UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

MARLLOS PAIVA PRADO

Contribuições ao Suporte Cognitivo em Teste de Software Unitário: Um Framework de Tarefas e uma Agenda de Pesquisa

Goiânia
2018

sistema de sib ufg

PRPG
PRÓ REITORIA DE PÓS GRADUAÇÃO

UFG

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

1. Identificação do material bibliográfico: **[ ] Dissertação** **[x] Tese**

2. Identificação da Tese ou Dissertação:

Nome completo do autor: Marllos Paiva Prado

Título do trabalho: Contribuições ao Suporte Cognitivo em Teste de Software Unitário: Um Framework de Tarefas e uma Agenda de Pesquisa

3. Informações de acesso ao documento:

Concorda com a liberação total do documento [x] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF da tese ou dissertação.

Marllos Paiva Prado
Assinatura do(a) autor(a)[2]

Ciente e de acordo:

Assinatura do(a) orientador(a)[2]

Data: 16 / 03 / 2018

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente
- Submissão de artigo em revista científica
- Publicação como capítulo de livro
- Publicação da dissertação/tese em livro

[2] A assinatura deve ser escaneada.

Versão atualizada em maio de 2017.

MARLLOS PAIVA PRADO

Contribuições ao Suporte Cognitivo em Teste de Software Unitário: Um Framework de Tarefas e uma Agenda de Pesquisa

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

Área de concentração: Ciência da Computação.

Orientador: Prof. Dr. Auri Marcelo Rizzo Vincenzi

Goiânia
2018

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Paiva Prado, Marllos
Contribuições ao Suporte Cognitivo em Teste de Software Unitário: Um Framework de Tarefas e uma Agenda de Pesquisa [manuscrito] / Marllos Paiva Prado. - 2018.
154 f.: il.

Orientador: Prof. Dr. Auri Marcelo Rizzo Vincenzi.
Tese (Doutorado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação em rede (UFG/UFMS), Goiânia, 2018.
Bibliografia. Apêndice.
Inclui lista de figuras, lista de tabelas.

1. Teste Unitário. 2. Suporte Cognitivo. 3. Aspectos Humanos. 4. Teste de Software. 5. Ferramentas de Teste. I. Marcelo Rizzo Vincenzi, Auri, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

UFMS

Ata de Defesa de Tese de Doutorado

Aos dezesseis dias do mês de março de dois mil e dezoito, no horário das catorze horas, foi realizada, nas dependências do Instituto de Informática da UFG, a defesa pública da Tese de Doutorado do aluno Marllos Paiva Prado, matrícula no. 2013 0345, intitulada **"Contribuições ao Suporte Cognitivo em Teste de Software Unitário: Um Framework de Tarefas e uma Agenda de Pesquisa"**.

A Banca Examinadora, constituída pelos professores:

Prof. Dr. Auri Marcelo Rizzo Vincenzi – INF/UFG – DC/UFSCar - orientador

Profa. Dra. Sandra Camargo Pinto Ferraz Fabbri – DC/UFSCar

Prof. Dr. Rodrigo Funabashi Jorge – FACOM/UFMS

Prof. Dr. Cássio Leonardo Rodrigues – INF/UFG

Prof. Dr. Renato de Freitas Bulcão Neto – INF/UFG

emitiu o resultado:

(X) Aprovado

( ) Aprovado com revisão

(A Banca Examinadora deve definir as exigências a serem cumpridas pelo aluno na revisão, ficando o orientador responsável pela verificação do cumprimento das mesmas.)

( ) Reprovado

com o seguinte parecer: _____

_____

_____

_____

Prof. Dr. Auri Marcelo Rizzo Vincenzi

Profa. Dra. Sandra Camargo Pinto Ferraz Fabbri

Prof. Dr. Rodrigo Funabashi Jorge

Prof. Dr. Cássio Leonardo Rodrigues

Renato Bulcão Neto

Prof. Dr. Renato de Freitas Bulcão Neto

Marllos Paiva Prado

Possui mestrado em ciência da computação pelo Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP). É professor no Instituto Federal de Goiás (IFG).

Ao meu querido avô, Afrânio Marques *"in memoriam"*.

Agradecimentos

Em primeiro lugar, agradeço a Deus e ao meu avô Afrânio Marques (*in memoriam*) pelo amor e por terem plantado em mim a sabedoria e a persistência necessárias para eu cumprir mais essa etapa da minha vida.

Agradeço à minha amada, Natália L. Nogueira, pelo carinho, dedicação e amor com que regou e manteve viva a minha vontade de crescer e amadurecer durante esse processo. Minha admiração e afeto por você só me fazem amá-la cada dia mais.

