TestCheck – Uma Abordagem Baseada em *Checklist* para Inspecionar Artefatos de Teste de Software

Jardelane Brito, Jeanne de Castro Trovão, Arilo Claudio Dias-Neto

Instituto de Computação (IComp) – Universidade Federal do Amazonas (UFAM)
Av. General Rodrigo Octávio, 6.200, Campus Universitário Senador Arthur Virgílio Filho – Setor Norte - Manaus - CEP 69.077-000 – Manaus – AM – Brasil

{jardelane.brito,arilo}@icomp.ufam.edu.br, jeannetrovao@gmail.com

Abstract. *The quality of the tests applied to a software project is an important factor for the quality of a software product. This paper presents a checklist-based approach, TestCheck, for inspection of software testing artefacts (test plans, cases, and procedures). The steps followed to develop TestCheck and the checklists composing TestCheck are described. Moreover, the results of empirical studies applied that evaluated efficiency and effectiveness of the proposed approach for defect detection in test artefacts are presented, and they indicate TestCheck's checklists make possible defects detection and this approach obtained better results when compared to an ad-hoc approach.*

Resumo. *A qualidade dos testes aplicados em um projeto de software é um fator determinante para a qualidade do produto final. Este artigo apresenta uma abordagem baseada em checklist, TestCheck, para inspeção de artefatos de teste de software (planos, casos e procedimentos de teste). São descritos os passos seguidos para a construção de TestCheck e os resultados de estudos experimentais que avaliaram a eficiência e eficácia da abordagem para detecção de defeitos. Os resultados indicam que os checklists propostos possibilitam a identificação de defeitos, e que esta abordagem apresenta melhores resultados quando comparada a uma abordagem ad-hoc.*

1. Introdução

Apesar dos avanços em Engenharia de Software na utilização de métodos e técnicas para obtenção de sistemas 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 et al., 2010; Poon et al., 2010].

Os testes, quando conduzidos sem objetivo, planejamento e técnicas adequadas, podem apresentar consequências desagradáveis para um projeto de software, pois a descoberta de falhas 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 e Basili, 2001] indica que o custo para corrigir uma falha revelada apenas 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 sistema. No entanto, 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].

Com isso, aplicar técnicas de garantia da qualidade nos artefatos produzidos ao longo do processo de testes pode ser uma solução viável para garantir a qualidade de testes em projetos de software. Neste contexto, inspeção de software [Fagan, 1976] é uma técnica de garantia de qualidade estabelecida para a detecção antecipada de defeitos no desenvolvimento do sistema, sendo aplicadas a qualquer artefato de software. É uma abordagem complementar dentro do processo de Verificação e Validação (V&V) para verificação e análise de sistema [Pressman, 2006].

Portanto, com o objetivo de minimizar e prevenir a ocorrência de defeitos em artefatos produzidos ao longo do processo de testes e levando em consideração os resultados que se tem obtido com a aplicação de técnicas de inspeção na revisão de artefatos de software [Shull et al., 2000; Conradi et al., 2003], este artigo apresenta *TestCheck*, uma abordagem baseada em *checklist* para inspeção de artefatos de teste. Os *checklists* que compõem *TestCheck* foram criados com base em padrão internacional para documentação de testes: IEEE-Std 829 [IEEE, 2008]. Além disso, essa abordagem foi avaliada por meio de estudos experimentais, que indicaram a viabilidade em utilizá-la para a identificação de defeitos em artefatos de teste.

Este artigo está estruturado da seguinte forma. A Seção 2 apresenta um referencial teórico sobre técnicas para inspecionar artefatos de software e documentação de testes, além de trabalhos relacionados que abordam a aplicação de inspeção no contexto de testes de software. A Seção 3 descreve a concepção de uma abordagem baseada em *checklist* de apoio à inspeção de artefatos de teste – *TestCheck*. A seção 4 apresenta os resultados de estudos experimentais conduzidos para avaliação da abordagem *TestCheck* que indicam a viabilidade da abordagem proposta. Por fim, a Seção 5 apresenta as conclusões deste trabalho e seus próximos passos.

2. Trabalhos Relacionados

2.1. Processo/Documentação de Teste de Software

O processo de testes de software tem como objetivo revelar falhas no sistema, para que estas sejam solucionadas e seja entregue um produto com qualidade, dentro de prazos e custos controlados e compatíveis com o mercado [Crespo et al., 2004]. A execução deste processo deve ocorrer ao longo do processo de desenvolvimento de um software, e possui diferentes tarefas de acordo com a atividade atual do projeto.

