a viabilidade de aplicação da abordagem proposta. Além disso, espera-se que os resultado obtidos, e o corpo de conhecimento construído decorrente de sua condução, forneça subsídios que permitam evoluir a abordagem para otimizar a sua aplicação tanto em relação ao esforço necessário para executá-la quanto à quantidade e tipo de defeitos que ela permite identificar por equipe de inspetores.

Para avaliar a abordagem *TestCheck* foi analisado se esta possibilita a identificação de defeitos em artefatos de teste (plano de teste, casos de teste e procedimento de teste) e em um tempo viável, evidenciando a eficácia/eficiência dos *Checklists* que a compõem, conforme descrito anteriormente.

### 4.3.1 Formulação de Hipóteses

A premissa por trás de uma abordagem de inspeção em geral é a orientação provida ao usuário no que diz respeito à condução da atividade em questão. Para avaliar essa premissa, foi realizada uma comparação entre os defeitos detectados pela abordagem proposta (*TestCheck*) e os resultados obtidos por uma abordagem *Ad Hoc*. Além disso, o objetivo deste primeiro estudo é prover um *baseline* a respeito da abordagem *TestCheck*, possibilitando assim futuras comparações ao longo de sua

evolução, e assim, compará-la inicialmente com a abordagem *Ad Hoc* contribui neste sentido. Portanto a hipótese nula que deseja-se refutar através desse estudo é:

**Hipótese nula (H0):** Não existe diferença entre os resultados encontrados pela abordagem *TestCheck* e a abordagem *Ad Hoc* em relação à detecção de defeitos em documentos de teste.

Para se obter resultados que venham a refutar ou a confirmar a hipótese H0, a eficácia e a eficiência das abordagens foram as características avaliadas. Portanto, para cada uma dessas duas características, as seguintes hipóteses foram definidas:

- **Eficácia:**
  - $D_T$ = quantidade de defeitos encontrados pelos inspetores que utilizaram a abordagem *TestCheck*.
  - $D_A$ = número total de defeitos encontrados pela abordagem *Ad Hoc*.

**Hipótese nula A (H0 A): A abordagem *TestCheck* permite identificar a mesma quantidade de defeitos identificados através da abordagem *Ad Hoc*.**

H0 A: $D_T = D_A$

- Hipótese Alternativa 1 A (H1 A): *TestCheck* permite encontrar mais defeitos que aqueles encontrados pela abordagem *Ad Hoc*.

  H1 A: $D_T > D_A$

- Hipótese Alternativa 2 A (H2 A): A abordagem da *Ad Hoc* permite encontrar mais defeitos que aqueles encontrados pela abordagem *TestCheck*.

  H2 A: $D_A > D_T$

- **Eficiência**
  - $T_T$ = Tempo médio das inspeções utilizando a abordagem *TestCheck*.
  - $D_T$ = Quantidade de defeitos encontrados pela abordagem *TestCheck.*
  - $T_A$ = Tempo médio das inspeções utilizando a abordagem *Ad Hoc*.
  - $D_A$ = Quantidade de defeitos encontrados pela abordagem *Ad Hoc*.

**Hipótese nula B (H0 B): A relação entre a quantidade de defeitos encontrados e o tempo médio de inspeção utilizando a abordagem *TestCheck* e a abordagem *Ad Hoc* são similares.**

H0 B: $D_T/T_T \approx D_A/T_A$

- Hipótese Alternativa 1 B (H1 B): A relação entre a quantidade de defeitos encontrados e o tempo médio de inspeção utilizando a abordagem *TestCheck* é maior que o da abordagem *Ad Hoc*.

  H0 B: $D_T/T_T > D_A/T_A$

- Hipótese Alternativa 2 B (H2 B): A relação entre a quantidade de defeitos encontrados e o tempo médio de inspeção utilizando a abordagem *TestCheck* é menor que o da abordagem *Ad Hoc*.

  H1 B: $D_T/T_T < D_A/T_A$

### 4.3.2 Seleção dos Participantes

Os participantes do primeiro estudo foram provenientes da lista de estudantes matriculados na disciplina Qualidade de Software nos cursos de graduação e pós-graduação do Instituto de Computação (IComp) da UFAM no período 2011/02. Entretanto, para participar do estudo, os estudantes tiveram que:

- Manifestar interesse em participar do estudo, assinando o Formulário de Consentimento (Apêndice B) e formulário de caracterização (Apêndice C), para termos conhecimento do grau de experiência de cada aluno;
- Participar do treinamento que foi conduzido como uma explicação dos *Checklists* que compõem a abordagem *TestCheck*, lista de discrepâncias e quaisquer dúvidas por parte dos alunos.
- Terem cursado alguma disciplina que abordou o tema teste de software ou terem participado de projetos reais atuando em tarefas relacionadas a teste de software, pois durante esse experimento será necessário algum conhecimento prévio sobre teste de software.

### 4.3.3 Seleção de Grupos

Cada aluno (inspetor) foi analisado e avaliado individualmente, através do questionário de caracterização que os mesmos preencheram com informações que permitiram identificar o grau de conhecimento, experiência e competência, tendo em vista que são alunos de graduação em estágio final e alunos de pós-graduação. Para este

estudo, os mesmos foram agrupados em equipes. Para compor as equipes, certas precauções foram tomadas visando reduzir eventuais fatores de confusão, por conta do nível de qualificação dos participantes. Para isso, os grupos foram mesclados de acordo com as informações contidas no formulário de caracterização.

### 4.3.4 Projetos a serem inspecionados

Foram inspecionados dois projetos contendo os artefatos de teste. Os artefatos do projeto *Industrial* são provenientes de um projeto real desenvolvido na indústria para um sistema de informação cujo objetivo de gerenciar usuários, seus perfis e funcionalidades em um software web para gestão de projetos. Os artefatos de teste do projeto *Acadêmico* são provenientes de um trabalho de uma disciplina de teste de software ministrada em 2010 onde alunos do IComp/UFAM (que não são os participantes do estudo) projetaram testes para um software de controle de uma biblioteca. Assim, os projetos entregues as equipes são reais e originais, não tendo sido inserido qualquer tipo de defeito pelos experimentalistas.

O critério adotado para a escolha destes projetos foi a disponibilidade do acesso aos artefatos e sua estrutura deveria estar em um formato que possibilitasse sua avaliação utilizando ambas as abordagens. Imagina-se que esses artefatos já tenham sido avaliados através de uma abordagem própria adotada por seus autores.

## 4.4. Execução do Estudo

O estudo de viabilidade foi conduzido de forma remota, ou seja, os participantes não realizaram a inspeção ao mesmo tempo e nem no mesmo lugar. Na primeira sessão, os candidatos responderam ao Formulário de Caracterização visando identificar a competência e a experiência relacionadas ao propósito do estudo, além de se buscar identificar o conhecimento do participante no domínio do problema ao qual os artefatos inspecionados pertencem. Após a caracterização, os participantes foram orientados a preencher um Formulário de Consentimento e receberam todas as informações a cerca do estudo, das atividades que deveriam ser realizadas e como deveriam proceder.

O estudo de viabilidade foi realizado em ambiente acadêmico, então para evitar qualquer resultado favorável ao estudo, em nenhum momento foi imposto nada aos possíveis participantes. O interesse em colaborar com a pesquisa deveria partir do próprio aluno, não havendo nenhum tipo de bonificação por participação. Assim 14 alunos manifestaram interesse em participar. Estes foram divididos em dois grupos com sete alunos cada, compondo as equipes A e B (Figura 18).

O estudo consistiu na simulação de uma sessão de inspeção de software. Os participantes em um primeiro momento aplicaram a técnica *Ad Hoc* em artefatos de teste de software. Cada participante recebeu individualmente um projeto (Industrial ou Acadêmico) distinto contendo um plano de teste, cinco casos de teste, cinco procedimentos de teste, para auxiliar e tirar qualquer dúvida foi entregue também um documento de requisitos, não sendo necessário inspecioná-lo. Neste momento, os membros da equipe A receberam os artefatos do projeto Industrial e os membros da equipe B receberam os artefatos do projeto Acadêmico (Figura 18). Após a inspeção, os participantes das equipes (A e B) entregaram as listas de discrepância (Apêndice D), ficando assim aptos a receber o treinamento para segunda etapa do projeto. Neste primeiro momento houve duas desistências, e assim deu-se prosseguimento ao estudo, com apenas 12 participantes (7 da equipe A e 5 da equipe B).

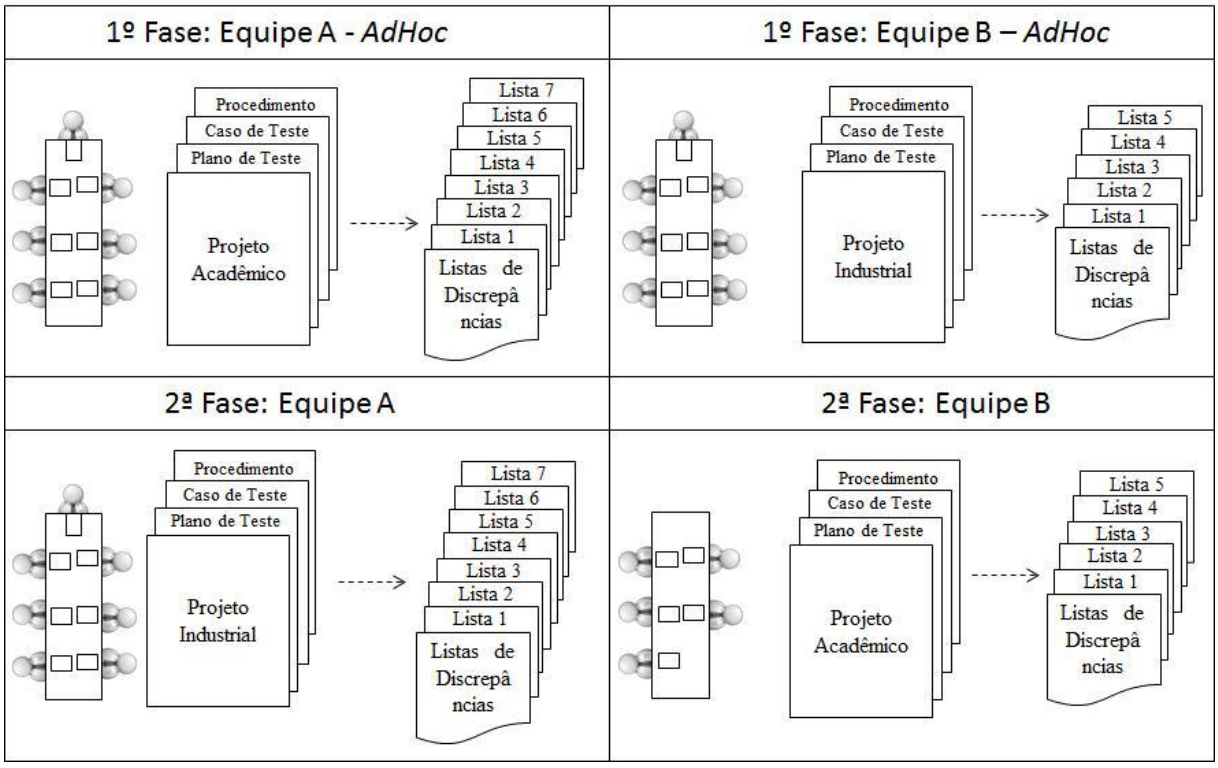



**Figura 18. Cenários de Inspeção dos Artefatos de Teste**

Após a entrega das listas de discrepância (pré-requisito para próxima etapa), os alunos receberam um treinamento para utilização da segunda abordagem, *TestCheck*. Eles foram treinados a respeito da utilização do *Checklist* desenvolvido (explicação dos itens que compõem os *Checklists* e lista de discrepâncias) e quaisquer dúvidas por parte dos alunos para que fosse possível sua aplicação sem qualquer intervenção do pesquisador. Após, o treinamento os projetos foram distribuídos entre as equipes.

Nesse momento foram invertidos os projetos, de forma que os participantes do Grupo A inspecionaram os artefatos de teste do Projeto Acadêmico, enquanto que os

participantes do Grupo B inspecionaram os artefatos de teste do Projeto Industrial, ambos utilizando *TestCheck* como abordagem de apoio. Da mesma forma, a inspeção foi conduzida de forma off-line.

Não foi imposto um limite de tempo para que ambas as avaliações fossem realizadas. No primeiro contado com os alunos estes receberam algumas informações, dentre elas foi pedido um máximo de atenção em relação ao tempo gasto durante todo o período de inspeção dos artefatos, ficando acordado que ao iniciar e ao terminar a inspeção o horário deveria ser rigorosamente anotado, para ser gerado o tempo em minutos utilizados durante todo o processo de inspeção. No entanto, ficou a cargo dos inspetores a forma como conduziriam a inspeção. Informação presente no relatório de discrepância.

Após a conclusão do experimento, foram coletados todos os Relatórios de Discrepância gerados por cada participante, e em seguida o experimentalista, com o apoio de especialistas em teste de software, classificaram as discrepâncias identificadas em defeitos ou em falso-positivos formando um único relatório de discrepância. Neste momento foram eliminados todos os defeitos ou falsos positivos duplicados, sendo considerado apenas um.

## 4.5. Ameaças à Validade

A partir da identificação de possíveis ameaças à validade dos resultados, não foram extraídas conclusões além das limitações impostas por estas ameaças, o que não contextualiza os resultados do estudo.

### 4.5.1. Validade de Conclusão

Relacionada a questões que ameaçam a habilidade de traçar conclusões corretas sobre o relacionamento entre tratamentos e resultados de um estudo experimental.

- **Confiança das Medidas:** dois dos objetivos que procuramos alcançar com esse estudo é identificar, em comparação aos resultados obtidos pela abordagem *Ad Hoc*, o número de defeitos reais e que tipos de defeito a abordagem *TestCheck* identificou. Sendo assim, uma potencial ameaça à validade de conclusão do estudo se refere à classificação de cada discrepância em defeitos por parte dos experimentalistas. Para isso, adotou-se o processo de revisão em pares, onde tal classificação foi sempre realizada por duas pessoas, tentando reduzir esta ameaça.
- **Heterogeneidade Aleatória dos Participantes:** os participantes foram alunos de graduação e pós-graduação em Engenharia de Software. Portanto, não se esperam

participantes com níveis de competências tão discrepantes que possam a vir comprometer a validade do estudo.

- **Confiabilidade na Implementação do Tratamento:** este risco está relacionado à aplicação do tratamento não ser similar entre os diferentes participantes do estudo. A abordagem *TestCheck* compreende um conjunto de documentos para sua aplicação, e um treinamento será realizado visando também padronizar a aplicação da abordagem.

### 4.5.2. Validade Interna

Relacionada ao risco de outros fatores não identificados a priori, terem influenciado um eventual relacionamento de causalidade entre tratamento e resultado, sem o conhecimento prévio do experimentador.

- **Instrumentação:** os formulários a serem utilizados no estudo experimental podem influenciar de forma significativa a condução do estudo, afetando negativamente seus resultados. Entretanto, por tratar-se de um estudo de viabilidade, essa questão não representa uma ameaça significativa à validade do estudo, haja vista que a avaliação da viabilidade de *TestCheck* também aplica-se à sua documentação associada.
- **Seleção:** a amostra dos participantes não será aleatória, visto que será proveniente da disponibilidade de alunos em uma disciplina ministrada pelos experimentalistas.
- **Plágio (*plagiarism*):** um típico risco na condução de estudos experimentais em ambiente acadêmico é a potencial troca de informações entre os participantes sobre as tarefas do estudo. Entretanto, acredita-se que essa ameaça não é pertinente, visto que os participantes se comprometeram a não se comunicarem durante a realização do estudo e o resultado da aplicação da abordagem não resultará em uma avaliação para a disciplina.

### 4.5.3. Validade de Construção

Problemas relacionados à ameaça de generalizar os resultados do estudo à teoria que o sustenta. As principais ameaças são:

- **Viés mono-operação:** o estudo de viabilidade irá utilizar dois projetos, cada projeto contendo artefatos diferentes: um Plano de Teste, cinco Casos de Teste e cinco Procedimento de Teste como objeto de inspeção; portanto, o estudo pode não ser representativo da teoria sob a qual se sustenta, uma vez que pode não ficar claro se a causa dos resultados do estudo é decorrente da aplicação de *TestCheck* ou da “facilidade” de detecção de defeitos no documento utilizado.