Ao meu orientador Auri Vincenzi, pelo exemplo, pela confiança, pela sabedoria, pela amizade e pela liberdade. O seu apoio foi determinante em todas as etapas dessa jornada. Sou muito grato por você ter feito parte da minha formação pessoal, acadêmica e profissional.

Agradeço aos meus familiares que me apoiaram e me ajudaram a crescer e a alcançar meus objetivos.

Agradeço ao Cláudio Araújo e à Fabiana Mendes pelo companheirismo e pela amizade de tantos anos.

Agradeço a todos os amigos e colegas do Instituto Federal de Goiás e do Instituto de Informática da UFG que me apoiaram e me ajudaram durante essa etapa.

Agradeço a todos os professores que aceitaram participar da minha banca avaliadora pelas contribuições, pelos questionamentos e pelas sugestões feitas no meu trabalho.

Agradeço a todos os professores que tive ao longo da minha vida e que se preocuparam com a minha formação.

Thanks to all my friends from UVic! Special thanks to Dr. Margaret-Anne Storey (Peggy) for her invaluable contribution, commitment, orientation, and friendship! Also, special thanks to my great friends Eric Verbeek and Alisha Brown for their invaluable companionship, caring, friendship and loyalty! My stay in Canada was memorable, and your involvement was a precious component of this experience.

Agradeço a todos os demais amigos que me apoiaram e torceram por mim. Ainda que eu não consiga citar todos nominalmente aqui, carrego cada um de vocês no coração com muito carinho.

Agradeço ao CNPq (202095/2014-2) e à FAPEG (2016/10267000693) pelo apoio financeiro.

The difficulty lies not so much in developing new ideas as in escaping from old ones,
John Maynard Keynes.

Resumo

Paiva Prado, Marllos. **Contribuições ao Suporte Cognitivo em Teste de Software Unitário: Um Framework de Tarefas e uma Agenda de Pesquisa**. Goiânia, 2018. 154p. Tese de Doutorado. Instituto de Informática, Universidade Federal de Goiás.

O teste unitário é uma importante atividade para a melhoria da qualidade do software. Ao longo dos anos, inúmeras ferramentas automatizadas foram propostas pela comunidade de pesquisa de teste para melhorar esta atividade. Contudo, a revisão da literatura de testes atual permite observar que esses esforços de pesquisa não têm considerado os aspectos humanos na proposição dessas ferramentas. Observa-se ainda que os praticantes de teste unitário não têm suporte do ferramental existente para resolução de algumas tarefas mentais associadas à atividade. Considerando-se esta lacuna, esta tese descreve uma sequência de estudos realizados com o intuito de entender, caracterizar e propor melhorias no suporte cognitivo provido pelas ferramentas de teste unitário. Tendo em vista a falta de estudos sobre suporte cognitivo para teste de software, empregou-se uma abordagem qualitativa e centrada na perspectiva dos profissionais de teste que atuam no nível unitário. Os resultados revelaram algumas tarefas primárias que requerem suporte cognitivo das ferramentas na prática de revisão de testes unitário, incluindo o monitoramento de tarefas de teste unitário pendentes e executadas e a navegação entre os artefatos relacionados às unidades testadas. Os resultados são resumidos em um *framework* e, com base nisso, é desenvolvida uma agenda de pesquisa como instrumento acionável para a comunidade de teste. As contribuições desta tese incluem sugestões de melhorias práticas para as ferramentas atuais e descrevem novas oportunidades de pesquisa no tema. Além disso, são explicados em detalhes os métodos utilizados nesta pesquisa.

Palavras–chave

Teste Unitário, Teste, Suporte Cognitivo, Aspectos Humanos no Teste de Software, Ferramentas de Teste

Abstract

Paiva Prado, Marllos. **Study, Definition and Proposal of Cognitive Support Resources for the Software Unit Testing**. Goiânia, 2018. 154p. PhD. Thesis. Instituto de Informática, Universidade Federal de Goiás.

Unit testing is an important activity for improving software quality. Over the years, numerous automated tools have been proposed by the testing research community to enhance this activity. However, this thesis' literature review revealed that several research efforts have not considered the human aspects in the proposal of such tools. Also, unit test practitioners are not having the support of the existing tools to solve some mental tasks associated with the activity. Motivated by this gap, this thesis describes a sequence of studies carried out with the purpose of understanding, characterizing and proposing improvements in the cognitive support provided by the test tools, considering a qualitative approach centered on the perspective of test professionals that work at the unit level. The results revealed some primary tasks that require cognitive support of the tools in unit test review practice, including monitoring of pending and executed unit test tasks and navigation between unit testing artifacts. A framework summarizes the results of this study. A research agenda is developed based on the framework and serves as an actionable instrument for the testing community. The contributions of this study include suggestions for practical improvements to current tools and describe new research opportunities in the topic. Also, the methods used in the research are explained in details.