Em [Dias-Neto e Travassos, 2006] é apresentado um exemplo de processo de testes, no qual suas atividades foram definidas baseando-se em fontes de conhecimento providas na literatura técnica e seus artefatos produzidos são os documentos especificados pelo IEEE-Std-829 (*Standard for Software Test Documentation*) [IEEE, 2008], que consiste em um padrão internacional cujo objetivo é descrever um conjunto de roteiros para diversos documentos produzidos ao longo de um processo de teste de software. Este processo é dividido em dois subprocessos:

- **Subprocesso de planejamento de teste:** responsável pelas atividades de preparação dos testes, desde a seleção da equipe e itens a serem testados, assim como a construção de planos, casos e procedimentos de teste.
- **Subprocesso de execução dos testes:** responsável pela execução dos testes conforme o planejamento, monitorando as atividades e registrando as eventuais falhas reveladas.

2.2. Inspeção aplicada a Teste de Software

Ao realizar uma investigação na literatura em busca de pesquisas que envolvessem aplicação de inspeções ao longo do processo de testes, percebeu-se que poucos trabalhos relacionados a este tema têm sido reportados na literatura técnica.

Em [Lanubile e Mallardo, 2007], é apresentada uma pesquisa sobre a aplicação de inspeção em códigos de casos de testes automatizados. Neste estudo, os autores perceberam que quando os casos de teste são defeituosos, o tempo dos desenvolvedores seria desperdiçado para resolver problemas que na verdade não são originários do código fonte. Com isso, postulou-se que a qualidade dos casos de teste poderia ser assegurada através de inspeções de software. Os autores usaram duas abordagens, *ad-hoc* e baseada em *checklist*. Os resultados indicaram que inspeção de software baseada em *checklist* melhorou a qualidade do código de teste, principalmente em relação a sua repetibilidade (uma das características mais importantes da automação de teste), além da 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 e Sakka, 2006] é descrita 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 e teste de aceitação, que foram reivindicadas para melhorar a qualidade do produto. Assim, foi proposta 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 melhor 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 e submetido a dois estudos. O primeiro estudo 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. Constatou-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.

Assim, visando atender a uma carência identificada a partir desta revisão da literatura, foi desenvolvida uma abordagem, denominada *TestCheck*, com o objetivo de melhorar a qualidade dos testes em projetos de software a partir da avaliação dos artefatos produzidos ao longo do seu processo.

3. *TestCheck* – Técnica para Inspeção de Artefatos de Teste de Software

Esta seção descreve o processo de criação de *TestCheck*, uma abordagem para inspeção de artefatos de teste. Para definir *TestCheck*, os autores se basearam nas características técnicas, identificadas por [Babar, 2004], para composição de uma abordagem de inspeção. 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).

A seguir, visando caracterizar a abordagem proposta, os resultados obtidos com a realização dessas tarefas são descritos.

3.1. Escolha da Técnica de Detecção de Defeitos

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*. No entanto, esta não oferece apoio sistemático ou procedimento de execução formal da inspeção. Além do mais, os resultados obtidos dependem da capacidade, competência e experiência do inspetor [Barcelos e Travassos, 2006].

Em seguida, avaliou-se a possibilidade do uso de técnicas de leitura. Estas são procedimentos que visam guiar individualmente os inspetores no entendimento de um artefato de software e, por consequência, na identificação de discrepâncias [Shull et al., 2000]. Abordagens de avaliação baseadas nessa técnica [Shull et al., 2000; Conradi et al., 2003] são mais eficientes na detecção de defeitos quando comparadas a outras técnicas de inspeção, como *checklists*. Porém, para que seja utilizada, o artefato deve ser representado em uma forma específica e padronizada, característica que ainda não pode ser atingida em artefatos de teste. Apesar da existência do padrão IEEE-Std 829 [IEEE, 2008] como roteiro para documentação dos artefatos de teste, ele ainda é pouco utilizado na prática, como pode ser observado em uma pesquisa de opinião aplicada na indústria de software [Dias-Neto et al., 2006]. Além disso, este padrão descreve apenas roteiros que podem ainda ser customizados ao 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 [Barcelos e Travassos, 2006]. 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 sistema.