- **Teste:** como forma de evitar possíveis influências no desempenho dos participantes, o objetivo do estudo não será apresentado aos participantes. Além disso, os participantes não serão envolvidos com discussão sobre as supostas vantagens e desvantagens da utilização de *TestCheck* e de abordagens de avaliação de artefatos de teste em geral. Entretanto, devido à presença da ameaça "Interação entre Tratamentos Diferentes", descrita anteriormente, não é possível garantir a ausência dessa ameaça.
- **Apreensão de Avaliação:** uma possível ameaça é a probabilidade dos participantes "maquiarem" seus dados devido a expectativas pessoais sobre a avaliação de seus resultados. Entretanto, essa ameaça não será considerada visto que o estudo será conduzido por alunos que participarão de forma espontânea, sem que seu resultado seja utilizado para avaliação dos alunos na disciplina.

### 4.5.4. Validade Externa

Questões relacionadas à ameaça dos resultados do estudo não serem generalizáveis a projetos reais na indústria. As principais ameaças identificadas foram:

- **Interação entre seleção e tratamento:** acadêmicos podem não ser substitutos para inspetores da indústria, por não possuírem experiência industrial. A amostra dos participantes selecionados é bastante heterogênea. Boa parte dos alunos possui experiência em inspeção de artefatos de software em ambiente industrial, quanto à outra parte dos alunos não podemos afirmar que os mesmos não possuindo experiência em inspeções na indústria não podem apresentar habilidades similares a inspetores menos experientes. Pesquisas têm revelado que muitos dos conceitos de Engenharia de Software ainda não estão difundidos e aplicados adequadamente na indústria de software brasileira (VILLELA, 2004).
- **Interação entre Ambiente e Tratamento:** ambiente acadêmico pode não simular integralmente as condições existentes em um ambiente de desenvolvimento industrial, porém por utilizar documentos de teste criados também em um ambiente acadêmico, acreditamos que essa ameaça não seja concretizada.

## 4.6. Resultado do Estudo de Viabilidade

Após as listas de discrepâncias terem sido integradas a uma única lista e ser feita a classificação entre defeitos e falso-positivos, foi realizada a análise dos dados. A Tabela 10 apresenta os resultados individuais obtidos por cada participante do estudo em relação às métricas analisadas neste estudo. O cálculo realizado para encontrar a eficácia é dado por Defeitos / Discrepância (BARCELOS, 2006) e eficiência pela fórmula Defeitos / Tempo

(KOLLANUS et al., 2009; BARCELOS et al., 2006). Discrepâncias equivalentes apontadas por mais de um inspetor eram diretamente classificadas como defeitos e apenas um registro era mantido. As discrepâncias restantes foram analisadas e avaliadas, sendo classificada como defeitos ou falso-positivos. Para esta análise, houve a participação do pesquisador envolvido e um especialista em teste de software para definição da lista final de defeitos.

**Tabela 10. Dados obtidos durante a execução do estudo.**

| Técnica | Projeto/ Grupo | Inspetor | Defeitos | Falso-Positivos | Tempo (*minutos*) | Eficácia (*def / dis*) | Eficiência (*def / min*) |
|---|---|---|---|---|---|---|---|
| *Ad Hoc* | Acadêmico (Grupo A) | P01 | 0 | 15 | 210 | 0,00 | 0,00 |
| | | P02 | 9 | 6 | 285 | 0,60 | 0,03 |
| | | P03 | 2 | 15 | 76 | 0,12 | 0,03 |
| | | P04 | 0 | 4 | 42 | 0,00 | 0,00 |
| | | P05 | 21 | 3 | 120 | 0,88 | 0,18 |
| | | P06 | 6 | 22 | 180 | 0,21 | 0,03 |
| | | P07 | 8 | 11 | 100 | 0,42 | 0,08 |
| *Ad Hoc* | Industrial (Grupo B) | P08 | 28 | 0 | 75 | 1,00 | 0,37 |
| | | P09 | 4 | 6 | 120 | 0,40 | 0,03 |
| | | P10 | 28 | 12 | 121 | 0,70 | 0,23 |
| | | P11 | 24 | 4 | 100 | 0,86 | 0,24 |
| | | P13 | 6 | 16 | 143 | 0,27 | 0,04 |
| *TestCheck* | Industrial (Grupo A) | P01 | 40 | 7 | 190 | 0,85 | 0,21 |
| | | P02 | 32 | 10 | 220 | 0,76 | 0,15 |
| | | P03 | 30 | 10 | 34 | 0,75 | 0,88 |
| | | P04 | 43 | 9 | 135 | 0,83 | 0,32 |
| | | P05 | 28 | 2 | 127 | 0,93 | 0,22 |
| | | P06 | 7 | 6 | 240 | 0,54 | 0,03 |
| | | P07 | 36 | 17 | 180 | 0,68 | 0,20 |
| *TestCheck* | Acadêmico (Grupo B) | P08 | 19 | 8 | 155 | 0,70 | 0,12 |
| | | P09 | 36 | 12 | 85 | 0,75 | 0,42 |
| | | P10 | 57 | 27 | 122 | 0,68 | 0,47 |
| | | P11 | 84 | 20 | 180 | 0,81 | 0,47 |
| | | P13 | 58 | 12 | 275 | 0,73 | 0,21 |

Após a execução do estudo, iniciaram-se as análises dos resultados obtidos. A análise quantitativa foi feita em relação às variáveis defeitos, falsos positivos, tempo, eficiência (defeitos / discrepância), sendo as duas últimas mais evidenciadas, conforme descrito nas próximas subseções.

### 4.6.1. Análise das Médias

A primeira análise realizada neste estudo foi a análise das médias obtidas para todas as variáveis. A Tabela 11 apresenta os resultados deste estudo, com os valores totais obtidos para cada variável e suas médias (entre parênteses).

**Tabela 11. Análise da Média do Estudo de Viabilidade 1.**

| Variável | Técnicas de Inspeção: Valor (Média) | | Projetos de Software: Valor (Média) | |
|---|---|---|---|---|
| | ***Ad Hoc*** | ***TestCheck*** | **Industrial** | **Acadêmico** |
| Defeitos | 136 (11,33) | 470 (39,17) | 306 (25,50) | 300 (25,00) |
| Falso-Positivos | 114 (9,5) | 140 (11,7) | 99 (8,25) | 165 (13,75) |
| Discrepâncias | 250 (20,83) | 610 (50,83) | 405 (33,75) | 455 (37,92) |
| Tempo | 1572 (131,0 min) | 1943 (161,9 min) | 1685 (140,4 min) | 1830 (152,5 min) |
| Eficácia | 0,54 def/disc | 0,77 def/disc | 0,75 def/disc | 0,65 def/disc |
| Eficiência | 0,08 def/min | 0,24 def/min | 0,18 def/min | 0,16 def/min |

Analisando a variável Defeitos, observou-se uma diferença significativa na média de defeitos encontradas por abordagem utilizada, sendo que a abordagem *TestCheck* apresentou melhores resultados quando comparada a *Ad Hoc* (39,17 x 11,33), enquanto que os projetos de software apresentaram médias similares para o número de defeitos (~25 defeitos).

Analisando a variável Falso-Positivos, é possível observar uma diferença significativa na média de falso-positivos encontrados por técnica. *TestCheck* gerou mais falso-positivos quando comparada à *Ad Hoc* (11,7 x 9,5), da mesma forma que os projetos de software industrial e acadêmico apresentaram diferença significativa na média (8,25 x 13,75) respectivamente.

Analisando a variável Tempo, observou-se uma diferença significativa na média de tempo para realização da inspeção por técnica utilizada. Neste estudo, *TestCheck* requer em média 30 minutos a mais para ser aplicada, o que era esperado devido aos instrumentos (*Checklists*) a serem aplicado pelos inspetores, enquanto que usando a técnica *Ad Hoc* nenhum instrumento adicional é provido. Analisando os projetos, observou-se que ambos obtiveram médias de tempo similares, com 12 minutos de variação entre eles.

Analisando a Eficácia, observa que *TestCheck* também possuiu, em média, maior eficácia nesta medida (0,77 x 0,54). Porém, observou-se que um resultado similar aconteceu analisando os projetos (o projeto industrial apresentou eficácia bem acima do projeto acadêmico – 0,75 x 0,65).

De forma similar, ao analisar a variável Eficiência, observa que *TestCheck* também possuiu, em média, maior eficácia nesta medida (0,77 x 0,54). Porém, observou-se um

resultado similar ao obtido para eficácia, com *TestCheck* obtendo uma eficiência maior que a abordagem *Ad Hoc* (0,24 x 0,08), enquanto que os projetos de software obtiveram eficiência similares.

### 4.6.2. Testes Estatísticos Aplicados nas Variáveis

A análise do estudo foi feita seguindo alguns passos, para cada variável avaliada:

1. Primeiramente foi realizada a Análise de Normalidade (*Shapiro-Wilk Test*) com o objetivo de avaliar se a distribuição obtida era normal ou não.
2. Para as distribuições normais, aplicou-se o Teste de Homocedasticidade (*Levene Test*) para verificar se as variâncias obtidas nos dados dos participantes eram homogêneas ou heterogêneas entre si.
    a. Para as distribuições homogêneas (homocedásticas), foi aplicado o teste estatístico Análise de Variância (ANOVA), conforme descrito nos trabalhos de DIAS-NETO (2006) e CONTE (2009).
    b. Para as distribuições heterogêneas, aplicou-se o teste estatístico *Welch ANOVA*.
3. Para variáveis não normais, aplicou-se o teste não paramétrico (*Wilcoxon/Kruskal Wallis*). Para isso, utilizou-se a ferramenta JMP 4[2] para apoiar a análise estatística.

- **Teste de Normalidade**

A Tabela 12 apresenta os resultados dos testes de normalidade e homocedasticidade para as variáveis analisadas neste estudo. Nestes resultados, observou-se que as variáveis Falso-Positivo, Tempo e Eficácia possuem distribuição Normal, podendo ser comprovado pelo valor do *p-value* que ficou acima de 0,05. Portanto foi preciso aplicar o teste de homocedasticidade nestas variáveis, que revelou que todas elas são homocedásticas. Assim, será necessário aplicar ANOVA para teste estatístico dessas variáveis. Já as variáveis Defeitos e Eficiência não possuem distribuição Normal. Assim, foi necessário aplicar testes não paramétricos nesta variáveis.

Analisando os projetos, obteve-se um resultado similar ao obtido para as técnicas de inspeção, conforme a Tabela 12.

**Tabela 12. Teste de Normalidade e Homocedasticidade**

| Análise das Técnicas | | | | Análise dos Projetos | | |
|---|---|---|---|---|---|---|
| **Variáveis Dependentes** | ***Shapiro-Wilk Test*** | | ***Levene Test*** | ***Shapiro-Wilk Test*** | | ***Levene Test*** |
| | ***Ad Hoc*** | ***TestCheck*** | | **Industrial** | **Acadêmico** | |
| ***Defeitos*** | *0,0309* | *0,5351* | *NT* | 0,1000 | *0,0301* | 0,0173 |
| **Falso-Positivo** | 0,5400 | 0,4004 | 0.8187 | 0,8904 | 0,6877 | 0,1160 |
| **Tempo** | 0,2081 | 0,9949 | 0.8475 | 0,1402 | 0,5957 | 0,3089 |
| **Eficácia** | 0,4744 | 0,8483 | 0.1331 | 0,3669 | 0.0671 | 0,0640 |
| ***Eficiência*** | *0,0110* | *0,0509* | *NT* | 0,0027 | *0,0123* | 0,1001 |

- **Teste Paramétrico (ANOVA) aplicado à Variável FALSO-POSITIVOS**

Conforme a Tabela 12, a distribuição dos dados desta variável é Normal e Homocedástica. Logo, um Teste Paramétrico (ANOVA) foi aplicado. Apesar de *TestCheck* ter gerado mais falso-positivos que a técnica *Ad Hoc,* estatisticamente não existe diferença entre as técnicas utilizadas (lado esquerdo da Figura 19 – *p-value* = 0,3078). No entanto, é um fator que foi analisado posteriormente, pois por algum motivo a técnica induziu os inspetores ao erro. Da mesma forma, analisando os resultados obtidos por projeto de software (lado direito da Figura 19), percebeu-se que o projeto não foi um fator que influenciou na quantidade de falsos positivos encontrados durante a inspeção (*p-value* = 0,0539).

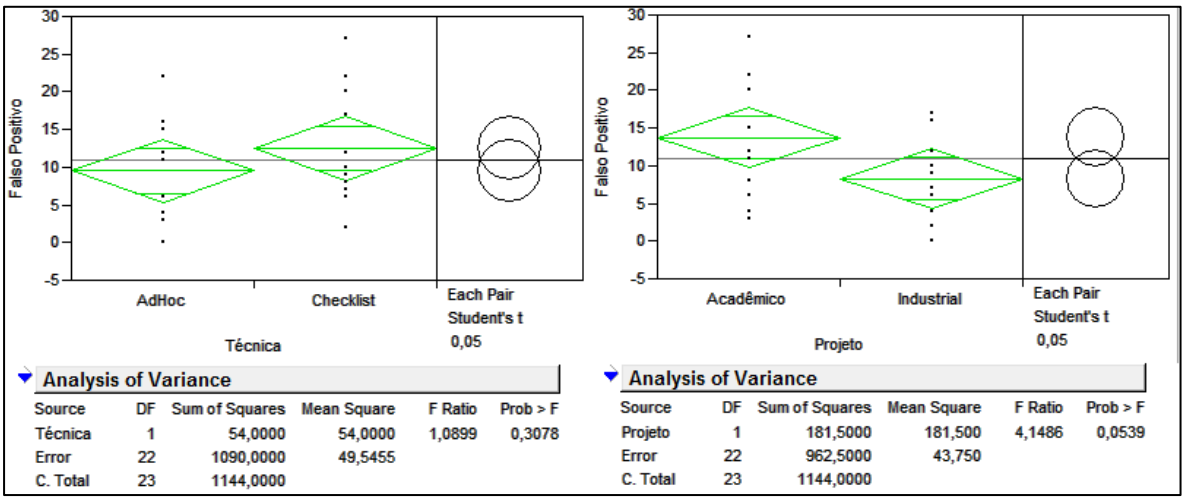



**Figura 19. ANOVA para a variável falso-positivos**

- **Teste Paramétrico (ANOVA) aplicado à Variável TEMPO**

Conforme a Tabela 12, a distribuição dos dados desta variável também é Normal e Homocedástica. Logo, novamente um Teste Paramétrico (ANOVA) foi aplicado. Ao analisar a variável tempo, constatou-se que a utilização do *TestCheck* requereu mais

tempo comparando-o com a inspeção utilizando a técnica A*d Hoc*. No entanto, a análise apresentada na Figura 20 indica que apesar de requerer mais tempo, esta não possui diferença estatística. O *p-value* obtido para esta variável (lado esquerdo da figura) ficou acima de 0,05 (*p-value* = 0,2697).

Analisando os resultados obtidos por projetos de software (lado direito da Figura 20), percebeu-se que o projeto não foi um fator que influenciou no tempo gasto utilizado para inspecionar os artefatos dos projetos (*p-value* = 0,6699).