Keywords

Unit Testing, Testing, Cognitive Support, Human Aspects in Software Testing.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

A evolução das ferramentas de teste de software tem papel fundamental na melhoria da atividade de teste. Para manter essa evolução, as pesquisas têm focado na automação como forma de reduzir os esforços e custos envolvidos com a realização da atividade. Paralelamente, o ser humano (profissional de teste) permanece sendo peça chave na condução dos testes na prática, além de ser o principal usuário e beneficiário das ferramentas oriundas dessas pesquisas. Entretanto, os aspectos humanos são, em geral, negligenciados na pesquisa dessas soluções, resultando em deficiências no suporte cognitivo provido pelas ferramentas aos profissionais no teste de software unitário.

Nesta tese, que envolve simultaneamente as áreas de Interação Humano-Computador (IHC) e de Teste de Software, é estudado, definido e proposto um *framework* de demandas de suporte cognitivo a serem atendidas pelas ferramentas no teste de software unitário. Com as pesquisas realizadas entende-se que esta tese constitui um trabalho inédito, uma vez que nenhuma pesquisa análoga foi encontrada na revisão da literatura.

1.1 Contexto e Motivação

A comunidade de pesquisa de engenharia de software tem se empenhado ao longo dos anos na pesquisa de soluções automatizadas capazes de realizar o teste de software. O objetivo geral é desenvolver ferramentas que consigam revelar erros automaticamente e com isso, reduzir os custos envolvidos na atividade — pessoas, capacitação, tempo etc. Neste sentido, observa-se que a pesquisa em automação de testes concentra seus esforços em dois grandes temas: (i) a geração de entradas capazes de exercitar o código e revelar erros onde os desenvolvedores potencialmente cometeram enganos (geração de testes); e (ii) a decisão sobre as saídas geradas pelos programas serem ou não corretas, de acordo com a especificação (oráculos de teste). Apesar de todo o mérito e rigor científico aplicado, esses esforços de pesquisa não têm sido traduzidos em soluções práticas para a indústria, permanecendo como um desafio a ser superado (XIE, 2017; ZENG et al., 2016; TILLMANN; HALLEUX; XIE, 2014; BRIAND, 2012; OSTERWEIL et al., 2008; BOSHERNITSAN; DOONG; SAVOIA, 2006). Enquanto isso, as empresas de desenvol-

vimento continuam dependentes do esforço intelectual de inúmeros profissionais de teste para promoverem uma melhor qualidade dos sistemas sob desenvolvimento.

Por outro lado, observa-se que a JUnit (BECK, 2004) — ferramenta destinada a apoiar a execução de testes unitários — segue como uma das mais importantes ferramentas de teste há décadas. Uma solução criada por pessoas da prática para tornar a vida dos profissionais de teste mais fácil. Uma ferramenta que desde seu surgimento, impactou significativamente a forma como os testes são realizados tanto na indústria quanto na pesquisa. Ao invés de testes totalmente manuais, difíceis de executar e replicar em tempo de execução, ela permitiu que testes modulares e precisos fossem executados, fatorados e reusados (RAMLER; CZECH; SCHLOSSER, 2003). Além disso, é uma solução que incorporou um útil mecanismo de *feedback* — uma barra de status vermelha/verde — o qual simplificou a percepção do usuário sobre o estado corrente da suíte de teste e tornou-se um símbolo da praticidade oferecida pela ferramenta (*"If the bar is green, the code is clean!"* (MEADE, 2009; Virginia Tech, 2013)). Em resumo: uma ferramenta que ao invés de tentar entregar o serviço pronto ao testador, buscou facilitar e potencializar as habilidades desse profissional na execução das tarefas.

O profissional de teste é uma parte essencial do ciclo de melhoria da qualidade do software. Logo, a pesquisa de suporte adequado para as tarefas mentais dos testadores[1] merece a mesma atenção dispensada para a pesquisa de qualquer outro aspecto técnico de automação em teste. Soma-se a isso o fato de que para qualquer novo grau de automação alcançado, o profissional pode precisar adaptar-se e orquestrar as soluções disponíveis, de forma a resolver lacunas nas quais a automação sozinha continua cara ou imatura. Desta maneira, a pesquisa de características nas ferramentas que possam melhorar as habilidades naturais dos profissionais e ajudá-los a aprimorar sua capacidade de descobrir *bugs* mostra-se adequada.