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.

3.2. Escolha dos Artefatos de Teste a serem Inspecionados

Para elaboração de *TestCheck*, usou-se a lista de artefatos sugeridos pelo IEEE-Std 829 [IEEE, 2008] como artefatos candidatos a serem inspecionados. Eles se aplicam a diferentes metodologias de desenvolvimento (ex: baseadas em processos ou ágeis): plano ou projeto de teste, especificação de caso ou procedimento de teste, log de teste, relatório de incidente de teste e relatório de resumo dos testes.

A seleção dos artefatos de teste a serem inspecionados se baseou, inicialmente, na fase do processo de teste, que é dividido nos subprocessos de planejamento e execução [Dias-Neto e Travassos, 2006]. Neste momento, optou-se por avaliar artefatos

produzidos durante o planejamento dos testes como forma de prevenir a incidência de defeitos durante a execução dos testes, além de antecipar o momento da 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 uma pesquisa de opinião publicada em [Dias-Neto et al., 2006] que foi aplicada em organizações de software brasileiras. Neste estudo, percebeu-se a maior ênfase das organizações de software na produção dos seguintes artefatos ao longo de seus processos de teste:

- **Plano de Teste:** apresenta o planejamento para execução do teste, incluindo a abrangência, objetivos, abordagem, divisão de tarefas, recursos (humanos e físicos), cronograma e riscos das atividades de teste [Crespo et al., 2004].
- **Especificação de Caso de Teste:** define os casos de teste, incluindo dados de entrada, resultados esperados, ações e condições gerais para a execução do teste [Crespo et al., 2004].
- **Especificação de Procedimento de Teste:** especifica os passos, suas pré-condições e restrições especiais para executar um conjunto de casos de teste [Crespo et al., 2004].

Os artefatos selecionados podem ser aplicados a testes em diferentes níveis (ex: unidade, integração, sistema e aceitação). Assim, os *checklists* que compõem a abordagem *TestCheck* foram elaborados de forma a serem aplicados a estes diferentes níveis, sendo formados por itens que proveem questionamentos a respeito da estrutura e conteúdo dos artefatos sem está 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.

3.3. Taxonomia de Defeitos

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 et al., 2000], devidamente interpretada para o contexto de Teste de Software (Tabela 1).

Tabela 1. Taxonomia de Defeitos

Tipos de Defeitos	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); 4. Quando um elemento do planejamento dos testes presente em um artefato (caso ou procedimento de teste) não foi definido no plano de teste; 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). 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.

3.4. Itens de Avaliação dos *Checklists*

Para compor os *checklists* que formam *TestCheck*, vários itens de avaliação foram definidos e agrupados de acordo com o artefato de teste a ser inspecionado. 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 1).

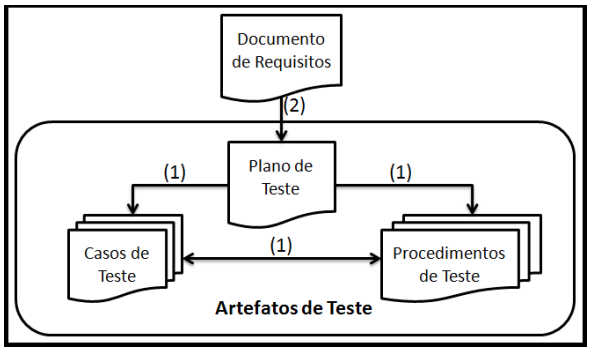


Figura 1. Perspectivas de avaliação dos itens que compõem *TestCheck*.

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.

3.4.1. *Checklist* para Inspeção de Plano de Teste

O plano de teste deve conter informações importantes para a condução dos testes em um projeto de software, conforme descrito anteriormente. Assim, defeitos inseridos neste documento podem comprometer o resultado final dos testes.

O padrão IEEE-Std 829 apresenta 16 itens que devem fazer parte deste artefato. Dois itens não foram considerados importantes para serem revisados neste artefato e não foram incluídos no *checklist* desenvolvido, pois eles representam conteúdos informativos que não seriam cruciais para a execução dos testes: Introdução (resumo dos testes a serem executados) e *Aprovação* (nome e espaço para assinatura de quem aprova o plano). Além desses, dois itens (Tarefas de Teste e Artefatos de Teste) foram unificados em um único item (Tarefas de Teste). Assim o *checklist* proposto é composto por 13 itens de avaliação, listados na Tabela 2, a serem avaliados por 38 questões.