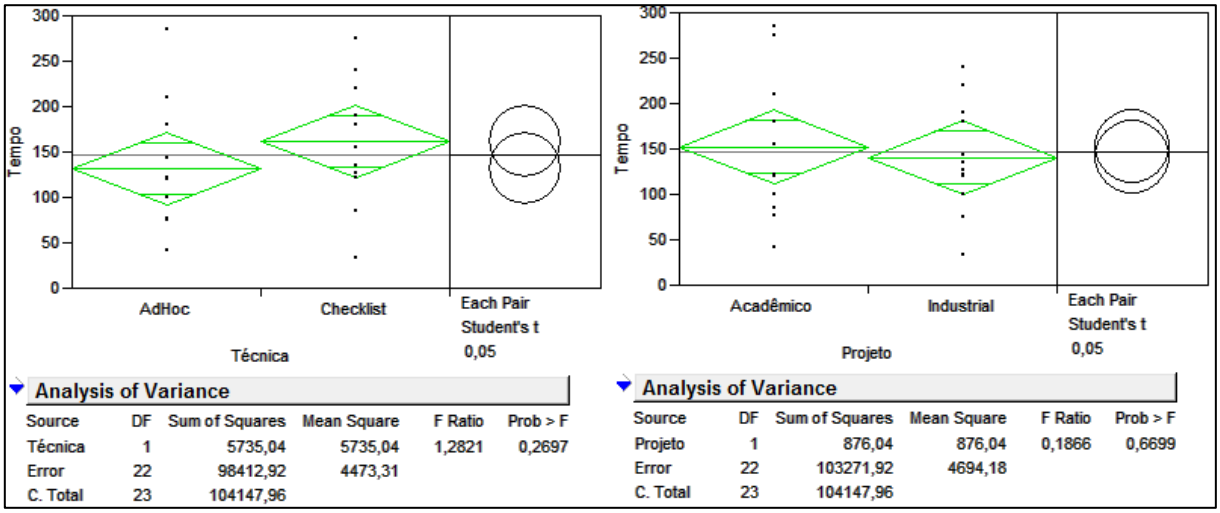



**Figura 20. ANOVA para a variável Tempo**

- **Teste Paramétrico (ANOVA) aplicado à Variável EFICÁCIA**

Conforme a Tabela 12, a distribuição dos dados desta variável também é Normal e Homocedástica. Logo, novamente um Teste Paramétrico (ANOVA) foi aplicado. Analisando o resultado para a variável Eficácia, foi possível constatar que existe diferença estatística entre as abordagens *TestCheck* e a técnica *Ad Hoc*, indicando que *TestCheck* provê uma melhor detecção de defeitos em relação ao número total de discrepância identificadas quando comparada à técnica *Ad Hoc*. Esta confirmação pode ser feita analisando o *p-value* obtido para esta variável (lado esquerdo da Figura 21), que ficou abaixo de 0,05 (*p-value* = 0,0101).

Analisando os resultados obtidos por projetos de software (lado direito da Figura 21), percebe-se que o projeto não foi um fator que influenciou na eficácia da inspeção. O valor do *p-value* de projetos ficou acima de 0,05 (*p-value* = 0,0616) e os círculos que representam os projetos possuem interseção entre si, indicando que não existe diferente estatística significativa entre eles.

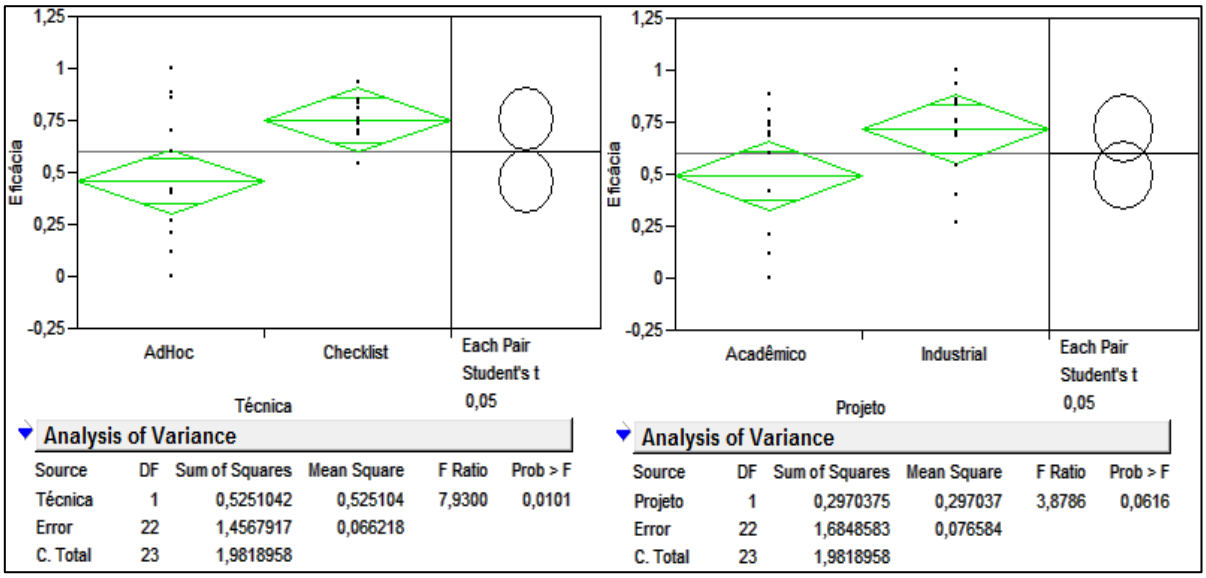



**Figura 21. ANOVA para a variável Eficácia.**

- **Teste Não Paramétrico (*Wilcoxon/Kruskal Wallis*) aplicado às Variáveis Defeitos e Eficiência**

Após aplicar o teste de normalidade ficou constatado que as variáveis Defeitos e Eficiência não possuem distribuição normal (Tabela 12). Portanto, foi aplicado teste não paramétrico para análise destas variáveis, cujo resultado está representado na Tabela 13.

Analisando o resultado para variável Defeitos, percebeu-se que existe diferença estatística entre as técnicas de inspeção avaliadas, podendo ser comprovada pelo valor do *p-value* que ficou abaixo de 0,05 (*p-value* = 0,004), enquanto que a análise por projeto de software não indicou diferença estatística entre eles (*p-value* = 0,4522). Portanto, os resultados indicam que a técnica *TestCheck* neste estudo possibilitou o aumento da taxa de detecção de defeitos quando comparada à técnica *Ad Hoc*.

Analisando a variável Eficiência, percebeu-se novamente que existe diferença estatística entre as técnicas de inspeção avaliadas, pois o valor do *p-value* ficou abaixo de 0,05 (*p-value* = 0,0266). Analisando os projetos, observou-se que estes não influenciaram nos resultados deste estudo, ficando comprovado através do *p-value* que ficou acima de 0,05 (*p-value* = 0,1001).

**Tabela 13. Teste não paramétrico para as variáveis Defeitos e Eficiência.**

| Variáveis Dependentes | *Wilcoxon/Kruskal Wallis Test* | |
|---|---|---|
| | **Por Técnica de Inspeção** | **Por Projeto de Software** |
| **Defeitos** | *p-value* = 0,0004 | *p-value* = 0,4522 |
| **Eficiência** | *p-value* = 0,0266 | *p-value* = 0,1001 |

Dessa forma, observou-se que a técnica *TestCheck* foi, neste estudo, o fator que influenciou estatisticamente na detecção de defeitos nos artefatos de teste de software.

Em seguida, foi realizada uma análise a respeito da cobertura dos defeitos encontrados por *TestCheck*, descrita na próxima subseção.

### 4.6.3. Análise da Cobertura de defeitos

Para analisar a cobertura dos defeitos detectados por cada técnica, foram removidos defeitos duplicados (encontrado por mais de um inspetor). Em seguida, separou-se defeitos detectados por cada ou ambas as técnicas. O resultado desta análise está apresentado na Figura 22.

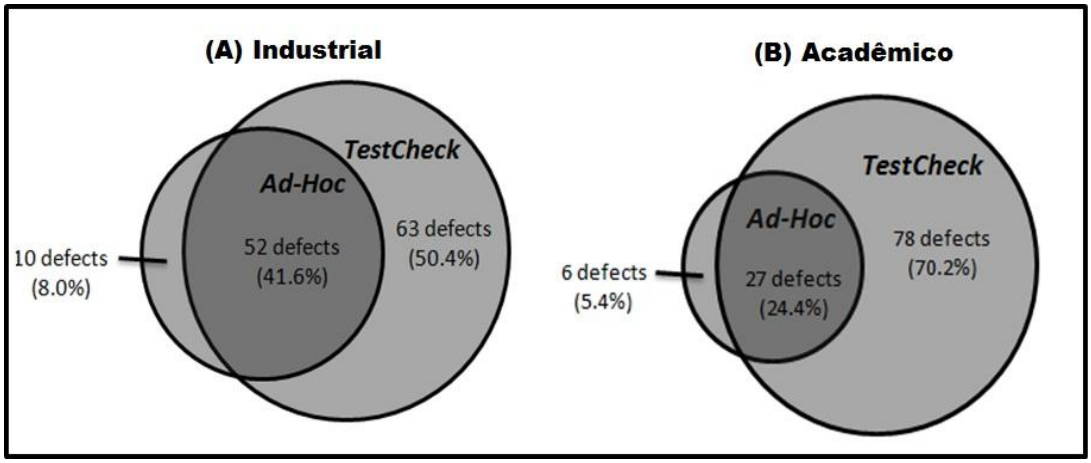



**Figura 22. Análise de interseção de defeitos detectados pelas técnicas (BRITO e DIAS-NETO, 2012).**

É possível observar que defeitos foram detectados pela técnica *Ad Hoc*, mas não foram detectados por *TestCheck*. Esses defeitos não estão relacionados a um item de avaliação específico que não estaria sendo suficientemente coberto por *TestCheck*. Eles foram identificados pelos participantes mais experientes a partir de seu conhecimento anterior em testes de software. Este é um comportamento normal em inspecção software e não pode ser prevista. Por outro lado, observou-se que *TestCheck* cobria a maior parte dos defeitos detectados pela técnica *Ad Hoc*. Além disso, um elevado número de defeitos foram encontradas apenas por *TestCheck*, o que sugere que esta contribui para detectar defeitos novos e não triviais que não são detectados por uma técnica *Ad Hoc* (BRITO e DIAS-NETO, 2012).

Considerando os diferentes tipos de defeitos detectados por *TestCheck* (Figura 23), observou-se que a distribuição dos tipos se assemelhou a resultados obtidos em outros estudos realizados na área de inspeção de software, como (Davis, 1990; Kalinowski et al., 2007).

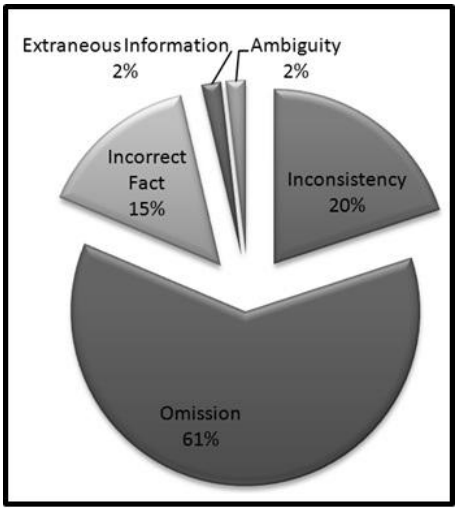



**Figura 23. Distribuição dos tipos de Defeitos detectados por *TestCheck* (BRITO e DIAS-NETO, 2012)**

### 4.6.4. Evolução de *TestCheck* a partir do Primeiro Estudo

Apesar do bom desempenho obtido por *TestCheck* no primeiro estudo de avaliação, entende-se que a técnica está em processo de amadurecimento. Então, foram analisados e avaliados pontos onde melhorias poderiam ser introduzidas. Um ponto que chamou bastante atenção foi a quantidade de falso-positivos gerados a partir da aplicação de *TestCheck*, sendo um possível candidato a melhoria. Foi obtida uma taxa de 11,7 falso-positivos por participante, considerada alta pelos pesquisadores, visto que a taxa de defeitos por participante detectados pela técnica foi de 39,16. Com isso, aproximadamente, a cada 3 defeitos, 1 falso-positivo foi gerado por *TestCheck*. Portanto, o objetivo principal da evolução de *TestCheck* em um primeiro momento foi diminuir a taxa de falso-positivos encontrados no primeiro estudo realizado e manter/aumentar a taxa de defeitos detectados, quando comparada à técnica *Ad Hoc*.

Assim, foi realizada uma análise individual para cada questão que compõe os *Checklists* que formam *TestCheck*. O objetivo foi identificar questões candidatas a serem evoluídas por terem tido um baixo desempenho durante o estudo. Para esta análise, foi considerada a relação entre o número de falso-positivos que a questão gerou e a quantidade de defeitos detectados por ela. Todas as questões dos 3 *Checklists* que compõem *TestCheck* que tiveram uma taxa de defeitos por falso-positivo maior ou igual a 3,0 (valor este obtido no cenário geral, conforme citado anteriormente), seriam potenciais candidatas a serem evoluídas para a próxima versão da técnica. Tal análise está descrita na Tabela 14, onde as questões que atingiram este patamar estão destacadas em cinza. Os itens que não geraram nenhum falso positivo não foram incluídos na análise e não foram apresentados nesta tabela para simplificar a descrição.

Ao total, 31 questões (de um total de 75 questões que compõem os 3 *Checklists*) foram selecionadas como candidatas à evolução e foram revisadas e algumas reescritas para a criação de uma nova versão de *TestCheck*. Buscou-se ao criar os questionamentos que estes fossem coerentes, com termos adequados ao contexto, e simplicidade, para impossibilitar várias interpretações. Algumas questões requereram maior atenção, como é o caso das questões (destacadas em negrito na Tabela 14) P02, P08 e P09 do plano de teste, C10 do caso de teste e o P07 do procedimento de teste, que tiveram uma alta taxa de falso-positivos. Vários fatores poderiam ter influenciado neste resultado, dentre os quais a forma como a questão foi escrita. Para uma melhor avaliação, pessoas alheias à pesquisa manifestaram opinião em relação ao entendimento, clareza, simplicidades das questões após sua leitura.

**Tabela 14. Defeitos e Falso-Positivos por questão de *TestCheck* (versão 1.0)**

| **Plano de Teste** | | | **Caso de Teste** | | | **Procedimento de Teste** | | |
|---|---|---|---|---|---|---|---|---|
| Questão | Defeitos | Falso Positivo | Questão | Defeitos | Falso Positivo | Questão | Defeitos | Falso Positivo |
| **P02** | 0 | 3 | C02 | 1 | 1 | P01 | 3 | 1 |
| P04 | 1 | 2 | C06 | 44 | 6 | P02 | 2 | 1 |
| P05 | 1 | 3 | C07 | 11 | 1 | P04 | 4 | 3 |
| P07 | 2 | 2 | C08 | 1 | 2 | **P07** | **0** | **40** |
| **P08** | 0 | 3 | **C10** | **42** | **18** | P08 | 3 | 1 |
| **P09** | 0 | 3 | C11 | 1 | 1 | P12 | 0 | 1 |
| P11 | 1 | 2 | C12 | 5 | 1 | P13 | 2 | 2 |
| P13 | 1 | 2 | C13 | 44 | 1 | P14 | 3 | 4 |
| P14 | 0 | 1 | C15 | 2 | 5 | P15 | 3 | 2 |
| P19 | 4 | 1 | C19 | 10 | 2 | P16 | 0 | 2 |
| P20 | 0 | 1 | | | | | | |
| P21 | 1 | 1 | | | | | | |
| P22 | 1 | 1 | | | | | | |
| P26 | 1 | 2 | | | | | | |
| P30 | 1 | 2 | | | | | | |
| P31 | 0 | 1 | | | | | | |
| P32 | 0 | 2 | | | | | | |
| P33 | 6 | 1 | | | | | | |

Com base nos comentários sugeridos pelos leitores e pela análise feita pelo experimentalista e um especialista em teste foi criado de uma nova versão de *TestCheck* (versão 2.0). Esta nova versão possui questões reformuladas, com o objetivo de facilitar o entendimento e duas questões foram removidas por entender que outras questões tinham o mesmo sentido e levava o inspetor a identificar o mesmo defeito.

## 4.7. Considerações Finais

Este capítulo apresentou o estudo de viabilidade de *TestCheck*, correspondente a primeira fase da metodologia de SHULL (2001). Seu objetivo foi analisar a viabilidade de *TestCheck* em relação à detecção de defeitos, conduzido em ambiente acadêmico, realizado por estudante de graduação e pós graduação. A partir dos resultados obtidos, foi descrito como ocorreu o processo de evolução da técnica, a fim de gerar uma nova versão dos *Checklists* que compõe a técnica de inspeção.