O ímpeto para realização deste trabalho de doutorado adveio das reflexões anteriores. A percepção da dicotomia entre o persistente enfoque da pesquisa na automação de testes e a forma independente com que soluções simples como a JUnit — e demais xUnits — emergiram da prática para se tornarem importantes aliadas do testador, relevou a existência de um potencial problema a ser explorado no domínio das ferramentas de teste. Contudo este problema encontrava-se indefinido e esparso: após revisão de diversas técnicas e ferramentas propostas na literatura com o intuito de facilitar a atividade de teste, somada à troca de informações no assunto com outros pesquisadores e profissionais da área, percebeu-se que o presente trabalho não se resumiria à proposição de uma nova ferramenta para atender uma lacuna já delimitada na área de testes. Ao contrário, o desa-

[1] Neste trabalho, a menos que especificado o contrário, os termos testador e profissional de teste são usados para denominar o profissional que conduz testes em suas atividades, mesmo que não exclusivamente.

fio seria a própria elucidação e proposição de um problema de pesquisa inédito na área: a pesquisa sobre o suporte cognitivo oferecido pelas ferramentas no nível de teste unitário.

O suporte cognitivo corresponde ao auxílio que as ferramentas provêm ao usuário em seu esforço mental de pensar, raciocinar, e criar (WALENSTEIN, 2002). As luzes de LED indicando que as teclas *"Caps Lock" ou "Insert"* do computador estão ligadas são exemplos simples de suporte cognitivo da vida cotidiana. Elas ajudam a contornar a limitação humana em recuperar com facilidade informações da memória de curto prazo sobre o estado atual do dispositivo de entrada de texto quando o usuário divide sua atenção com outras tarefas. Outro exemplo do suporte cognitivo provido por ferramentas no cotidiano é o uso de uma calculadora simples para auxiliar em operações matemáticas extensas: embora este recurso não seja um pré-requisito para realização dos cálculos, a capacidade de registrar, calcular e recuperar valores intermediários com agilidade e precisão auxilia na capacidade de memorização, contribuindo, em geral, para resultados mais rápidos e confiáveis. Desta maneira, a pesquisa do suporte cognitivo na computação corresponde ao estudo do conjunto de propriedades e requisitos que as ferramentas de software devem prover aos usuários, para auxiliá-los na aplicação dos processos cognitivos envolvidos no desempenho de uma atividade. Esta pesquisa deve ser centrada no usuário e em sua perspectiva sobre os problemas enfrentados (ERNST; STOREY; ALLEN, 2005).

Nas ciências exatas é comum seguir-se o paradigma positivista para resolver os problemas de pesquisa. Isso envolve a definição de hipóteses, estabelecidas sobre um arcabouço pré-definido de leis e teorias aceitas com base em modelos geralmente testáveis. Todavia, a pesquisa do suporte cognitivo ocorre dentro daquela que é, possivelmente, a área mais orientada às ciências humanas dentro da computação: a Interação Humano-Computador. Além disso, a cognição humana não se enquadra em um modelo único, imutável ou predizível sendo variável entre os indivíduos e ainda pouco compreendida. Essas características exigem uma perspectiva de pesquisa centrada em aspectos subjetivos dos seres humanos e orientada pelo paradigma sócio-construtivista. Essa diferença de paradigmas traz implicações práticas tanto no modo de se pensar e planejar a própria pesquisa quanto na forma de se considerar e avaliar seus resultados, conforme detalhado ao longo deste trabalho.

Nesse sentido, a tese defendida neste trabalho é a de que a pesquisa de ferramentas de teste deve ser reformulada, considerando uma abordagem centrada nos testadores, para atender as necessidades de suporte cognitivo destes profissionais na condução do teste unitário.