Por limitação de espaço não é possível apresentar todo o *checklist* proposto para inspeção do plano de teste e suas questões. A Tabela 3 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 2. Itens que compõem o *checklist* para inspeção de planos de teste

Itens de Avaliação	Objetivo de suas questões	Nº de 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.	4
Características de Qualidade	Avaliar a descrição de quais características de qualidade serão testadas ou não a partir do 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 o 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	Avaliar 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

Tabela 3. 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

Um extrato do *checklist* para avaliação de plano de teste está exibido na Figura 2.

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?			

Figura 2. Parte do *checklist* para inspeção 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. Um item não foi incluído no *checklist* desenvolvido: Requisitos Procedurais Especiais (restrições dos procedimentos de teste que executam um caso de teste). Este item foi incluído no próximo *checklist* a ser descrito (para inspecionar procedimento de teste). Assim, o *checklist* desenvolvido para avaliação de casos de teste possui 6 itens de avaliação (Tabela 4) detalhados em 19 questões.

Tabela 4. Itens que compõem o *checklist* para inspeção de casos de teste

Itens de Avaliação	Objetivo com esta avaliação	Nº de 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.	4
Especificações de Saída / Resultados esperados	Avaliar a corretude e completude dos resultados ou comportamentos esperados após os testes.	4
Necessidades do Ambiente de Teste	Avaliar a descrição das características físicas e softwares/hardware necessários para execução de um caso de testes.	2
Dependência entre casos de teste	Avaliar a corretude do relacionamento de dependência entre os casos de teste do projeto.	3

Por limitação de espaço também não é possível apresentar todo o *checklist* proposto e suas questões. A Tabela 5 apresenta o detalhamento de um item de avaliação que compõe este *checklist*: ESPECIFICAÇÕES DE ENTRADA.

Um extrato do *checklist* para avaliação de caso de teste está exibido na Figura 3.

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?			

Figura 3. Parte do *checklist* para inspeção de caso de teste.

Tabela 5. Descrição das questões para o item ESPECIFICAÇÕES DE ENTRADA

Descrição: caso de teste é formado por dados de entrada a serem usados para exercitar o software sob teste		
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 e Inconsistência

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 para o campo CPF, podendo este ser numérico ou literal.	C09	Omissão

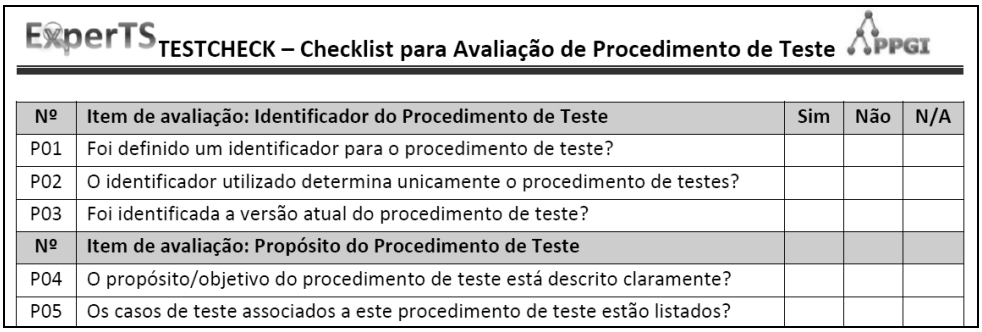
3.4.3. *Checklist* para Inspeção de Especificação de Procedimento de Teste

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 para avaliação de procedimentos de teste possui 4 itens de avaliação (Tabela 6) e 16 questões.

Tabela 6. Itens que compõem o *checklist* para inspeção de procedimentos de teste

Itens de Avaliação	**Objetivo com esta avaliação**	**Nº de 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 e corretude dos passos e descritos para um procedimento de teste e seu sequenciamento.	7

Por limitação de espaço também não é possível apresentar o detalhamento das questões que compõem o *checklist* proposto para inspeção do procedimento de teste. No entanto, este segue a mesma estrutura apresentada para os artefatos anteriores (plano e caso de teste), descritos na Tabela 3 e Tabela 5. Um extrato do *checklist* para avaliação de caso de teste está exibido na Figura 4.


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?			

Figura 4. Parte do *checklist* para inspeção de procedimento de teste.