O próximo capítulo descreve ainda a primeira fase da metodologia experimental, correspondente ao segundo estudo de viabilidade. Este foi conduzido seguindo os passos do primeiro estudo, agora utilizando a nova versão de *TestCheck*. Seu objetivo foi avaliar a diminuição da quantidade de falso positivos gerados no primeiro estudo.

# CAPÍTULO 5: SEGUNDO ESTUDO DE VIABILIDADE PARA AVALIAÇÃO DE *TESTCHECK*

*Este capítulo descreve o segundo estudo realizado, continuando na primeira etapa da metodologia adotada, que consiste em um segundo estudo de viabilidade onde foi aplicada uma seção de inspeção com o objetivo de diminuir a taxa de falso-positivos quando utilizando a técnica TestCheck*

## 5.1 Introdução

Para avaliar a evolução de *TestCheck* a partir da nova versão produzida após o primeiro estudo de viabilidade, fez-se necessário avaliar os resultados obtidos em um segundo estudo experimental (segundo estudo de viabilidade) para analisar se as mudanças implementadas refletem em melhorias para a técnica proposta. O segundo estudo foi realizado seguindo os moldes do primeiro, onde o objetivo foi avaliar a aplicação de *TestCheck* quando comparada, novamente, a uma técnica *Ad Hoc*. Ele constituiu-se de duas rodadas de inspeções utilizando as técnicas *Ad Hoc* e *TestCheck* e foi realizado em Dezembro de 2011 em ambiente acadêmico.

O foco deste estudo foi a avaliação da redução da taxa de falso-positivos gerados e manutenção da taxa de detecção de defeitos maior que a taxa obtida para a técnica *Ad Hoc*.

## 5.2 Definição do Segundo Estudo de Viabilidade

Com base nos resultados obtidos com o estudo de viabilidade, que demonstrou indícios da viabilidade de *TestCheck* na detecção de defeitos em artefatos de teste de software, este segundo estudo tem o propósito de amadurecer a técnica e diminuir a taxa de falso-positivos identificados pela técnica sem impactar na sua eficiência e eficácia quando comparada a uma técnica *Ad Hoc*. Portanto ao final deste estudo será possível responder a questões:

1. A aplicação de *TestCheck* para avaliação de artefatos de teste mantém o resultado do primeiro estudo em termos de eficiência e eficácia em relação à identificação de defeitos?
2. As melhorias realizadas na técnica reduziram a quantidade de falsos positivos encontrados?

Assim, formalizando o objetivo deste segundo estudo, tem-se:

- **Analisar** a aplicação da abordagem *TestCheck*.
- **Com o propósito de** compreender.
- **Em relação** à redução da taxa de falso-positivos e manutenção de sua eficácia e eficiência.
- **Do ponto de vista** da pesquisadora.
- **No contexto da** inspeção de artefatos de teste (Plano, Caso e Procedimento de teste) retirados de repositórios da Internet.

## 5.2 Planejamento e Execução do Segundo Estudo de Viabilidade

O segundo estudo de viabilidade foi conduzido seguindo os passos realizados no primeiro estudo. Ele foi realizado em ambiente acadêmico, onde os participantes foram estudantes de graduação provenientes de uma disciplina de Engenharia de Software da Universidade Federal do Amazonas do curso de Sistemas de Informação e Ciência da Computação. Ao total, 19 alunos manifestaram interesse em participar. Estes foram divididos em dois grupos (A) de 9 e outro (B) com 10 alunos. Para distribuir esses alunos em grupos, foi avaliado o formulário de caracterização, mesclando cada equipe de acordo com conhecimento e grau de experiência de cada participante.

Foram utilizados os mesmos projetos de software utilizando no primeiro estudo (*Industrial* e *Acadêmico*) e seus artefatos de teste (1 plano de teste, 5 casos de teste, 5 procedimento de teste, documento de requisitos para quaisquer esclarecimentos sobre os artefatos de teste). O mesmo procedimento foi seguido para execução da primeira inspeção utilizando a técnica *Ad Hoc* e consolidação da lista final de discrepâncias. A inspeção também foi realizada de forma *off-line*, evitando dessa forma que informações fossem trocadas entre os participantes.

Os membros da equipe A (10 pessoas) receberam o projeto Acadêmico e a equipe B (9 pessoas) ficou com o projeto Industrial. Após a execução da atividade de inspeção, os participantes das equipes (A e B) entregaram as listas de discrepância, sendo que houve 9 desistências, ficando apenas 6 pessoas na equipe A e 4 na equipe B, aptos a receber o treinamento para utilização da segunda abordagem, *TestCheck*, em artefatos de teste.

O treinamento ocorreu de forma similar o primeiro estudo. Após o treinamento, os projetos foram distribuídos entre as equipes. Nesse momento, foram invertidos os projetos, de forma que os participantes do Grupo A inspecionaram os artefatos de teste do Projeto Industrial, enquanto que os participantes do Grupo B inspecionaram os artefatos de teste do Projeto Acadêmico, ambos utilizando *TestCheck* como abordagem de apoio. Da mesma forma, a inspeção foi conduzida de forma off-line. Por este estudo apresentar características semelhantes ao primeiro estudo, às ameaças a validade foram simplesmente replicadas.

## 5.3 Resultado do Segundo Estudo de Viabilidade

Para avaliação da técnica *TestCheck*, foram coletados os dados para as mesmas variáveis analisadas no estudo 1, descrito na seção 4.6.1. A Tabela 15 apresenta um resumo dos dados obtidos para cada participante.

**Tabela 15. Dados Obtidos no Segundo Estudo de Viabilidade**

| Participantes | Técnica de Inspeção | Projeto | Defeitos | Falso-Positivo | Tempo (minutos) | Eficácia (*def* / *dis*) | Eficiência |
|---|---|---|---|---|---|---|---|
| P01 | *Ad Hoc* | Industrial | 38 | 4 | 75 | 0,90 | 0,51 |
| P02 | | | 4 | 2 | 40 | 0,67 | 0,10 |
| P03 | | | 6 | 1 | 95 | 0,86 | 0,06 |
| P04 | | | 0 | 5 | 80 | 0,00 | 0,00 |
| P05 | | Acadêmico | 9 | 1 | 130 | 0,90 | 0,07 |
| P06 | | | 5 | 1 | 58 | 0,83 | 0,09 |
| P07 | | | 12 | 1 | 60 | 0,92 | 0,20 |
| P08 | | | 8 | 2 | 60 | 0,80 | 0,13 |
| P09 | | | 17 | 2 | 176 | 0,89 | 0,10 |
| P10 | | | 20 | 10 | 458 | 0,67 | 0,04 |
| P01 | *TestCheck* | Acadêmico | 56 | 8 | 330 | 0,88 | 0,17 |
| P02 | | | 22 | 2 | 180 | 0,92 | 0,12 |
| P03 | | | 40 | 8 | 230 | 0,83 | 0,17 |
| P04 | | | 22 | 1 | 240 | 0,96 | 0,09 |
| P05 | | Industrial | 14 | 5 | 60 | 0,74 | 0,23 |
| P06 | | | 8 | 2 | 134 | 0,80 | 0,06 |
| P07 | | | 29 | 6 | 342 | 0,83 | 0,08 |
| P08 | | | 20 | 4 | 200 | 0,83 | 0,10 |
| P09 | | | 76 | 11 | 370 | 0,87 | 0,21 |
| P10 | | | 63 | 2 | 360 | 0,97 | 0,18 |

### 5.3.1 Análise de *TestCheck* em Relação ao Número de Falso-Positivos

Como o foco principal deste segundo estudo foi avaliar a evolução de *TestCheck* em relação à quantidade de falso-positivos encontrados por inspetor e a manutenção da taxa de detecção de defeitos, primeiramente este aspecto será avaliado.

A Tabela 16 apresenta um comparativo entre as técnicas de inspeção focando na quantidade de defeitos detectados e falso-positivos gerados por participante. A taxa de falso-positivos no primeiro estudo foi de 3:1, ou seja, a cada três defeitos um falso positivo era gerado. No segundo estudo essa taxa foi de 7:1, a cada sete defeitos encontrados um falso positivo era gerado, o que permite constatar que para este estudo a evolução da abordagem conseguiu alcançar o objetivo em relação a diminuição da taxa de falso positivo, sem, no entanto, comprometer o segundo objetivo do estudo que foi manter a taxa de detecção de defeitos de *TestCheck* alta quando comparada à técnica *Ad Hoc*.

**Tabela 16. Resultados comparativos das técnicas.**

| Medidas | *Ad Hoc* | | *TestCheck* | |
|---|---|---|---|---|
| | **Falso-Positivos** | **Defeitos** | **Falso-Positivos** | **Defeitos** |
| **Total** | 29 | 119 | 49 | 350 |
| **Média** | 2,9 | 11,9 | 4,9 | 35,0 |
| **Desvio Padrão** | 2,846 | 10,969 | 3,314 | 22,852 |

Analisando os dados (número de defeitos e falso-positivos) obtidos individualmente para cada questão que compõem os 3 *Checklists* que formam *TestCheck*, foi possível perceber uma melhor distribuição dos falso-positivos, não ficando concentrado em um número reduzido de questões, como ocorrido no primeiro estudo. No entanto, questões, destacadas na Tabela 17, que geraram a partir de 3 falso-positivos devem ser investigadas e evoluídas em uma nova versão de *TestCheck*. Questões que não identificaram nenhum falso positivo, não foram omitidas na Tabela 17.

**Tabela 17. Defeitos e Falso-Positivos por questão de *TestCheck* (versão 2.0)**

| **Plano de Teste** | | | **Caso de Teste** | | | **Procedimento de Teste** | | |
|---|---|---|---|---|---|---|---|---|
| Questão | Defeitos | Falsos Positivos | Questão | Defeitos | Falsos Positivos | Questão | Defeitos | Falsos Positivos |
| P03 | 7 | 1 | C03 | 26 | 4 | P03 | 30 | 2 |
| P04 | 0 | 2 | C04 | 0 | 1 | P05 | 14 | 2 |
| P05 | 0 | 2 | C05 | 2 | 3 | P06 | 25 | 1 |
| P07 | 0 | 1 | C06 | 30 | 5 | P07 | 2 | 3 |
| P08 | 4 | 2 | C07 | 3 | 1 | P09 | 20 | 3 |
| P10 | 1 | 3 | C09 | 4 | 1 | P10 | 20 | 1 |
| P11 | 0 | 1 | C10 | 30 | 3 | P12 | 1 | 1 |
| P14 | 3 | 1 | C11 | 2 | 1 | P13 | 4 | 4 |
| P15 | 7 | 1 | C12 | 2 | 2 | P15 | 55 | 12 |
| P25 | 5 | 1 | C13 | 22 | 2 | | | |
| P26 | 2 | 1 | C14 | 0 | 1 | | | |
| P27 | 3 | 1 | C15 | 6 | 1 | | | |
| P28 | 3 | 1 | C17 | 25 | 3 | | | |
| P34 | 2 | 1 | C18 | 1 | 3 | | | |
| P38 | 1 | 1 | | | | | | |

Não se pode extinguir a presença de falso-positivos, pois este é um elemento que faz parte do processo de inspeção e depende de diversos fatores que não necessariamente podem ser controlados. No entanto, estes podem ser otimizados reduzindo alguns destes fatores que levam à ocorrência de falso-positivos, principalmente em relação à redação das questões que formam os *Checklists* eliminando ambiguidades que podem levar a múltiplas interpretações das questões apresentadas.

### 5.3.2 Resultado do Segundo Estudo de Viabilidade – Análise Quantitativa

A coleta e análise dos dados ocorreram da mesma forma realizada no primeiro estudo. Após a entrega das listas de discrepâncias, estas foram integradas a uma única lista que consequentemente foi analisada para verificar se as discrepâncias eram únicas ou estavam repetidas, em seguida as discrepâncias foram classificadas em defeitos ou falso-positivos. A análise foi realizada seguindo os moldes do primeiro estudo, portanto, será apresentada a análise das médias em seguida a análise dos testes estatísticos.

### 5.3.3 Análise das Médias

Foi realizada primeiramente a análise das médias obtidas para todas as variáveis e os resultados deste estudo, com os valores totais obtidos para cada variável e suas médias (entre parênteses) estão representados na Tabela 18.

**Tabela 18. Análise da Média do Estudo de Viabilidade 2.**

| Variável | Técnicas de Inspeção: Valor (Média) | | Projetos de Software: Valor (Média) | |
|---|---|---|---|---|
| | ***Ad Hoc*** | ***TestCheck*** | **Industrial** | **Acadêmico** |
| Defeitos | 119 (11,9) | 350 (35,0) | 258 (25,8) | 211 (21,1) |
| Falso-Positivos | 29 (2,9) | 49 (4,9) | 42 (4,2) | 36 (3,6) |
| Discrepâncias | 148 (14,8) | 399 (39,9) | 300 (30,0) | 247 (27,7) |
| Tempo | 1232 (123,2 min) | 2446 (244,6 min) | 1756 (175,6 min) | 1922 (192,2 min) |
| Eficácia | 0,80 def/disc | 0,88 def/disc | 0,86 def/disc | 0,85 def/disc |
| Eficiência | 0,09 def/min | 0,14 def/min | 0,15 def/min | 0,11 def/min |

Análise individual das variáveis:

- **Defeitos**: apresenta diferença significativa na média de defeitos encontrados por técnica de inspeção, sendo que *TestCheck* apresentou melhores resultados quando comparada à *Ad Hoc* (35,0 x 11,9). Já os projetos de software apresentaram médias similares para o número de defeitos (~21 defeitos).
- **Falso-Positivos:** existe uma pequena diferença na média de falso-positivos encontrados. *TestCheck* gerou mais falso-positivos quando comparada à *Ad Hoc* (4,9 x 2,9). A diferença entre os projetos manteve-se similar (4,2 x 3,6).
- **Tempo**: observou-se uma diferença na média de tempo para realização da inspeção por técnica utilizada. *TestCheck* requereu praticamente o dobro de tempo, no entanto esse foi um comportamento esperado devido aos instrumentos (*Checklists*) a serem aplicado pelos inspetores, enquanto que usando a técnica *Ad*

*Hoc* nenhum instrumento adicional é provido. Analisando os projetos, observou-se que ambos obtiveram médias de tempo similares, aproximadamente 183,9 minutos.

- **Eficácia:** em média *TestCheck* possui maior eficácia (0,77 x 0,54) em relação a técnica *AdHoc*. Analisando o resultado dos projetos, observou-se resultado similar (0,85 x 0,86).
- **Eficiência**: *TestCheck* possuiu, em média, maior eficácia nesta medida (0,77 x 0,54). Porém, observou-se um resultado similar ao obtido para eficácia, com *TestCheck* obtendo uma eficiência maior que a técnica *Ad Hoc* (0,14 x 0,09). Os projetos de software obtiveram eficiências similares em média 0,13.

### 5.3.4 Testes Estatísticos aplicados às Variáveis

Após análise das médias variável a variável, iniciou-se a análise estatística dos resultados obtidos. A análise quantitativa foi feita em relação às variáveis defeitos, falsos positivos, tempo, eficácia e eficiência seguindo os passos do primeiro estudo de viabilidade: foi realizada a Análise de Normalidade (*Shapiro-Wilk Test*) com o objetivo de avaliar se a distribuição obtida era normal ou não. Para as distribuições normais, aplicou-se o Teste de Homocedasticidade (*Levene Test*) para verificar se as variâncias obtidas nos dados dos participantes eram homogêneas ou heterogêneas entre si. Para as distribuições homogêneas (homocedásticas), foi aplicado o teste estatístico Análise de Variância (ANOVA). Para as distribuições heterogêneas, aplicou-se o teste estatístico *Welch ANOVA*. Para variáveis não normais, aplicou-se o teste não paramétrico (*Wilcoxon/Kruskal Wallis Test*). Novamente, utilizou-se a ferramenta JMP 4 para apoiar a análise estatística.