1.2 Objetivos

O objetivo geral desta tese consiste em caracterizar o problema da deficiência de suporte cognitivo provido pelas ferramentas de teste de software no nível unitário e, com base em necessidades identificadas a partir da perspectiva de praticantes da atividade, definir um *framework* que sirva de modelo teórico para a proposição de soluções e novas pesquisas no tema. Para alcançar este objetivo geral, alguns objetivos específicos foram estabelecidos:

1. Identificar o estado da arte de ferramentas e técnicas de visualização aplicadas à compreensão de software; Neste trabalho, considera-se visualização qualquer representação gráfica provida na interface da ferramenta com o intuito de facilitar a abstração e/ou interação do testador com a atividade alvo.
2. Identificar o estado da arte de ferramentas e técnicas de visualização aplicadas ao teste de software;
3. Caracterizar, por meio de trabalhos da literatura, o problema da negligência dos aspectos humanos na proposição de ferramentas de teste de software e definir direções de pesquisa que orientem a pesquisa do suporte cognitivo no teste de software unitário.
4. Definir, por meio de um estudo qualitativo que considere a perspectiva de profissionais de teste, um *framework* de requisitos de suporte cognitivo para ferramentas de teste, considerando a prática de revisão de testes unitário.

1.3 Metodologia

Desde a concepção deste trabalho, uma das principais preocupações foi entender como as ferramentas de teste poderiam contribuir para a melhoria da prática da atividade para o profissional de teste. A metodologia para alcançar o objetivo geral e os objetivos específicos do trabalho (Seção 1.2) está alinhada com essa preocupação e se subdivide em cinco etapas:

1. Na primeira etapa da pesquisa, buscou-se entender, por meio de um mapeamento sistemático inicial, o que existia na literatura em termos de técnicas e ferramentas de visualização aplicadas à compreensão de software. Inicialmente, considerou-se que a melhoria da prática da atividade de teste passaria pela utilização da visualização, considerando a difusão deste recurso em problemas computacionais que envolvem a necessidade de facilitar a abstração dos usuários. Este estudo piloto viabilizou a primeira publicação deste projeto e forneceu um conhecimento inicial sobre a variedade, os tipos e as características comuns das ferramentas e técnicas no

domínio considerado. Os detalhes desse estudo encontram-se descritos em Prado et al. (2013). Os resultados e o conhecimento tácito adquirido com esse mapeamento, viabilizaram o planejamento e a condução da segunda etapa da pesquisa.

2. A segunda etapa da pesquisa consistiu de um novo mapeamento sistemático, desta vez abordando as técnicas e ferramentas de visualização orientadas ao teste de software em geral. Neste estudo, foi possível delinear o perfil das publicações no tema e identificar algumas características relevantes ao estudo, como: (i) a oscilação das publicações ao longo dos anos em três importantes bibliotecas digitais na área de engenharia de software; (ii) a falta de orientação das soluções a um nível específico de teste; (iii) o maior enfoque da literatura em ferramentas e técnicas de visualização orientadas às etapas de planejamento dos testes; (iv) a predominância de representações visuais tradicionais isto é, bidimensionais baseadas em interação via cursor; (v) o escasso apoio das soluções no treinamento dos usuário para o uso das visualizações propostas; (vi) a precariedade dos trabalhos em detalhar como as soluções que empregam visualizações foram definidas e avaliadas. Novas perguntas emergiram após essa revisão da literatura: se existem tantas ferramentas aplicando técnicas de visualização com o intuito de facilitar o teste de software, por que as avaliações das propostas têm sido tão precárias? Por que essas propostas não estão difundidas entre os praticantes da indústria? Se existe um clamor persistente da indústria de desenvolvimento pela necessidade de melhores formas de testar o software, por que técnicas de teste populares há décadas na academia não são incorporadas com facilidade pelos profissionais de teste na indústria (e.g. teste de mutação, critérios de fluxo de dados)? Por que nem mesmo o uso da visualização, como facilitadora no uso dessas técnicas de teste, tem sido suficiente para promover a transferência entre academia e indústria? Existem inúmeros esforços voltados para a automação de teste mas onde os benefícios dessa automação estão sendo observados na prática? Estas questões conduziram à terceira etapa do trabalho: a determinação de como os aspectos humanos estão sendo abordados na pesquisa de ferramentas de teste de software.
3. Para realizar a terceira etapa do projeto, um novo levantamento da literatura foi realizado, desta vez orientado ao entendimento de como o aspecto humano é tratado na pesquisa de teste de software. Esta etapa do projeto foi realizada durante o intercâmbio sanduíche, em colaboração com o grupo de pesquisa The Chisel Group (2015) da *University of Victoria*, dedicado à pesquisa da interação humano–computador na engenharia de software. Neste novo estudo foi constatado que o aspecto humano tem sido não somente negligenciado, mas explicitamente dissociado da pesquisa de teste de software. Além disso, foram identificadas três dimensões de demandas cognitivas a serem suportadas pelas ferramentas de teste no nível unitário, com base

em publicações que reportam a prática desta atividade. Essas dimensões foram sumarizadas em um *framework* inicial para orientar nossos esforços de pesquisa subsequentes. Este estudo resultou na segunda publicação deste projeto (PRADO et al., 2015) dentro da trilha principal da 26ª edição do *International Symposium on Software Reliability Engineering (ISSRE)*, em uma categoria especialmente dedicada a artigos que representam ideias inovadoras e provocativas na área *(WAP)*.