A versão completa dos *checklists* que compõem *TestCheck* pode ser encontrada em http://www.icomp.ufam.edu.br/experts/phocadownload/experts-testcheck-v3.zip.

Concluída a fase de concepção da abordagem de apoio à inspeção de artefatos de teste, na próxima seção serão apresentados os resultados de estudos experimentais conduzidos para avaliação do *checklist* proposto.

4. Avaliação Experimental de *TestCheck*

Após a concepção dos *checklists* que compõem a abordagem *TestCheck* apresentados na seção 3, iniciou-se a fase de avaliação desta abordagem. Para isso, a metodologia definida por [Shull et al., 2001] tem sido utilizada com o propósito de prover uma avaliação experimental de *TestCheck* desde a avaliação de sua viabilidade em relação ao seu propósito de identificar defeitos em artefatos de teste de software até a sua transferência para ambientes industriais. Até o momento foram realizados dois estudos de viabilidade, o que corresponde à primeira fase da metodologia adotada. Seguindo a metodologia, os próximos passos seriam a condução de um Estudo de Observação, Estudo de Caso no ciclo de vida de um projeto e por fim a transferência para indústria.

Tais estudos tiveram o objetivo de 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. Esta seção descreve partes dos resultados obtidos nestes estudos experimentais realizados para avaliação de *TestCheck*, com ênfase na análise de cobertura e tipos de defeitos detectados por *TestCheck*. Detalhes a respeito do planejamento, projeto, execução e análise estatística em ambos os estudos podem ser encontrados em [Brito e Dias-Neto, 2012].

4.1. Estudo 1: Estudo de Viabilidade

O principal objetivo desse estudo foi identificar se *TestCheck* realmente faz o que ela se propõe a fazer e se é viável continuar a despender recursos para desenvolvê-la. Com isso, procurou-se avaliar principalmente se os *checklists* que compõem a abordagem permitem que um inspetor identifique defeitos em artefatos de teste.

Para a realização desse estudo, foram utilizados documentos de teste de dois projetos diferentes, um desenvolvido no contexto de uma disciplina de Engenharia de Software na UFAM (chamado *Acadêmico*) e outro desenvolvido em um projeto real que teve como objetivo a construção de um sistema de gerenciamento de usuários e seus acessos a funcionalidades em um sistema de informação (chamado *Industrial*). Cada projeto foi composto por 1 plano de teste, 5 casos de teste, 5 procedimentos de teste e 1 documento de requisitos (que serviu como apoio e não foi inspecionado).

Ambos os artefatos de teste dos projetos foram criados de acordo com as recomendações do padrão IEEE-Std 829 [IEEE, 2008]. Em relação à corretude desses documentos, nenhum defeito foi adicionado pelos pesquisadores, sendo avaliadas as versões produzidas pelos seus autores originais.

Neste estudo foram aplicadas duas abordagens para inspeção dos artefatos de teste a fim de prover uma comparação dos resultados: *TestCheck* e *Ad-Hoc*. A abordagem *ad-hoc* foi aplicada por não ter sido identificada na literatura técnica outra abordagem similar à *TestCheck* que pudesse ser aplicada nos estudos, conforme relatado na Seção 2. Além disso, o objetivo deste primeiro estudo é prover um *baseline* a respeito da abordagem *TestCheck*, possibilitando futuras comparações ao longo de sua evolução. Desta forma, avaliá-la inicialmente com uma abordagem *ad-hoc* contribui neste sentido.

Ao total, 12 alunos de graduação e pós-graduação participaram deste estudo. Todos aplicaram ambas as abordagens com os artefatos dos dois projetos definidos, primeiramente *ad-hoc* para evitar viés durante o estudo e em seguida *TestCheck* (após um treinamento realizado). A Tabela 7 apresenta os valores obtidos e as médias para as variáveis: número de defeitos, número de falso-positivos, tempo (em minutos), eficácia (defeitos / [defeitos + falso-positivos]) e eficiência (defeitos / tempo). É possível observar que para as variáveis defeitos, eficácia e eficiência, *TestCheck* provê melhores resultados quando comparada aos resultados obtidos para a abordagem *ad-hoc*.

Tabela 7. Resultado do Estudo 1 [Brito e Dias-Neto, 2012].