- **Teste de Normalidade**

A Tabela 19 apresenta os resultados dos testes de normalidade e homocedasticidade para as variáveis analisadas neste estudo. Nestes resultados, observou-se que a variável Defeitos possui distribuição Normal, podendo ser comprovado pelo valor do *p-value* que ficou acima de 0,05. Portanto foi preciso aplicar o teste de homocedasticidade nesta variável, que revelou-se homocedásticas. Assim, será necessário aplicar ANOVA para teste estatístico dessa variável. Para as demais variáveis analisados no estudo, os resultados indicaram que estas não possuem distribuição Normal. Assim, foi necessário realizar testes não paramétricos.

**Tabela 19. Teste de Normalidade e Homocedasticidade**

| Análise da Técnica de Inspeção | | | | Análise dos Projetos de Software | | |
|---|---|---|---|---|---|---|
| **Variáveis Dependentes** | ***Shapiro-Wilk Test*** | | ***Levene Test*** | ***Shapiro-Wilk Test*** | | ***Levene Test*** |
| | ***Ad Hoc*** | ***TestCheck*** | | **Industrial** | **Acadêmico** | |
| **Defeitos** | 0,0591 | 0,2738 | 0,0091 | 0,0962 | 0,0638 | 0.0263 |
| **Falso-Positivo** | 0,0064 | 0,2247 | NT | 0,0302 | 0,0131 | NT |
| **Tempo** | 0,0002 | 0,4990 | NT | 0,0291 | 0,2501 | NT |
| **Eficácia** | 0,0050 | 0,2218 | NT | 0,0069 | 0,2035 | NT |
| **Eficiência** | 0,0013 | 0,6066 | NT | 0,0222 | 0,8144 | NT |

- **Teste Paramétrico (ANOVA) aplicado à Variável DEFEITOS**

Constatado que a variável Defeitos é normal e homocedástica, é necessário aplicar Teste Paramétrico (ANOVA). Foi aplicada com as mesmas condições do primeiro estudo. Analisando o resultado**,** foi possível constatar que *TestCheck*, estatisticamente, detecta mais defeitos quando comparada à *Ad Hoc. A*firmação esta que pode ser comprovada analisando o *p-value* obtido para esta variável (lado esquerdo da Figura 24), que ficou abaixo de 0,05 (*p-value* = 0,0099). Analisando os resultados obtidos por projetos de software (lado direito da Figura 24), percebeu-se que o projeto não foi um fator que influenciou na detecção de defeitos durante a inspeção, pois o valor do *p-value* ficou acima de 0,05 (*p-value* = 0,6313) e os círculos que representam os projetos possuem interseção entre si, indicando que não existe diferente estatística significativa entre eles.

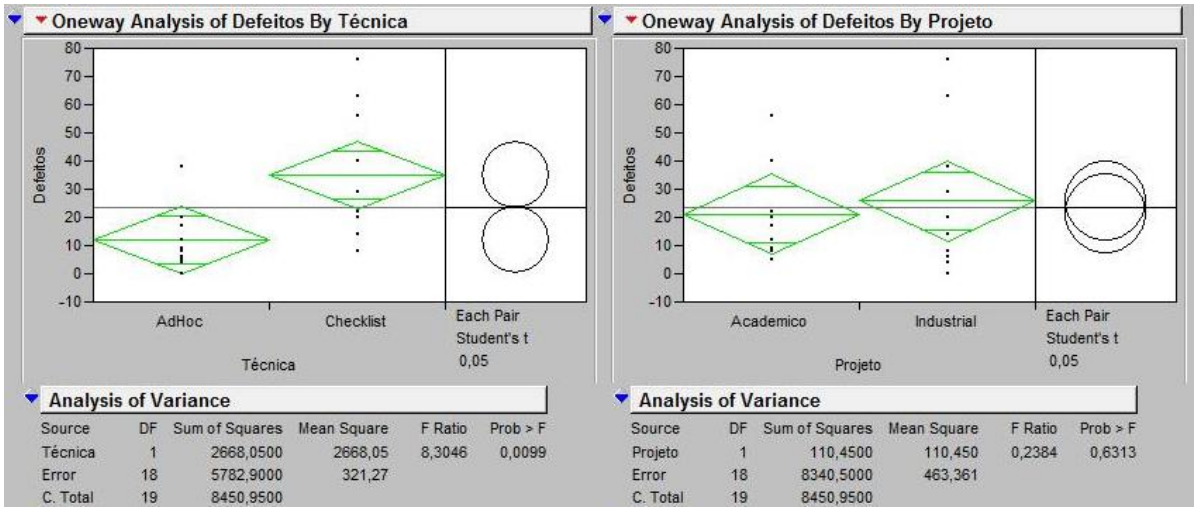



**Figura 24. ANOVA para a variável Defeitos.**

- **Teste Não Paramétrico (*Wilcoxon/Kruskal Wallis Test*) para as variáveis Falso Positivo, Tempo, Eficácia e Eficiência.**

Após aplicar o teste de normalidade, ficou constatado que as variáveis Falso Positivos, Tempo, Eficácia e Eficiência não possuem distribuição normal (Tabela 19). Portanto, foi aplicado teste não paramétrico para análise destas variáveis, cujo resultado está representado na Tabela 20.

Analisando o resultado para as variáveis Falso Positivos, Eficácia e Eficiência, percebeu-se que para este estudo não existe diferença estatística entre as técnicas de inspeção, podendo ser comprovada pelo valor do *p-value* que ficou acima de 0,05 para cada uma delas. Analisando a variável Tempo, percebeu-se que existe diferença estatística sendo comprovada pelo valor do *p-value* que ficou abaixo de 0,05. Portanto, a técnica *Ad Hoc* neste estudo demanda menos tempo para execução do processo de inspeção, o que já era esperado, conforme descrito durante a análise das médias obtidas para esta variável (Seção 5.3.3).

Analisando os projetos de software, observou-se que estes não influenciaram no resultado deste estudo, ficando comprovado através do *p-value* que ficou acima de 0,05 para todas as variáveis analisadas (Tabela 20).

**Tabela 20. Teste não paramétrico para as variáveis Falso Positivos, Tempo, Eficácia e Eficiência.**

| Variáveis Dependentes | *Wilcoxon/Kruskal Wallis Test* | |
|---|---|---|
| | **Por Técnica** | **Por Projeto** |
| **Falso Positivo** | *p-value* = 0,2067 | *p-value* = 0,2358 |
| **Tempo** | *p-value* = 0,0019 | *p-value* = 0,8796 |
| **Eficácia** | *p-value* = 0,6514 | *p-value* = 0,2507 |
| **Eficiência** | *p-value* = 0,0790 | *p-value* = 0,8702 |

Dessa forma, observou-se que a técnica *TestCheck* foi, neste estudo, o fator que influenciou estatisticamente no alto número de defeitos detectados nos artefatos de teste de software e no tempo para realização da inspeção, enquanto que as demais variáveis não tiveram diferenças estatísticas.

Em seguida, assim, como no primeiro estudo de viabilidade, foi realizada uma análise a respeito da cobertura dos defeitos encontrados por *TestCheck*, descrita na próxima subseção.

### 5.3.5 Análise da Cobertura de defeitos

A análise de cobertura dos defeitos detectados por cada técnica no segundo estudo de viabilidade se deu da mesma forma que no primeiro estudo. Defeitos duplicados foram

removidos. Em seguida, separou-se defeitos detectados por cada ou ambas as técnicas (Figura 25).

Novamente observou-se que *TestCheck* cobre a maior parte dos defeitos detectados pela técnica *Ad Hoc*. Analisando o projeto Industrial, percebeu-se que apenas 2 defeitos foram detectados apenas utilizando a técnica *ad-hoc,* no entanto para o projeto Acadêmico notou-se que todos os defeitos detectados pela técnica *ad-hoc* também foram detectados por *TestCheck*. O resultado deste estudo confirma as evidencias observadas no primeiro estudo, ou seja, *TestCheck* contribui para detectar defeitos novos e não triviais que não são detectados por uma técnica A*d Hoc* (BRITO e DIAS-NETO, 2012).

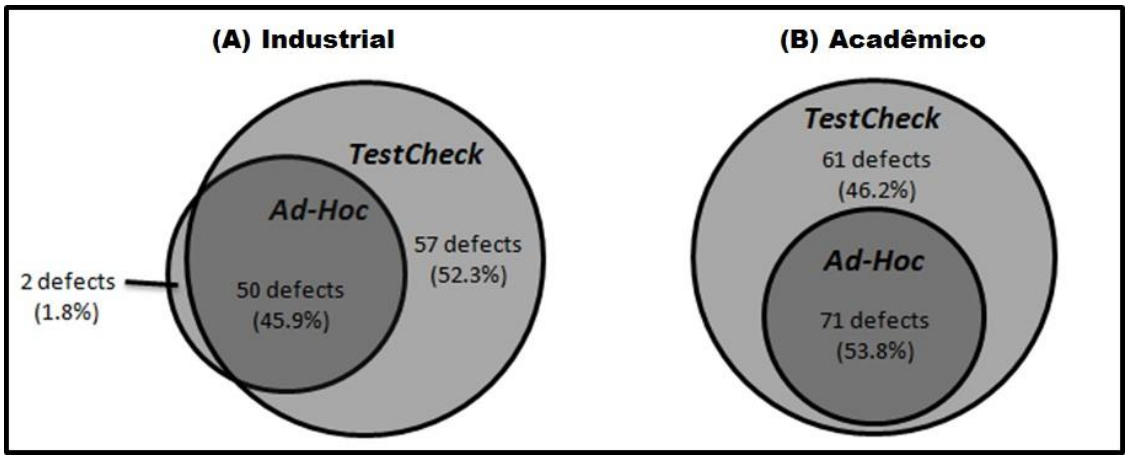



**Figura 25. Análise da Interseção de defeitos detectados pelas técnicas (BRITO e DIAS-NETO, 2012).**

A Figura 26 apresenta a distribuição dos tipos de defeitos detectados por *TestCheck* no estudo de viabilidade 2. Observou-se que novamente a distribuição dos tipos de defeito se assemelhou a resultados obtidos em outros estudos realizados, como (Davis, 1990; Kalinowski et al., 2007), conforme citado anteriormente.

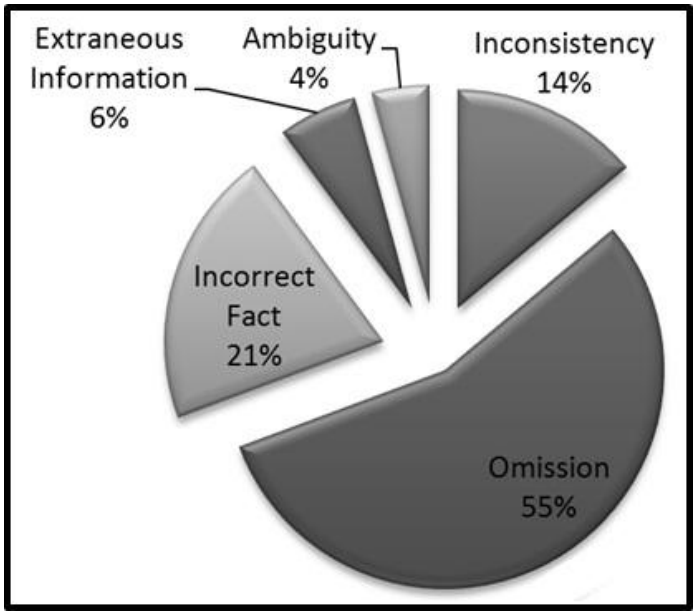



**Figura 26. Tipos de Defeitos detectados por *TestCheck* (BRITO e DIAS-NETO, 2012)**

## 5.4 Considerações Finais

Este segundo estudo de viabilidade realizado faz parte da metodologia proposta por SHULL (2001) adotada nesta pesquisa. Ele teve como objetivo diminuir a taxa de falso-positivos em relação ao resultado obtido no primeiro estudo, além de manter a alta taxa de detecção de defeitos. Para isso, foi utilizada a segunda versão dos *Checklists* que compõe a técnica. Este estudo serviu também para o seu amadurecimento, pois os resultados indicaram novos pontos de melhorias para uma próxima evolução da técnica.

O próximo capítulo trata do segundo estudo de viabilidade aplicado na indústria, correspondente à terceira fase da metodologia adotada. O objetivo do estudo foi avaliar se a abordagem é adequada dentro de um contexto industrial. Para isto, profissionais da indústria atuaram como inspetores de artefatos de teste e deram seu parecer.

# 6. ESTUDO DE OBSERVAÇÃO NA INDÚSTRIA

*Este capítulo descreve a segunda etapa da metodologia adotada, que consiste em um estudo de observação, ocorrido na indústria que objetiva analisar se a abordagem proposta se adequa ao ciclo de desenvolvimento de Software em um contexto industrial.*

## 6.2 Introdução

Como parte da metodologia adotada no estudo (SHULL et al., 2001) adotada nesta pesquisa, foram realizados e descritos nos capítulos anteriores o estudo de viabilidade e o estudo de observação. Estes estudos avaliaram a abordagem proposta e permitiram que modificações e melhorias fossem realizadas. Estes estudos foram realizados em ambiente acadêmico com estudantes de graduação e pós-graduação da UFAM, caracterizado como *in vitro*, ou seja, refere-se à experimentação no laboratório sob condições controladas (TRAVASSOS et al., 2002).

A segunda fase da metodologia adotada sugere a aplicação da nova técnica objetivando analisar seu amadurecimento. Tendo em vista que os resultados adquiridos no primeiro e segundo experimentos apresentaram indícios da viabilidade de *TestCheck*, esta fase foi realizada em um instituto de pesquisa da região norte e têm como finalidade avaliar se a abordagem é adequada/viável para ser utilizada em um ambiente real. Foi analisada sua capacidade de detectar defeitos (avaliação quantitativa) e a satisfação por parte dos inspetores da indústria que utilizaram abordagem (avaliação qualitativa).

Portanto, este capítulo apresentará o estudo de observação realizado, seu propósito, planejamento, execução, assim como os resultados obtidos.

## 6.3 Definição do Estudo de Observação na Indústria

### 6.3.1 Propósito

O propósito deste estudo é responder à questão: “A abordagem *TestCheck* é adequada no contexto industrial para avaliação de artefatos de teste?”. A partir deste estudo, foi observado como *TestCheck* se comporta dentro de um processo Industrial sendo utilizada por profissionais em projetos reais.

### 6.3.2 Perspectiva

A perspectiva é do ponto de vista dos pesquisadores, que desejam analisar principalmente a viabilidade e satisfação de inspetores ao utilizar a abordagem *TestCheck*.

### 6.3.3 Objetivos Específicos

- **Analisar** a aplicação, viabilidade e satisfação da abordagem *TestCheck*.
- **Com o propósito de** avaliar.
- **Em relação** adequação em um contexto real.
- **Do ponto de vista** dos profissionais da indústria.
- **No contexto de** projetos de software.

### 6.3.4 Caracterização do Projeto de Software Inspecionado

Os artefatos de teste inspecionados são provenientes de um projeto para desenvolvimento de uma aplicação web que gerencia campanhas de publicidade, propagandas e os clientes responsáveis pelas campanhas. O objetivo do software é fornecer anúncios (*advertisements* – ADS) com base na localização do usuário e preferências inicialmente definidas a todos os usuários.

## 6.4 Planejamento do Estudo de Observação

Para executar um estudo de observação na indústria, é necessário que a indústria esteja disposta a investir tempo e dinheiro, tendo em vista que os inspetores são membros da organização e farão a inspeção como parte de suas atividades diárias. No entanto, algumas modificações foram realizadas neste estudo, para que este se adequasse as exigências da indústria. Uma restrição foi em relação ao tempo. Assim, as inspeções foram realizadas com um número reduzido de artefatos, de forma que acontecesse em um único período do dia (4 horas de trabalho) de forma a evitar que os profissionais prejudicassem o funcionamento da empresa com horas dedicadas a outras atividades. Mesmo sendo relacionada às suas atividades do dia-a-dia, esta não era uma atividade que fazia parte do cronograma da empresa.