4. A partir dos resultados anteriores, iniciou-se a quarta fase do projeto para entender como as dimensões cognitivas do *framework* inicial estavam sendo abordadas nas ferramentas de teste unitário levantadas na segunda etapa da pesquisa. Das sessenta e quatro ferramentas selecionadas, doze aplicavam-se ao teste unitário. Este estudo permitiu identificar diversas lacunas e oportunidades de melhorias nos trabalhos avaliados, relacionadas ao suporte cognitivo. Também foi possível observar que a falta de envolvimento do testador foi uma característica comum entre os trabalhos com problemas de suporte cognitivo. Este estudo possibilitou a terceira publicação oriunda deste trabalho de doutorado (PRADO; VINCENZI, 2016).
5. Baseado nos resultados anteriores, constatou-se a necessidade de um novo estudo (quinta etapa), considerando uma abordagem de pesquisa centrada nos profissionais de teste com experiência em teste unitário. Para tanto, um questionário online composto de vinte e quatro questões (doze abertas e doze objetivas) foi elaborado e respondido por cinquenta e oito voluntários de diversas localidades e com variados níveis de experiência nesse nível de teste. O método *Qualitative Content Analysis* foi aplicado na sua forma indutiva para análise das respostas fornecidas pelos voluntários e, com base nas categorias que emergiram da análise, um *framework* de tarefas de teste unitário que demandam suporte cognitivo foi derivado. Com base no *framework* gerado, uma agenda de pesquisa foi proposta. Esta agenda de pesquisa serve como um instrumento acionável para a comunidade de teste de software evoluir o suporte cognitivo oferecido pelas ferramentas de teste unitário atuais e propor novos estudos no tema. O estudo realizado na quinta etapa resultou em um novo artigo, o qual foi submetido e aceito no *Journal of Systems and Software (JSS)* (PRADO; VINCENZI, 2018).

1.4 Contribuições e Experiências

A seguir, são listadas as contribuições oriundas deste trabalho:

- A caracterização de um problema de pesquisa inédito na área: a pesquisa do suporte cognitivo das ferramentas de teste no nível unitário. Esta caracterização habilita que a pesquisa seja orientada a apoiar as tarefas mentais do profissional de teste unitário

nas diversas práticas da atividade. Neste estudo, trata-se em específico do suporte cognitivo para a prática de revisão dos testes unitários.

- Um *framework* de tarefas que demandam suporte cognitivo das ferramentas, para a prática de revisão de teste unitário. O *framework* é o resultado de um estudo qualitativo indutivo, centrado na perspectiva dos próprios profissionais de teste. O *framework* descreve necessidades a serem atendidas na evolução das ferramentas de teste unitário.
- Uma agenda de pesquisa. A agenda de pesquisa constitui um instrumento acionável desta pesquisa. Ela serve para orientar a comunidade de testes sobre como os resultados do *framework* podem ser utilizados para informar novas pesquisas e melhorar o suporte cognitivo das ferramentas existentes.
- Um arcabouço de artefatos (planejamento, ferramentas, detalhamento da execução e resultados) referentes ao mapeamento sistemático realizado para identificar técnicas e ferramentas de visualização que apoiam a atividade de teste de software (PRADO; VINCENZI; NASCIMENTO, 2014). Este arcabouço habilita a verificação, replicação e expansão do mapeamento por outros pesquisadores.
- A caracterização de como a literatura tem dissociado os fatores humanos da pesquisa de ferramentas de teste de software. Por meio desta caracterização tornamos explícita a necessidade de transferência do enfoque tradicional de pesquisa da automatização para uma abordagem centrada no usuário e sua perspectiva.
- A definição de um *framework* inicial para orientar a pesquisa de suporte cognitivo no teste unitário. Esse *framework* agrega três dimensões gerais de demandas cognitivas, definidas com base em necessidades que se encontravam difusamente reportadas na literatura. Além de ajudar no desdobramento desta pesquisa, esse *framework* pode ser reaproveitado em novos estudos que venham a considerar o suporte cognitivo em outras práticas da atividade de teste unitário.
- A caracterização de como as ferramentas que aplicam visualização no teste unitário (propostas na literatura) têm atendido às demandas de suporte cognitivo representadas no *framework* inicial. Essa caracterização permitiu verificar deficiências e oportunidades de melhoria prominentes do suporte cognitivo provido por essas ferramentas.