Fatores		Defeitos		Falso-Positivos		Tempo (minutos)		Eficácia	Eficiência
		TOT	MED	TOT	MED	TOT	MED	TOT	TOT
Técnicas	*Ad-hoc*	136	11,33	114	9,5	1572	131	54,4%	0,08
	TestCheck	470	39,16	150	12,5	1943	161,9	75,80%	0,24
Projetos	Industrial	306	25,5	99	8,25	1685	140,4	75,55%	0,18
	Acadêmico	300	25,0	165	13,75	1830	152,5	64,51%	0,16

Como citado anteriormente, foram aplicados testes estatísticos (Análise da Média, *Outlier* e de Variância – ANOVA), e os resultados foram publicados em [Brito e Dias-Neto, 2012].

- **Análise de Cobertura de Defeitos**

Para analisar a cobertura dos defeitos detectados por cada técnica, foram removidos defeitos duplicados (encontrado por mais de um inspector). Em seguida, separou-se defeitos detectados por cada ou ambas as técnicas. O resultado desta análise está apresentado na Figura 5.

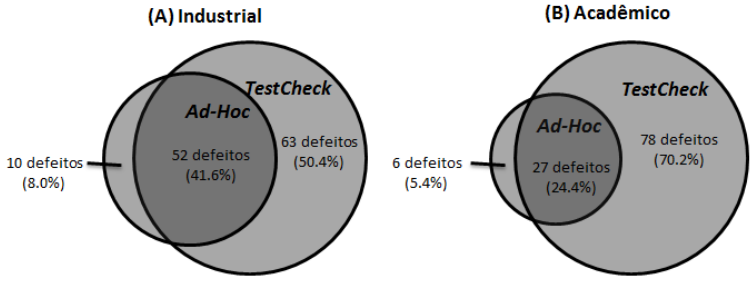


Figura 5. Análise de interseção de defeitos detectados pelas técnicas – Estudo 1.

É possível observar que defeitos foram detectados pela abordagem *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 abordagem *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 abordagem *ad-hoc*.

Considerando os diferentes tipos de defeitos detectados por *TestCheck* (Figura 6), 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].

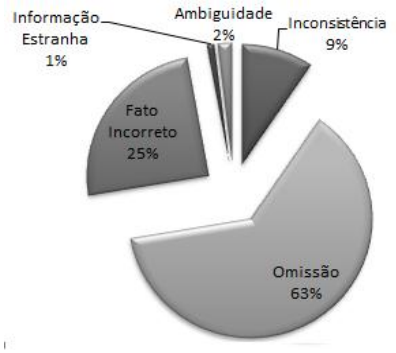


Figura 6. Distribuição dos tipos de defeitos detectados por *TestCheck* no Estudo 1

Apesar do bom desempenho neste primeiro estudo, sabe-se que esta técnica ainda está em fase de amadurecimento. Então, é importante avaliar pontos onde melhorias poderiam ser introduzidas. Um dos pontos a serem melhorados foi o número de **falso-positivos** gerados a partir da aplicação de *TestCheck*. Obteve-se taxa de 12,5 falso-positivos gerados por participante, considerada alta pelos pesquisadores, visto que a taxa de defeitos detectados pela técnica foi de 39,16 por participante. Com isso, aproximadamente a cada 3 defeitos, 1 falso-positivo foi gerado por *TestCheck*. Portanto, o objetivo de evolução é diminuir a taxa de falso-positivos encontrados no primeiro estudo realizado e manter o aumento da taxa de defeitos encontrados utilizando *TestCheck* quando comparada à técnica *ad-hoc*.

Após melhorias realizadas nos *checklists* (modificação de 15 questões e remoção de 2 questões), foi gerado uma nova versão de *TestCheck*, que foi avaliada em um segundo estudo, descrito na próxima subseção [Brito Dias-Neto, 2012].

4.2. Estudo 2: Estudo de Viabilidade

Para avaliar a evolução de *TestCheck*, fez-se necessário um segundo estudo experimental 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 da técnica *TestCheck* quando comparada a uma abordagem *ad-hoc*. Ele contou com 10 alunos de graduação de outra disciplina de Engenharia de Software diferente em relação ao primeiro estudo, e seu foco foi avaliar a redução da taxa de falso-positivos gerados e manutenção da taxa de detecção de defeitos maior que a taxa obtida para a abordagem *ad-hoc*. Os resultados obtidos do segundo estudo estão descritos na Tabela 8.