Os participantes do estudo são provenientes de um instituo de pesquisa da região norte, profissionais com diferente nível de experiência que atuam como membros da equipe de teste em projetos de software. A

Tabela 21 apresenta informações sobre os participantes. Neste estudo, um dos participantes é o autor dos documentos inspecionados, e este participará da seção de inspeção para que seja observado seu comportamento em relação aos demais inspetores.

**Tabela 21. Caracterização dos participantes do estudo de observação.**

| Inspetor | Cargo Exercido | Tempo de Experiência na Indústria | Experiência em Inspeção de Software | Experiência em técnica de inspeção | Experiência em Inspeção de Teste |
|---|---|---|---|---|---|
| 01 | Analista de Testes | 2 anos | Média | Baixa | Média |
| 02 | Analista de Teste | Acima de 4 anos | Média | Média | Alta |
| 03 | Estagiário em testes | 6 meses | Baixa | Baixa | Baixa |
| 04 | Gerente de Teste | Acima de 4 anos | Média | Baixa | Média |

Para participar do estudo, os profissionais tiveram que:

- Manifestar interesse em participar do estudo, assinando o Formulário de Consentimento e Formulário de Caracterização, para termos conhecimento do grau de experiência de cada participante;
- Participar do questionário “Pós-Experimento” (Apêndice E) e do treinamento que foi conduzido como uma explicação dos *Checklists* que compõem a abordagem *TestCheck* e sua lista de discrepâncias.
- Serem profissionais que atuem na área.

## 6.5 Execução do Estudo

Este estudo de observação propõe responder uma questão levantada: A abordagem empregada é adequada às necessidades da indústria? Para responder esta questão, foram analisadas as opiniões dos participantes da inspeção. Portanto, será avaliado se a abordagem *TestCheck* possibilita aos inspetores identificar defeitos em artefatos de teste (plano de teste, casos de teste e procedimento de teste) em um tempo viável, evidenciando a eficácia/eficiência dos *Checklists* que compõe a abordagem.

Em se tratando de profissionais da indústria, que possuem restrição de tempo, o processo de inspeção foi modificado. O estudo de observação na indústria foi conduzido em apenas uma sessão e todos os participantes realizaram a inspeção ao mesmo tempo e no mesmo ambiente. Nessa sessão, os candidatos responderam ao Formulário de Caracterização visando identificar a competência e a experiência relacionadas ao propósito do estudo, além de buscar identificar o conhecimento do participante no domínio do problema ao qual o artefato inspecionado pertence. Após a caracterização, os participantes foram orientados a preencher o Formulário de Consentimento.

Os artefatos de teste inspecionados (plano, casos e procedimentos de teste) consistem em documentos de um projeto real da própria organização e que já haviam sido avaliados através de uma abordagem própria (*Ad Hoc*), porém sem registro do seu desempenho (ex.: tempo e número de defeitos). Após todo procedimento da primeira etapa, os inspetores receberam um treinamento em relação à abordagem *TestCheck* e a maneira que deveriam proceder durante a atividade. Assim, deu-se inicio a execução da inspeção com o preenchimento da lista de discrepâncias.

Após a conclusão da inspeção, foi pedido a cada participante que preenchesse o Questionário de Avaliação Pós-Experimento, de forma a obter a sua avaliação qualitativa a respeito da aplicação da abordagem *TestCheck*.

De posse dos Relatórios de Discrepância gerados por cada participante, foi realizada uma única reunião de discriminação para classificação das discrepâncias em defeitos ou falso-positivo. Para qualificar uma discrepância como defeito ou falso positivo, contou-se com o apoio dos próprios participantes e de especialistas em teste de software (como ocorre em qualquer processo de inspeção tradicional).

## 6.6 Validade dos Resultados

Apesar da identificação de possíveis ameaças à validade dos resultados, não foram extraídas conclusões além das limitações impostas por estas ameaças, o que não invalida os resultados do estudo.

### 6.6.1 Validade de Conclusão

- **Confiança das Medidas:** o objetivo que procurou-se alcançar com esse estudo foi identificar o número de defeitos reais e que tipo de defeito a abordagem *TestCheck* identificou com maior intensidade. Sendo assim, uma potencial ameaça à validade de conclusão do estudo se refere à classificação de cada discrepância em defeitos por parte dos experimentalistas. Entretanto, esse processo é subjetivo, o que representa uma ameaça.

### 6.6.2 Validade Interna

- **Participantes:** um problema em potencial está relacionado à amostra dos participantes. No entanto, por se tratar de profissionais da área que atuam na indústria, assume-se que os mesmos compõe uma amostra representativa dos inspetores, com participantes de diferentes níveis de conhecimento.
- **Competência:** Uma possível ameaça à validade é a competência necessária para realização da atividade de inspeção. No entanto, por se tratar de

profissionais da área com habilidades e competência descritas no formulário de caracterização, esta ameaça seria minimizada.

### 6.6.3 Validade de Construção

- **Viés mono-operação:** o estudo de observação na indústria irá utilizar projetos reais, contendo artefatos diferentes: um único Plano de Teste, cinco Casos de Teste e cinco Procedimento de Teste como objeto de inspeção. Portanto, por se tratar de artefatos desenvolvidos na indústria, assume-se que é representativo da teoria sob a qual se sustenta, portanto o resultado apresentado será decorrente da aplicação de *TestCheck* e não da "facilidade" de detecção de defeitos no documento utilizado, visto que tais artefatos já haviam sido inspecionados anteriormente utilizando uma técnica *Ad Hoc* adotada pela organização.

## 6.7 Resultados do Estudo Observação na Indústria

### 6.7.1 Análise Quantitativa

Analisando os resultados do estudo (Tabela 22), percebeu-se que o inspetor 04 não detectou nenhum defeito nos artefatos inspecionados. Uma possível explicação para tal cenário seja o fato de que ele seria o autor dos documentos revisados, o profissional mais experiente do grupo. Este foi incluído entre os inspetores com o objetivo de avaliar seu comportamento, analisando se os *Checklists* ajudariam a identificar elementos importantes que poderiam conter nos artefatos. No entanto, este comportamento não foi constatado, e o autor do documento acreditou que os artefatos estavam corretos de acordo com seu ponto de vista. Outra possível explicação para o baixo desempenho deste inspetor foi o tempo destinado à inspeção (10 minutos), conforme será ainda explicado nesta seção.

Apesar de o autor do documento não ter reportado qualquer discrepância durante a fase de detecção de defeitos, na fase de discriminação, onde as discrepâncias foram classificadas em defeitos ou falso positivo, este concordou com a classificação dos demais inspetores e especialistas em teste a respeito dos defeitos indicados.

**Tabela 22. Resultado do estudo de observação realizado na Indústria.**

| Inspetor | Discrepâncias | Falso-Positivos | Defeitos | Tempo (minutos) | Eficácia (def / dis) | Eficiência |
|---|---|---|---|---|---|---|
| 01 | 8 | 1 | 7 | 85 | 87% | 0,08 |
| 02 | 8 | 0 | 8 | 90 | 100% | 0,09 |
| 03 | 12 | 0 | 12 | 80 | 100% | 0,15 |
| 04 | Inspetor Removido | | | | | |
| Total | 28 | 1 | 27 | 265 | 96% | 0,10 |

Por se tratar de artefatos de um projeto real já inspecionado previamente, o número de defeitos detectados foi considerado satisfatório pelos próprios membros da

organização. Todos os inspetores (com exceção do autor do documento) detectaram um grande volume de defeitos, considerando que o projeto já havia sido revisado pela própria empresa e já havia sido considerado apto a ser executado. Outro ponto a ser discutido observando a Tabela 22 foi a baixa quantidade de falso-positivos gerados pela utilização da abordagem proposta. Este resultado pode ser atribuído a vários fatores, inclusive a evolução dos *Checklists* após os dois primeiros estudos. Outro fator a ser considerado é a experiência prévia de parte dos inspetores em inspeção de software.

Analisando a utilização da abordagem *TestCheck* neste estudo, considerou-se que os resultados foram positivos. Além da alta taxa de detecção de defeitos, percebeu-se que os defeitos encontrados eram de alta severidade, relacionados a funcionalidades importantes do software, o que afetaria a qualidade dos testes. Em relação à eficiência e eficácia, constatou-se que a abordagem *TestCheck* resultou em 96% de eficácia (defeitos por discrepâncias) na detecção de defeitos e eficiência de 0,10 defeitos por minuto.

### 6.7.2 Análise Qualitativa

Analisando os resultados obtidos e a opinião de cada inspetor (Tabela 23), é possível identificar a possibilidade de aplicação para abordagem no contexto industrial. Algumas modificações seriam necessárias, e com base do questionário de avaliação pós-experimento é possível melhorar a abordagem para futuros experimentos.

**Tabela 23. Avaliação realizada por cada Inspetor**

| Questionamentos | Inspetores | | | |
|---|---|---|---|---|
| | **01** | **02** | **03** | **04** |
| Grau de Dificuldade da aplicação da abordagem. | Muito Fácil | Muito Fácil | Fácil | Fácil |
| De que maneira a técnica auxiliou a identificar defeitos nos artefatos? | 3 | 3 | 3 | 2 |
| A abordagem é capaz de avaliar é propor melhorias, mudanças, apontadas onde é possível encontra os erros? | Sim | Sim | Sim | Não |
| A abordagem é capaz de avaliar a qualidade de um artefato de teste? | Sim | Sim | Sim | Não |
| Existe Algum item de avaliação que não foi compreendido? | Não | Não | Sim | Sim |
| Você utilizaria em inspeções futuras? | Sim | Sim | Sim | Não |

**Legenda**

1. Negativamente. A abordagem atuou como um obstáculo. O meu desempenho teria sido melhor se eu não tivesse utilizado.
2. Neutro. Acho que encontraria os mesmos defeitos caso não tivesse utilizado a abordagem.
3. Positivamente. A abordagem auxiliou na detecção de defeitos. Talvez não tivesse detectado alguns defeitos caso não tivesse utilizado.

Analisando as respostas aos questionamentos, chegou-se à seguinte conclusão: em relação à dificuldade na utilização de *TestCheck*, as opiniões variaram de *Fácil* a *Muito*

*Fácil*, o que indicaria que a abordagem não apresentou grandes dificuldade de utilização. Em relação à detecção de defeitos, três inspetores (dentre os quatro que participaram) acreditam que a abordagem ajudou positivamente, pois foi possível detectar defeitos tendo o documento já sido revisado. Outro ponto positivo destacado pelos inspetores é o fato da abordagem possibilitar melhorias/mudanças nos artefatos, além de permitir a avaliação da qualidade do artefato de teste. Por conta desses benefícios, os inspetores (01,02 e 03) acreditam que seja possível utilizar a abordagem em um contexto industrial, atentando para pontos de melhorias descritos na Tabela 24.

A melhoria sugerida pelos inspetores 01 e 03 refere-se à criação de *Checklists* direcionados a empresas que utilizem metodologias ágeis. No entanto, nesta pesquisa procurou-se desenvolver uma abordagem que pudesse ser aplicada a diferentes contextos, independentemente da metodologia de desenvolvimento utilizada por uma organização. Assim, a abordagem *TestCheck* seria adaptável, tendo em vista que os itens do *Checklists* são abrangentes, o que possibilita as empresas escolher os itens que se adequam ao seu contexto da empresa.

**Tabela 24. Sugestões de melhoria dos inspetores**

| Inspetor | Sugestões de Melhoria |
|---|---|
| 01 | Existem alguns pontos onde documentos utilizados no framework Scrum não podem ser aplicados. Acho que desenvolver um *Checklist* somente para metodologias ágeis seria bastante interessante. |
| 02 | Talvez um glossário com a explicação de alguns itens fosse interessante para profissionais iniciantes na área. |
| 03 | Direcionar o *Checklist* para empresas que adotam metodologias ágeis. |
| 04 | Customizar levando em consideração a metodologia da empresa. |

Todas as sugestões foram levadas em consideração e analisadas, pois a pesquisa foca na adequação da abordagem ao contexto industrial. A sugestão feita pelo Inspetor 02 foi considerada muito pertinente. Assim, para a nova versão de *TestCheck* será incluído um glossário de termos. Em relação à customização, sugestão proposta pelo Inspetor 04, os *Checklists* pertencentes a abordagem podem ser customizados. Porém, isso precisaria ser feito manualmente, o que seria custoso em um projeto. Uma possível melhoria seria implementar a abordagem *TestCheck* em um apoio ferramental de forma a possibilitar a sua customização minimizando o esforço desta tarefa.

### 6.7.3 Comportamento do autor do documento durante a Inspeção

Sabe-se que não é uma boa prática que o autor do artefato participe do processo de inspeção. No entanto, com o objetivo de analisar seu comportamento este foi convidado e aceitou participar do estudo.

Analisando os dados obtidos por este inspetor pode-se concluir que o mesmo não estava interessado em participar do estudo como inspetor, mas sim como autor do documento. Esta informação pode ser confirmada através dos seguintes dados: (a) O tempo despendido para realização desta tarefa foi de apenas 10 minutos; (b) as informações colocadas na Tabela 22, na qual o autor afirma que a abordagem não o ajudou a encontrar nenhum defeito, pois este afirma que encontraria os mesmos defeitos sem o auxílio do Checklist, o que é contraditório, pois o mesmo não encontrou nenhum defeito.

Portanto, percebe-se que não é uma boa prática que o autor inspecione seu próprio documento, evitando assim, que resultados de experimentos sejam comprometidos por conta de um único inspetor.

## 6.8 Considerações Finais

Este capítulo abordou o estudo de observação realizado em um instituto de pesquisa da região norte com objetivo de averiguar se a abordagem proposta se adequa ao ciclo de desenvolvimento de uma empresa. Foram analisados resultados quantitativos em relação à eficiência e eficácia na detecção de defeitos e resultados qualitativos (opinião dos participantes). O capítulo apresentou também as possíveis melhorias reportadas pelos próprios inspetores em relação à abordagem *TestCheck*, com o intuito de facilitar a utilização desta em empresas de software.

No próximo capítulo serão apresentadas considerações finais e contribuições deste trabalho, assim como as perspectivas de trabalhos futuros para continuação desta pesquisa.

# 7. CONCLUSÕES E TRABALHOS FUTUROS

*Neste capítulo serão apresentadas as conclusões desta dissertação, apresentando as suas contribuições e trabalhos futuros que fornecem a direção para que seja dada continuidade a pesquisa relacionado à abordagem TestCheck que apoia a detecção de defeitos em artefatos de teste, permitindo qualidade nos artefatos desenvolvidos.*

## 7.1 Considerações Finais

A crescente complexidade nos softwares desenvolvidos requerem testes de qualidade, e estes devem ser planejados de tal modo que satisfaçam seu objetivo principal que é revelar falhas. Durante o processo de testes, artefatos são desenvolvidos para em seguida serem executados. Portanto, é necessário que estes artefatos sejam de qualidade.

Ao se planejar testes, é gerado o artefato Plano de Teste que contem informações importantes para a condução dos testes em um projeto de software, e que descreve os casos e procedimentos de teste a serem executados. O caso de teste, por sua vez, é constituído por um conjunto de dados de entrada, condições de execução e resultados esperados. A descrição incorreta de alguma dessas informações pode resultar em testes inadequados. Por fim, o artefato procedimento de teste especifica os passos necessários para execução do teste, e qualquer informação imprecisa pode comprometer os testes executados. Portanto, é necessário que estes artefatos sejam construídos com qualidade.

Nesta pesquisa, não foram encontradas abordagens que avaliassem os artefatos de testes. Assim, este trabalho propôs a abordagem *TestCheck* para apoiar a detecção de defeitos em artefatos de teste de software, visando melhorar a qualidades destes artefatos e como consequência aumentar a qualidade dos testes e do produto desenvolvido. Para analisar a viabilidade e amadurecimento da abordagem proposta, três experimentos foram realizados seguindo a metodologia proposta por SHULL et al. (2001). Os resultados indicam a viabilidade de *TestCheck* para detecção de defeitos em artefatos de teste, sendo esta eficaz e eficiente quando comparada a uma abordagem *Ad Hoc*.