Este trabalho de doutorado foi parcialmente conduzido no exterior (Canadá) sob supervisão da Dra. Margaret-Anne Storey e em colaboração com sua equipe de pesquisa. A experiência contribuiu bastante com o enriquecimento e desenvolvimento deste trabalho. Por meio do intercâmbio, foi possível dialogar e trocar experiências entre a área de pesquisa original do autor deste trabalho (Teste de Software) e a área de pesquisa do grupo estrangeiro (Interação Humano-Computador). A troca de experiências permitiu, por exemplo: ampliar o entendimento sobre como considerar adequadamente o papel central

do ser humano no desenvolvimento desta pesquisa; aprofundar o conhecimento sobre a metodologia qualitativa de pesquisa; melhorar as habilidades de escrita científica do autor na língua inglesa; expor o trabalho em desenvolvimento a outros pesquisadores da universidade de destino *(University of Victoria)* e da rede de colaboradores da supervisora estrangeira. Embora esse tenha sido nosso primeiro contato com esse grupo de pesquisa, a obtenção dos primeiros resultados logo nos seis primeiros meses de trabalho permitiu que os laços fossem estreitados e parcerias futuras pudessem ser estabelecidas.

1.5 Organização do Trabalho

Neste capítulo foi apresentado o contexto no qual este trabalho se insere, os objetivos, a metodologia e as contribuições geradas. No Capítulo 2 são descritos os fundamentos, níveis e etapas de teste, assim como as principais técnicas e critérios de teste existentes. No Capítulo 3 são apresentados os principais fundamentos sobre os aspectos cognitivos na área da Interação Humano-Computador. No Capítulo 4 são descritos o planejamento, condução, resultados e análise do mapeamento sistemático realizado a respeito das ferramentas e técnicas de visualização que apoiam a atividade de teste. No Capítulo 5 discute-se o problema da negligência dos fatores humanos na pesquisa de ferramentas de teste, os problemas cognitivos identificados na atividade de teste unitário e a forma como essas necessidades foram compiladas para orientar os passos seguintes da pesquisa. No Capítulo 6 são analisadas as deficiências e oportunidades de melhoria do suporte cognitivo provido pelas ferramentas que utilizam visualização no apoio ao teste unitário. No Capítulo 7 são descritos os detalhes do estudo qualitativo que resultou no *framework* de suporte cognitivo para a prática de revisão de testes unitários, proposto nesta tese. Adicionalmente, uma agenda de pesquisa foi estabelecida para mostrar como o *framework* pode ser utilizado para informar novos estudos e melhorar o suporte cognitivo das ferramentas existentes. As conclusões e trabalhos futuros são pontuados no Capítulo 8. Por fim, são relacionadas as bibliografias utilizadas para escrita dessa tese.

CAPÍTULO 2

Teste de Software

2.1 Considerações Iniciais

O teste de software é uma das principais atividades da garantia de qualidade de software. Ele pode ser aplicado ao longo de todo o ciclo de desenvolvimento e sobre os mais diversos tipos de artefatos de software—especificação, modelos, códigos, dentre outros. O objetivo da atividade de teste é revelar a presença de erros nos artefatos sob teste (MYERS, 1978), contribuindo, desta forma, para a melhoria da qualidade geral do software ao longo de seu ciclo de vida.

Neste capítulo são apresentados os termos e conceitos de teste de software que fundamentam o restante do trabalho. Para tanto, definições, técnicas e critérios de teste são explicados, com base na literatura atual de teste.

2.2 Conceitos e Definições

O estudo do teste de software requer a definição de algumas palavras-chaves. O objetivo é facilitar a compreensão desses termos em cada contexto que se aplicam. Neste trabalho, adota-se a terminologia apresentada a seguir [1], baseada no padrão *IEEE* 24765-2010 (IEEE, 2010):

1. Produto/Artefato sob Teste ou Sistema sob Teste (*Product Under Test*, *System Under Test ou SUT)*: sistema, subsistema ou componente de software sob teste;
2. Dado de Teste (*Test Data*: Dado de entrada fornecido à interface pública do artefato sob teste durante a execução do caso de teste;
3. Caso de Teste (*Test Case)*: Conjunto de dados de teste, condições de execução e resultados esperados de um produto sob teste para atingir um objetivo específico de teste isto é, verificar a conformidade com um determinado requisito;

[1] Com exceção de trechos ou referências a trabalhos de terceiros, no qual será mantido o significado do trabalho original