Analisando a evolução de *TestCheck* (Tabela 8), a taxa de falso-positivos gerados ficou em 4,9 por inspetor, resultado melhor que o obtido no estudo anterior no qual a taxa foi de 12,5. Ou seja, analisando apenas a média obtida, o objetivo com a evolução de *TestCheck* foi alcançado sem comprometer o segundo objetivo do estudo que foi manter a taxa de detecção de defeitos de *TestCheck* alta quando comparada à técnica *ad-hoc* (35,0 por participante). Em média, a cada 7 defeitos, 1 falso-positivo é gerado pela técnica *TestCheck*, enquanto que no estudo anterior a relação foi de 3:1.

Tabela 8. Resultados comparativos das técnicas [Brito e Dias-Neto, 2012].

Fatores		Defeitos		Falso-Positivos		Tempo (minutos)		Eficácia	Eficiência
		TOT	MED	TOT	MED	TOT	MED	TOT	TOT
Técnicas	*Ad-hoc*	119	11,9	29	2,9	1232	123,2	80,4%	0,09
	TestCheck	350	35,0	49	4,9	2446	244,6	87,7%	0,14

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.

- **Análise de 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 7.

Novamente observou-se que *TestCheck* cobre a maior parte dos defeitos detectados pela abordagem *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 pela abordagem *TestCheck*. O resultado deste estudo confirma as evidencias observadas no primeiro estudo, ou seja, a abordagem contribui para detectar defeitos novos e não triviais que não são detectados por uma abordagem *ad-hoc*.

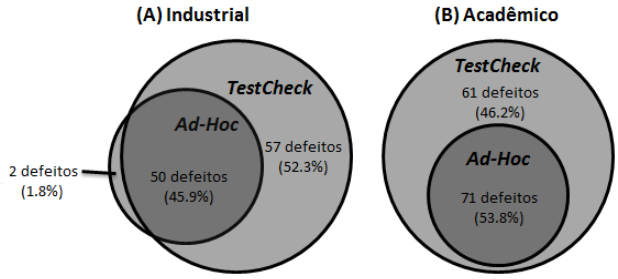


Figura 7. Análise de interseção de defeitos detectados pelas técnicas – Estudo 2.

A Figura 8 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.

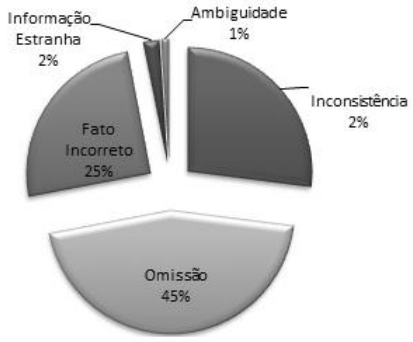


Figura 8. Distribuição dos tipos de defeitos detectados por *TestCheck* no Estudo 2

5. Conclusões e Trabalhos Futuros

Este artigo apresentou a abordagem *TestCheck*, que serve de apoio para detecção de defeitos em artefatos de teste de software, e os resultados de dois estudos experimentais

(estudo de viabilidade e de observação) que analisaram sua eficiência e eficácia em comparação a uma abordagem *ad-hoc*. A técnica tem sido desenvolvida a partir de recomendações de padrões internacionais para documentação dos testes, e sua estratégia de concepção segue passos descritos na literatura técnica para a elaboração de técnicas de inspeção de software.

A análise dos estudos realizados sugere resultados positivos para abordagem *TestCheck* e possíveis melhorias, que estão sendo aplicadas em seu processo de amadurecimento. Ao apresentar os resultados da eficiência e eficácia, observou-se que ambos os resultados apontam maior detecção de defeitos para abordagem *TestCheck*. Portanto, para esta análise *TestCheck* seria mais eficiente e eficaz. É importante ressaltar que esta análise possibilitou identificar possíveis melhorias nos *checklists* que devem ser feitas para o próximo estudo, identificando quais questões proporcionaram um maior número de falso-positivos e quais não proporcionaram a detecção de defeitos.

Como trabalhos futuros, os próximos passos são evoluir os *checklists*. Para isso, está sendo realizada uma análise das questões que tiveram um fraco desempenho no segundo estudo tendo como parâmetro a quantidade de falsos positivos que cada uma gerou comparando com a quantidade de defeitos. Em seguida, novos estudos serão realizados com a nova versão concebida, agora aplicando-a em um estudo de caso na indústria de software, seguindo a metodologia científica proposta em [Shull et al., 2001].