Esta pesquisa possibilitou a publicação de dois artigos em conferências. Um abordava os estudos experimentais realizados para avaliar a abordagem proposta, aceito em *Experimental Software Engineering Latin American Workshop* (ESELAW) e o segundo tratava da abordagem desenvolvida publicado no Simpósio Brasileiro de Qualidade de Software (SBQS).

## 7.2 Contribuições

As principais contribuições oferecidas por esta pesquisa à comunidade de teste de software são:

- Desenvolvimento e disponibilização de uma abordagem baseada em *Checklist* (*TestCheck*) para apoiar a detecção de defeitos em artefatos de teste de software, melhorando a qualidade dos testes.
- Mapeamento de características necessárias em artefatos de teste, que podem funcionar como diretrizes de apoio à elaboração (não apenas avaliação) destes artefatos em um projeto de software.
- Elaboração de um pacote de estudos visando a avaliação de técnicas de inspeção de artefatos de teste, desde sua concepção até o passo inicial visando sua transferência para indústria.

## 7.3 Limitações

Algumas limitações foram observadas no decorrer da pesquisa:

- Os estudos de avaliação da abordagem não são conclusivos, pois foram aplicados em contextos específicos que não permitem a expansão de seus resultados para outros cenários. Novos estudos em outros cenários com diferentes artefatos são necessários.
- Foram desenvolvidos *Checklists* apenas para um subconjunto dos artefatos produzidos ao longo do processo de teste de software. Tal atividade poderia ser expandida para outros artefatos (ex: Laudo de Teste [Relatório de Resumo dos Testes]).
- Na forma atual, a customização dos *Checklists* para aplicação em diferentes cenários é possível, porém custosa para ser realizada manualmente. Um mecanismo de apoio à customização da abordagem poderia ser empregado visando reduzir este esforço.

## 7.4 Trabalhos Futuros

Apesar de não serem conclusivos, os dados obtidos nos estudos indicam que a abordagem é viável para o que se propôs realizar. Assim, como próximos passos estão previstos:

- A continuidade da metodologia proposta em (SHULL et al., 2001) que foi seguida

para avaliação de *TestCheck*, visando a longo prazo sua transferência para ambiente industrial. Estes novos estudos contribuiriam para o amadurecimento da abordagem, introduzindo melhorias dos itens de avaliação e aprimoramento dos questionamentos.

- Implementar a abordagem *TestCheck* em um apoio ferramenta de forma a possibilitar a sua customização para projetos ou organizações específicas. Neste cenário, os profissionais do projeto poderiam escolher os itens que deveriam compor os *Checklists* para ser utilizado. Este passo está em andamento, onde os *Checklists* que compõem *TestCheck* estão sendo integrados à infraestrutura Maraká (DIAS-NETO, 2006), de apoio ao planejamento e controle de teste de software.

# Referências Bibliográficas


ACKERMAN, A., BUCHWALD, L., LEWSKI, F., (1989), “Software Inspections: An Effective Verification Process”, IEEE Software, vol. 6, no. 3, pp.31-37.

AGILEMANIFESTO: (2001) Manifesto for Agile Software Development. Disponível em: http://agilemanifesto.org/, acessado em 20 de abril de 2011.

BAHSOON, R., EMMERICH, W., (2003), "Evaluating Software architectures: development, stability, and evolution". In: Book of Abstracts of the ACS/IEEE International Conference on Computer Systems and Applications, pp. 47, Tunis, Tunisia, Julho.

BARCELOS, R., F., (2006), “Uma abordagem para inspeção de documentos arquiteturais baseada em Checklist”, Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro.

BARCELOS, R.F.; TRAVASSOS, G.H., (2006) “ArqCheck: Uma abordagem para inspeção de documentos arquiteturais baseada em checklist” In: Simpósio Brasileiro de Qualidade de Software (SBQS), PP. 174-188, Vila Velha-ES.

BASILI, G., CALDIERA, F., LANUBILE, F., SHULL., (1996), "Studies on reading techniques". In: Proceedings of the Twenty-First Annual Software Engineering Workshop, Greenbelt, Maryland, Dezembro.

BECK, K., (2003), “Test Drive Development – by example”. 1. ed. Boston: Addison-Wesley, p. 240, 2003.

BECK, K., ANDRES. C. (2005), “Extreme Programming Explained: embrace change”. 2. Ed. Upper Saddle River: Addison-Wesley. 189 p.

BERTINI, L., A., (2006), “Técnicas de inspeção aplicadas a avalição de requisitos de sistemas de Software: Um estudo comparativo”, Dissertação de Mestrado, Universidade Metodista de Piracicaba.

BOEHM, B., BASILI, V., (2001), “Software Defect Reduction Top 10 List,” IEEE Computer, vol. 34(1): pp. 135-137.

BRITO, J., DIAS-NETO, A., C., (2012) “Conduzindo Estudos Experimentais para Avaliação de uma Técnica de Inspeção de Artefatos de Teste de Software” - Experimental Software Engineering Latin American Workshop (ESELAW)

BRITO, J., DIAS-NETO, A., C., (2012) “TestCheck – Uma Abordagem Baseada em Checklist para Inspecionar Artefatos de Teste de Software” – Simposio Brasileiro de Qualidade de Software (SBQS)

BRYKCZYNSKI, B., (1999), “A Survey of Software Inspection Checklists” ACM SIGSOFT pp. 83-89

CLEMENTS, P., KAZMAN, R., KLEIN, M., (2002), Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley.

CHENG, B., JEFFERY, R.,(1996), “Comparing Inspection Strategies for Software Requirement Specifications” The University of New South Wales, Australia - IEEE

CRESPO, A. N., SILVA, O., J., BORGES, C., A., SALVIANO, C., F., ARGOLLO, M., JINO, M., (2004) “Uma metodologia para teste de Software no Contexto da Melhoria de Processo”, In: III Simpósio Brasileiro de Qualidade de Software (SBQS 2004), Brasília.

CONTE, T., U., (2009) “Técnica de Inspeção de Usabilidade Baseada em Perspectiva de Projeto Web”, Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro.

DELAMARO, M.E., MALDONADO, J.C., JINO, M., (2007), “Introdução ao Teste de Software”, Rio de Janeiro: Elsevier

DIAS-NETO, A. C., (2010) “Evolving a Computerized Infrastructure to support the Selection of Model-Based Testing Techniques”, In: IFIP International Conference on Testing Software and Systems (ICTSS'10)

DIAS-NETO, A., C., (2006), “Uma Infra-Estrutura Computacional para Apoiar o Planejamento e Controle de Teste de Software”, Dissertação de Mestrado, COPPE/UFRJ, Rio Janeiro.

DOBRICA, L., NIEMELA, E., (2002), "A survey on Software architecture analysis methods", IEEE Transactions on Software Engineering, v. 28, n. 7, pp. 638-653.

FAGAN M., E., (1976) “Design and Code Inspections to Reduce Erros in Program Development” - IBM Systems Journal. Riverton. NJ. V.15. n.3.p.182-211.

HEDBERG, H., SAKKA J., (2006), “Technical Reviews in Agile Development: Case Mobile-DTM” - International Conference on Quality Software (QSIC'06) IEEE - Department of Information Processing Science, University of Oulu, Finland

HUMPHREY, W.S., (1989) Managing the Software process, Addison-Wesley

IEEE Standard 829-1998: Standard for Software Test Documentation, IEEE Press.

IEEE Standard 829-2008: Standard for Software and System Test Documentation (Revision of IEEE Std 829-1998)

ITOKEN, J., V., MANTYLA, M,V. LASSENIUS, C., (2007), “Defect Detection Efficiency: Test Case Based vs. Exploratory Testing ” Helsinki University of Technology, Software Business and Engineering Institute P.O. BOX 9210, FIN-02015 TKK, Finland

KALINOWSKI, M., 2004, “Infra-Estrutura Computacional de Apoio ao Processo de Inspeção de Software”, Dissertação de Mestrado, Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ, Rio de Janeiro.

KALINOWSKI, M. SPÍNOLA, R., O. DIAS-NETO, A., C. BOTT, A. TRAVASSOS, G., H., (2007), “Inspeção de Requisitos de Software em Desenvolvimento Incremental: Uma experiência prática”, VI Simpósio Brasileiro de Qualidade de Software.

KANER, C., (2006), —Exploratory Testing‖, Florida Institute of Technology, Quality Assurance Institute Worldwide Annual Software Testing Conference, Orlando, FL, Novembro.

KITCHENHAM, B., BRERETON, P., BUDGEN, D., TURNER, M., BAILEY, J., LINKMAN, S. (2009), “Systematic literature reviews in Software engineering”, Inf. Softw. Technol. 51, 1 (Jan. 2009), 7-15.

KOLLANUS, S., KOSKINEN, J., (2009), "Survey of Software Inspection Research" - Department of Computer Science and Information Systems, University of Jyväskylä, Finland - The Open Software Engineering Journal, pp. 15-34

JACOBSEN, I., GRADY B., RUMBAUGH, J., (1999), “The Unified Software Development Process”.

LAITENBERGER, O., DEBAUD, J.M., (1997). Perspective-based Reading of Code documents at Robert *Bosch GmbH. Information and Software Technology, 39:781–791.*

LANUBILE, F., MALLARDO, T., (2007) “Inspecting Automated Test Code: A Preliminary Study” - Department di Informatica, University of Bari, IEEE Standard 829-1998: Standard for Software Test Documentation, IEEE Press.

LUO, L., (2010) “Software Testing Techniques - Technology Maturation and Research Strategies” In: Institute for Software Research International – Carnegie Mellon University Pittsburgh.

MAFRA, S., N., TRAVASSOS, G., H., (2005), "Técnicas de Leitura de Software: Uma revisão Sistemática". In: *Simpósio Brasileiro de Engenharia de Software*, Uberlândia, MG, Brasil, Outubro.

MALDONADO, J. C., (1991), “Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software”, Tese de Doutorado, DCA/FEE/UNICAMP, Campinas, SP, Julho.

PERRY, W., (1995) “Effective Methods for Software Testing”, Wiley.

PFLEEGER. S.L., (2004), “Engenharia de Software – Teoria e Pratica”, Prentice Hall, 2ª Edição. São Paulo.

POON, I. P., TSE, T.H., KUO, F.C., (2010) “Contributions of tester experience and a Checklist guideline to the identification of categories and choices for Software testing” In: Software Quality Journal

PORTO, D. P. (2009) “CRISTA: Um apoio computacional para atividades de inspeção e compreensão de código”. Dissertação de Mestrado em Computação – Universidade Federal de São Carlos, São Carlos.

PRANGE, H.F., (2007) “Uma Avaliação Empírica de um Ambiente Favorável para o Desenvolvimento Dirigido por Testes”, Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro – Rio de Janeiro.

PRESSMAN, R.S., (2006) “Engenharia de Software”. 6º Ed: São Paulo: MCGRAW-Hill.

ROCHA, A. R. C., MALDONADO, J. C., WEBER, K. C., (2001) “Qualidade de Software – Teoria e prática”, Prentice Hall, São Paulo.

RODRÍGUEZ, M., GENERO, M., TORRE, D., BLASCO. B., PIATTINI. M. (2010), “A Methodology for Continuous Quality Assessment of Software Artefacts” In: International Conference on Quality Software.

RYBER, T. (2007), “Essential Software Test Design”. Vol: 2010, Issue: 07/22, Publisher: Fearless Consulting, pp 39 – 45.

SAUER, C., JEFFERY., D.R., LAND, L., YETTON, P., (2000) "The Effecticveness of Software Development Technical Review: A Behaviorally Motivated Program of Research", IEEE Transactions on Software Engineering, 26(1): 1-14, Janeiro

SCHWABER, K. (1995) Scrum Development Process. In: OOPSLA’95 Workshop on Business Object Design and Implementation. Springer-Verlag, Atlanta.

SEI. CMMI para Desenvolvimento (CMMI-DEV, 2006), versão 1.2, CMU/SEI-2006-TR-008. Software Engineering Institute.

SHULL, F., (1998), Developing Techniques for Using Software Documents: A Series of Emprirical Studies, PhD Thesis, Department of Computer Science, university of Maryland, USA, pp 4.

SHULL, F., CARVER, J., TRAVASSOS, G.H., (2001), "An Empirical Methodology for Introducing Software Processes". In: Proceedings of European Software Engineering Conference, pp. 288-296, Setembro.

SHULL, F., RUS, I., BASILI. V, (2000) “How perspective based reading can improve requirements inspections”, IEEE Computer, v. 33, n. 7, pp 73-79.

SILVA, L.F.S., (2004), “Uma abordagem com apoio ferramental para aplicação de técnicas de leitura baseada em perspectiva”, Dissertação de Mestrado, Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ.

SOFTEX. MPS.BR (2011) – Melhoria de Processo do Software Brasileiro - Guia Geral, Guia de Implementação e Guia de Avaliação. 2009. Disponível em: <http://www.softex.br/mpsbr>. Acesso em: 30 de novembro.

SOMMERVILLE, I, (2007), “Engenharia de Software” - 8 Ed.- São Paulo: Pearson Addison-Wesley.

TRAVASSOS, G., T., GUROV, D., AMARAL, E., A., G., (2002), ”Relatório Técnico – Introdução à Engenharia de Software Experimental ” – Programa de Engenharia de Sistemas e Computação COPPE/UFRJ.

VILLELA, K., (2004),¨Definição e Construção de Ambientes de Desenvolvimento de Software Orientados à Organização ¨, Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

# Apêndice A – *Checklists* para inspeção de Artefatos de Teste de Software

A opção N/A (Não se aplica) deve ser marcada se para aquele artefato a ser inspecionado for considerado que não é possível aplicar o item ou algum questionamento pertencente a ele para avaliar o artefato.

## A.1. *Checklist* para Avaliação de Plano de Teste

| Nº | Item de avaliação: Identificador do Plano de Teste | Sim | Não | N/A |
|---|---|---|---|---|
| P01 | Foi definido um identificador para o plano de teste? | | | |
| P02 | O identificador utilizado determina unicamente o plano de testes? | | | |
| P03 | Está sendo identificada a versão atual do plano de teste? | | | |
| **Nº** | **Item de avaliação: Item de Teste** | **Sim** | **Não** | **N/A** |
| P04 | Os itens de teste definidos no plano de teste estão seguindo o mesmo padrão em relação às partes do Software a serem testadas? | | | |
| P05 | Os itens de teste listados no plano de testes realmente deveriam ser testados? Foi percebida ausência de algum item importante? | | | |
| P06 | Foram definidas as referências para os artefatos que descrevem os itens de testes? | | | |
| P07 | Todos os itens de teste são únicos, não havendo repetição ou sobreposição? | | | |
| **Nº** | **Item de avaliação: Características de Qualidade** | **Sim** | **Não** | **N/A** |
| P08 | Todas as características de qualidade selecionadas para os testes estão presentes como requisitos funcionais/ não funcionais do Software? | | | |
| P09 | Há necessidade de teste para todas as características de qualidade listadas? | | | |
| P10 | As características de qualidade que não serão testadas tiveram motivo adequadamente justificado? | | | |
| **Nº** | **Item de avaliação: Tipos de Teste** | **Sim** | **Não** | **N/A** |
| P11 | Os tipos de teste indicados podem ser aplicados nos itens de teste definidos? | | | |
| P12 | Todos os tipos de testes necessários estão descritos no plano de teste? | | | |
| **Nº** | **Item de avaliação: Abordagem de Teste** | **Sim** | **Não** | **N/A** |
| P13 | As abordagens de teste descritas são adequadas aos tipos de teste definidos? | | | |
| P14 | As abordagens de teste foram descritas com detalhes (métodos, técnica, critérios e ferramentas) possibilitando seu uso no projeto dos testes? | | | |
| **Nº** | **Item de avaliação: Critérios para aceitação e rejeição de itens de teste** | **Sim** | **Não** | **N/A** |
| P15 | Os critérios para aceitação definidos cobrem todos os itens de teste? | | | |
| P16 | Os critérios de aceitação dos itens de teste são objetivos e atingíveis? | | | |

| | | | | |
|---|---|---|---|---|
| P17 | Critérios para aceitação com restrição de itens de teste foram definidos? | | | |
| **Nº** | **Item de avaliação: Critérios de suspensão e retomada dos testes** | **Sim** | **Não** | **N/A** |
| P18 | Os critérios para repetição dos testes para novas versões do Software (testes de regressão) foram estabelecidos, indicando quais elementos serão retestados? | | | |
| **Nº** | **Item de avaliação: Tarefas de Testes** | **Sim** | **Não** | **N/A** |
| P19 | Todas as tarefas necessárias para a condução dos testes foram definidas? | | | |
| P20 | Todas as tarefas de teste listadas são realmente necessárias? | | | |
| P21 | As dependências entre as tarefas de testes foram identificadas? | | | |
| P22 | As habilidades necessárias para execução das tarefas foram identificadas? | | | |
| P23 | Os artefatos a serem gerados por cada tarefa do processo foram definidos? | | | |
| **Nº** | **Item de avaliação: Responsabilidades/ Perfil da Equipe** | **Sim** | **Não** | **N/A** |
| P24 | Os responsáveis por cada tarefa do processo de testes foram definidos? | | | |
| P25 | A divisão de responsabilidades entre os membros da equipe de teste é possível de ser implementada, sem causar sobreposição de responsabilidades? | | | |
| P26 | Os responsáveis pela liberação dos itens de teste e pela disponibilidade dos recursos físicos do ambiente de teste foram identificados? | | | |
| **Nº** | **Item de avaliação: Necessidades do Ambiente de Teste** | **Sim** | **Não** | **N/A** |
| P27 | As características físicas das instalações necessárias para a realização dos testes (ex: hardware e configuração de rede) foram definidas? | | | |
| P28 | Os Softwares necessários para a realização dos testes (ex: ferramentas de automação e bancos de dados) foram especificados? | | | |
| **Nº** | **Item de avaliação: Necessidade de treinamento** | **Sim** | **Não** | **N/A** |
| P29 | Foram especificadas as necessidades de treinamento para realização dos testes? | | | |
| P30 | Foram definidos os pré-requisitos necessários para participação no treinamento e os profissionais que participarão? | | | |
| P31 | Os treinamentos foram programados para um período adequado do projeto? | | | |
| **Nº** | **Item de avaliação: Cronograma** | **Sim** | **Não** | **N/A** |
| P32 | As tarefas que serão executadas foram detalhadas estimando o seu tempo de execução e os respectivos responsáveis? | | | |
| P33 | As datas estipuladas para as tarefas estão de acordo com a sua complexidade? | | | |
| **Nº** | **Item de avaliação: Riscos e contingências** | **Sim** | **Não** | **N/A** |
| P34 | Todos os riscos reportados se aplicam ao projeto de Software em questão? | | | |
| P35 | Foi indicado o impacto no projeto caso cada risco se concretize? | | | |
| P36 | Foi indicada a probabilidade de ocorrência de cada risco? | | | |
| P37 | Foi descrito o plano de contingência para cada possível risco identificado? | | | |
| P38 | Foi descrito o plano de mitigação para cada possível risco identificado? | | | |

## A.2. *Checklist* para Avaliação de Caso de Teste

| Nº | Item de avaliação: Identificador do Caso de Teste | Sim | Não | N/A |
|---|---|---|---|---|
| C01 | Foi definido um identificador para o caso de teste? | | | |
| C02 | O identificador utilizado determina unicamente o caso de testes? | | | |
| C03 | Foi identificada a versão atual do documento? | | | |
| **Nº** | **Item de avaliação: Item de Teste** | | | |
| C04 | Foram identificados os itens de teste associados a este caso de teste? | | | |
| C05 | Os itens de teste associados a este caso de teste foram especificados no plano de teste? | | | |
| C06 | Os itens de teste estão associados corretamente a este caso de teste, sem a existência de item de teste associado incorretamente? | | | |
| **Nº** | **Item de avaliação: Especificações de Entrada** | | | |
| C07 | Todas as entradas necessárias para executar este caso de teste foram especificadas? | | | |
| C08 | Todas as entradas especificadas são de fato importantes e necessárias para execução dos testes? | | | |
| C09 | Foram definidos os tipos de dados e valores (ou intervalo ou categoria de valores) associados a cada entrada? | | | |
| C10 | As pré-condições ou pré-requisitos (se necessário) para execução do caso de teste foram especificados? | | | |
| **Nº** | **Item de avaliação: Especificações de Saída /Resultados esperados** | | | |
| C11 | Todas as saídas e resultados esperados após a execução do caso de teste foram determinados? | | | |
| C12 | Todos os comportamentos esperados do Software (ex: tempo de resposta ou estado interno) foram descritos? | | | |
| C13 | Todos os resultados e comportamentos esperados após a execução do caso de teste são de fato necessários e viáveis de serem obtidos? | | | |
| C14 | Todos os resultados e comportamentos esperados após a execução do caso de teste estão descritos de forma legível, com suas unidades, de forma a evitar erros de entendimento? | | | |
| **Nº** | **Item de avaliação: Necessidades do Ambiente de Teste** | | | |
| C15 | As características físicas das instalações necessárias para a execução do caso de teste (ex: hardware e configuração de rede) foram definidas? | | | |
| C16 | Os Softwares necessários para a execução deste caso de teste (ex: ferramentas de automação e bancos de dados) foram especificados? | | | |
| **Nº** | **Item de avaliação: Dependência entre casos de teste** | | | |
| C17 | As dependências em relação a outros casos de testes foram definidas e justificadas? | | | |
| C18 | As dependências especificadas em relação a outros casos de testes realmente existem? | | | |
| C19 | Os outros casos de testes- com quem o caso de teste que está sendo avaliado possui dependência - existem e podem ser recuperados a partir de seus identificadores? | | | |

## A.3. *Checklist* para Avaliação de Procedimento de Teste

| Nº | Item de avaliação: Identificador do Procedimento de Teste | Sim | Não | N/A |
|---|---|---|---|---|
| P01 | Foi definido um identificador para o procedimento de teste? | | | |
| P02 | O identificador utilizado determina unicamente o procedimento de testes? | | | |
| P03 | Foi identificada a versão atual do procedimento de teste? | | | |
| **Nº** | **Item de avaliação: Propósito do Procedimento de Teste** | | | |
| P04 | O propósito/objetivo do procedimento de teste está descrito claramente? | | | |
| P05 | Os casos de teste associados a este procedimento de teste estão listados? | | | |
| **Nº** | **Item de avaliação: Requisitos Especiais** | | | |
| P06 | Há uma indicação da forma de execução do procedimento de teste (ex: manual ou automatizado)? | | | |
| P07 | Os pré-requisitos (se necessário) para execução do procedimento de teste foram especificados? | | | |
| P08 | O ambiente no qual o procedimento de teste será executado está descrito adequadamente (ex: sistema operacional, ferramentas, bancos de dados, sistemas externos)? | | | |
| P09 | As habilidades ou treinamentos necessários para execução do procedimento de teste (especialmente se este for automatizado) estão descritas? | | | |
| **Nº** | **Item de avaliação: Passos do Procedimento** | | | |
| P10 | O passo inicial para a execução do procedimento de teste está descrito? | | | |
| P11 | O passo final para a conclusão do procedimento de teste está descrito? | | | |
| P12 | A ordem dos passos possui uma sequência lógica para execução do teste? | | | |
| P13 | Algum passo que possa prejudicar a execução do teste foi omitido? | | | |
| P14 | Todos os passos estão descritos de forma clara? | | | |
| P15 | Os casos de teste associados a cada passo (se necessário) foram definidos? | | | |
| P16 | Há passos desnecessários descritos no procedimento de teste? | | | |

# Apêndice B – Formulário de Consentimento

## "Experimentação em Teste de Software"

Eu declaro ter mais de 18 anos de idade e que concordo em participar em estudos não invasivos conduzidos pelo Prof. Arilo Cláudio Dias Neto e a pesquisadora Jardelane de Brito Costa, como parte das atividades dos cursos de graduação e pós-graduação do IComp da UFAM/AM. Estes estudos visam compreender a viabilidade de aplicação de técnicas de inspeção em Artefatos de Teste, tentando entender sob que circunstâncias estas técnicas podem ser efetivamente utilizada.

**PROCEDIMENTO**

Conceitos sobre qualidade, técnicas de inspeção e Teste de Software serão apresentados. Eu entendo que serei ensinado como as tecnologias podem ser aplicadas e serei solicitado a aplicá-las no projeto distribuído durante o curso. Neste projeto, técnica de inspeção usando checklist será aplicada por mim visando permitir pensar sobre seu uso e avaliá-lo. Eu entendo que, uma vez o curso terminado, o trabalho que desenvolvi, será estudado visando entender a eficiência e eficácia da técnica que me foi ensinada.

Eu entendo que esta atividade preenche parte dos requisitos do curso e serão avaliados como tal. O pesquisador conduzirá o estudo consistindo da coleta, análise e relato dos dados da atividade de inspeção. Eu entendo que não tenho obrigação alguma em contribuir com informação sobre meu desempenho na atividade, e que posso solicitar a retirada de meus resultados do experimento a qualquer momento e sem qualquer penalidade ou prejuízo. Eu entendo que não existirá nenhum crédito ou benefício extra por participar deste estudo, e que não haverá qualquer impacto negativo em minha avaliação por não participar do estudo. Eu entendo também que quando os dados forem coletados e analisados, meu nome será removido dos dados e que este não será utilizado em nenhum momento durante a análise ou quando os resultados forem apresentados.

**CONFIDENCIALIDADE**

Toda informação coletada neste estudo é confidencial, e meu nome não será identificado em momento algum. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo da técnica e documentos apresentados e que fazem parte do experimento.

**BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA.**

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado visando atender os requisitos do curso, independente de participar ou não deste estudo, mas que os pesquisadores esperam aprender mais sobre quão eficiente é a utilização de tecnologias de software e os benefícios trazidos por este estudo para o contexto da Engenharia de Software.

Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada à minha pessoa não seja incluída no estudo. Eu entendo que minha participação no estudo não afetará minha nota final de qualquer forma, e que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas e processos para a Engenharia de Software.

PROFESSOR RESPONSÁVEL

Prof. Arilo Claudio Dias Neto

Programa de Pós-graduação em Informática

UFAM

Nome (em letra de forma): _____

Assinatura: _____ Data:_____

# Apêndice C - Formulário de Caracterização

**Nome:** _____ **Curso:** _____

**Formação Geral**

Qual é sua experiência anterior em inspeção de software na prática? (marque aqueles itens que melhor se aplicam)

__ nunca inspecionei artefatos de software.

__ tenho inspecionado artefatos de software para conhecimento próprio.

__ tenho inspecionado artefatos de software como parte de uma equipe, relacionado a um curso.

__ tenho inspecionado artefatos de software como parte de uma equipe, na indústria.

Por favor, explique sua resposta. Inclua o número de semestres ou número de anos de experiência relevante em inspeção de software: _____

**Experiência em Inspeção de Artefatos de Software**

**Por favor, indique o grau de sua experiência nesta seção seguindo a escala de 5 pontos abaixo:**

1 = nenhum

2 = estudei em aula ou em livro

3 = pratiquei em 1 projeto em sala de aula

4 = usei em 1 projeto na indústria

5 = usei em vários projetos na indústria

**Experiência com elaboração de artefatos de software**

- Experiência escrevendo artefatos de software seguindo algum *template* 1 2 3 4 5
- Experiência escrevendo artefatos de software 1 2 3 4 5

**Experiência em Teste de Software**

- Experiência em planejar testes 1 2 3 4 5
- Experiência em projetar casos e procedimentos de testes 1 2 3 4 5
- Experiência em documentar atividades de teste 1 2 3 4 5
- Experiência no padrão IEEE 829 (para documentação de testes) 1 2 3 4 5

**Experiência em Inspeção artefatos**

- Experiência em revisão de artefatos de software? 1 2 3 4 5
- Experiência em inspeção de artefatos de software 1 2 3 4 5
- Experiência em inspecionar artefatos usando checklist 1 2 3 4 5
- Experiência em inspecionar artefatos de testes 1 2 3 4 5

# Apêndice D - Lista de Discrepância usada na Inspeção *Ad Hoc*

**Relatório de Discrepância – Inspeção *Ad Hoc***

***Projeto <<preencher nome do projeto>>***

**Revisor:** _____

**Tempo de inspeção (em minutos):** _____

**Início da Inspeção:** ____/____/____

**Data de Término da Inspeção:** ____/____/____

Observações:

- Nº discrepância: numerar de 1 até N, onde N é a quantidade de discrepâncias encontradas durante a revisão.
- Nº página: indicar o número da página onde a discrepância foi localizada.
- Artefato de Teste: indicar qual artefato será inspecionado (Plano de Teste, Procedimento de Teste e Caso de Teste)
- Tipo Defeito: indicar qual o tipo do defeito supostamente encontrado. São eles: omissão, fato incorreto, informação estranha, ambiguidade, inconsistência ou outro tipo (que recomando só usar quando DE FATO for algum outro tipo de defeito).
- Descrição: detalhamento do defeito supostamente identificado (sem pensar na solução, mas sim focando no problema encontrado).

Taxonomia de Defeitos

| Tipo Defeito | Descrição |
|---|---|
| Omissão | Quando uma determinada informação importante para execução do teste não foi incluído no artefato. |
| Fato Incorreto | Quando uma informação apresentada no artefato de teste não foi colocada de maneira correta. |
| Informação Estranha | Quando é apresentado informações desnecessárias. |
| Ambiguidade | As informações apresentadas no artefato de teste, da margem para dupla interpretação, impossibilitando o entendimento e dificultando a execução do item. |
| Inconsistência | Quando as informações contidas nos artefatos entram em conflito com outra informação do mesmo tópico ou em tópicos diferentes. |

Lista de Discrepâncias

| Nº Discrepância | Nº Página | Artefato de Teste | Tipo Defeito | Descrição |
|---|---|---|---|---|
| **01** | | | | |
| **02** | | | | |

# Apêndice E – Questionário de Avaliação Pós Experimento

**QUESTIONÁRIO DE AVALIAÇÃO PÓS-EXPERIMENTO**

Por favor, permita que registremos sua opinião sobre a aplicação da abordagem para inspeção de artefatos de teste de Software baseada em *checklist* (*TestCheck*), preenchendo o questionário abaixo. A sua participação e opinião é muito importante para condução da nossa pesquisa.

**1. Classifique o grau de dificuldade da aplicação da abordagem baseada em *checklist*?**

Muito Fácil ( ) Fácil ( ) Difícil ( ) Muito Difícil ( )

**2. Quais os aspectos da abordagem que tornam sua aplicação fácil / difícil de usar?**

**3. Você identificaria os defeitos utilizando apenas seus conhecimentos (*Ad Hoc*)? Algum defeito você só conseguiria detectar utilizando os itens de avaliação presentes nos *Checklists*?**

**4. De que maneira a técnica o auxiliou a identificar defeitos nos artefatos?**

___ Negativamente. A abordagem baseada em checklist atuou como um obstáculo. O meu desempenho teria sido melhor se eu não o tivesse utilizado.

___ Neutro. Acho que encontraria os mesmos defeitos caso não tivesse utilizado a abordagem.

___ Positivamente. A abordagem me auxiliou na detecção de defeitos. Talvez não tivesse detectado alguns defeitos caso não a tivesse utilizado.

**5. A abordagem é capaz de avaliar e propor melhorias, mudanças, apontando onde é possível encontrar o (s) erro (s)?**

Sim ( ) Não ( ) Talvez ( ) ( ) N/A _____

**5. A abordagem é capaz de avaliar a qualidade de um artefato de teste de Software?**

Sim ( ) Não ( ) Talvez ( ) ( ) N/A _____

**6. Existe algum item de avaliação que não foi compreendido?**

Não ( ) Sim ( ) Identifique _____

**7. O quão custoso é sua utilização?**

**8. Você utilizaria a abordagem em inspeções futuras?**

Sim ( ) Não ( ) Talvez ( )

**8. Em sua opinião, como a abordagem poderia ser melhorada?**

**Espaço reservado para quaisquer comentários que julgar pertinente.**

**Muito obrigada por sua participação!**