Agradecimentos

Os autores agradecem à FAPEAM, CNPq (processo 575696/2008-7) e INCT-SEC (processos 573963/2008-8 e 08/57870-9) pelo apoio para a realização desta pesquisa, aos participantes dos estudos realizados pela colaboração.

Referências Bibliográficas

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.

Brito, J.; Dias-Neto, A.C. (2012), "Conduzindo Estudos Experimentais para Avaliação de uma Técnica de Inspeção de Artefatos de Teste de Software", In: Experimental Software Engineering Latin American Workshop (ESELAW), Buenos Aires, Abril.

Boehm, B., Basili, V. (2001), "Software Defect Reduction Top 10 List", IEEE Computer, vol. 34(1): pp. 135-137.

Conradi, R., Mohagheghi, P.; Arif, T.; "Object-Oriented Reading Techniques for Inspection of UML Models - An Industrial Experiment". In: European Conference on Object-Oriented Programming, Darmstadt, Germany, Vol. 2743, pp. 69-81.

Crespo, A.N. et al. (2004); Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo; In: Simpósio Brasileiro de Qualidade de Software (SBQS), Brasília, pp. 271-285,.

Davis, A. (1990); "Software Requirement Analysis and Specification", Prentice-Hall International.

Dias-Neto, A.C., Natali, A.C.; Rocha, A.R., Travassos, G.H. (2006), "Caracterização do Estado da Prática das Atividades de Teste em um Cenário de Desenvolvimento de Software Brasileiro", V Simpósio Brasileiro de Qualidade de Software, Vila Velha, pp. 27-41.

Dias-Neto, A.C.; Travassos, G.H. (2006), "Maraká: Uma Infra-Estrutura Computacional para Apoiar o Planejamento e Controle de Teste de Software", V Simpósio Brasileiro

de Qualidade de Software, Vila Velha, pp. 248-262.

Fagan. M. (1976); "Design and Code Inspections to Reduce Errors 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, pp. 347-353.

IEEE Standard 829-2008 (2008): Standard for Software Test Documentation, IEEE Press.

Itoken, J. V.; Mantyla, M,V.; Lassenius, C. (2007), "Defect Detection Efficiency: Test Case Based vs. Exploratory Testing", In: First International Symposium on Empirical Software Engineering and Measurement, pp. 61-70, DOI 10.1109/ESEM.2007.56.

Kalinoswki, M.; Spínola, R. O.; Dias-Neto, A. C.; Bott, A.; Travassos, G. H. (2007), Inspeções de Requisitos de Software em Desenvolvimento Incremental: Uma Experiência Prática. In: Simpósio Brasileiro de Qualidade de Software, Porto de Galinhas.

Babar, M.A. (2004). Scenarios, "Quality Attributes, and Patterns: Capturing and Using their Synergistic Relationships for Product Line Architectures". Fraunhofer Institute Experimental Software Engineering, Germany.

Lanubile, F.; Mallardo, T. (2007), " Inspecting automated test code: a preliminary study", In Proceedings of the 8th international conference on Agile processes in software engineering and extreme programming (XP'07), Giulio Concas, Ernesto Damiani, Marco Scotto, and Giancarlo Succi (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 115-122.

Luo, L. (2010), Software Testing Techniques - Technology Maturation and Research Strategies; In: Institute for Software Research International - Carnegie Mellon University, Pittsburgh, Technical Report 17-939A.

Poon, P.L.; Tse, T. H.; Tang, S.F.; Kuo, F.C. (2011), "Contributions of tester experience and a checklist guideline to the identification of categories and choices for software testing", *Software Quality Control* 19, 1 (March 2011), pp. 141-163. DOI=10.1007/s11219-010-9109-4 http://dx.doi.org/10.1007/s11219-010-9109-4.

Pressman, R.S. (2006), "Engenharia de Software", 6. Ed: São Paulo: MCGRAW-Hill.

Shull, F., Carver, J., Travassos, G. H. (2001), "An empirical methodology for introducing software processes", In Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-9). ACM, New York, NY, USA, pp. 288-296. DOI=10.1145/503209.503248.

Shull, F., Rus, I., Basili, V. (2000), "How perspective based reading can improve requirements inspections", IEEE Computer, v. 33, n. 7, pp 73-79.