4. Conjunto de Teste (*Test Suite)*: Coleção de um ou mais casos de teste;
5. Critério de Teste (*Test Criteria)*: Define os requisitos que o conjunto de teste deve atender para passar no teste.
6. Engano (*Mistake)*: Ação humana que ocasiona um resultado incorreto (e.g. ação incorreta realizada pelo programador);
7. Defeito (*Fault)*: Passo, processo ou definição de dados incorreto (e.g. instrução ou comando incorreto);
8. Erro (*Error*): Diferença entre valor obtido e valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução do artefato sob teste; e
9. Falha (*Failure)*: Produção de uma saída incorreta com relação à especificação.

Além da terminologia apresentada, este trabalho segue as considerações de Myers et al. (2004) em que: (i) um bom caso de teste é aquele que tem grande chance de revelar falhas; e (ii) um teste bem sucedido é aquele que descobre um defeito inédito no artefato sob teste.

A definição e aplicação dos casos de teste se dão de acordo com o nível de granularidade dos artefatos sob teste. Em geral, três níveis são considerados: teste unitário, teste de integração e teste de sistema (PRESSMAN; MAXIM, 2014).

2.2.1 Teste Unitário

No teste unitário (ou teste de unidade) os menores componentes coesos de um sistema são considerados para a finalidade de teste. Em geral, esses componentes são as funções, procedimentos, métodos ou classes, dependendo do paradigma e da linguagem consideradas. A finalidade desse nível de teste é encontrar defeitos de lógica e implementação nesses elementos.

Na literatura, diferentes autores têm entendimentos distintos acerca do que seja uma unidade de software. O padrão *IEEE* 610.12-1990 (IEEE, 1990) define que uma unidade é um componente de software que não pode ser subdividido. Nesse sentido, no paradigma procedimental o procedimento ou função definem a menor unidade a ser testada. Na orientação a objetos, a unidade pode corresponder a uma classe (PERRY; KAISER, 1990; BINDER, 1999) ou método (VINCENZI, 2004). Os autores que definem a classe como menor unidade a ser testada justificam que essa é a menor unidade funcional no paradigma orientado a objetos—ou seja, que não depende de outras para executar. Por outro lado, uma classe é formada por um conjunto de métodos e atributos que se integram para executar um conjunto de operações. Desta forma, a interação entre os métodos é passível de ser testada—teste inter-método (HARROLD; ROTHERMEL, 1994)—o que justificaria escolher o método como menor unidade sob teste (VINCENZI, 2004; COLANZI, 1999). Todavia, na prática, essa definição não é rígida e o conceito de

unidade pode ainda considerar uma combinação entre dois ou mais desses elementos (ver seção 7.6.1).

Independentemente de qual seja o conceito escolhido para se definir a unidade a ser testada, as unidades do software podem depender ou estar interligadas a outros componentes do sistema que não estejam disponíveis durante o teste unitário. Para tanto, o teste unitário utiliza *drivers* e *stub* como artifícios que ajudam a contornar essa situação. De acordo com Vincenzi (2004) um *driver* é um elemento que coordena o teste da unidade sob teste. Ele recebe os dados de teste, encaminha os dados na forma de parâmetro para a unidade sob teste, captura os resultados gerados pela unidade testada e os apresenta ao testador. O *stub* é um elemento que simula o comportamento de um componente a ser usado pela unidade sob teste durante os testes. Em geral, a simulação executada por um *stub* envolve o mínimo de computação e manipulação de dados. O objetivo dos *drivers* e *stubs* é minimizar os custos de ter que se esperar que as dependências de uma unidade estejam prontas/disponíveis para o teste. Isso contribui para a condução dos testes desde as etapas iniciais do ciclo de desenvolvimento.

No paradigma de programação orientado a objetos, caso se considere o método como menor unidade a ser testada, a classe funciona como um *driver* para os seus próprios métodos. Nesse sentido, a classe recebe as entradas, as encaminha ao método, coleta os resultados gerados pelo método e os apresenta ao usuário. Além disso, o método depende da existência da classe para ser executado e, portanto, testado.

2.2.2 Teste de Integração

O teste unitário é insuficiente para detectar determinados tipos de erros durante o teste de um sistema. Considerando-se que as unidades coexistem e colaboram entre si para desempenho das funções de um sistema, erros podem surgir durante essa integração. (DELAMARO, 1997) cita como exemplos de situações que podem causar erros dessa natureza: a influência inesperada de outras unidades do sistema cuja integração não foi antevista; problemas em estruturas de dados globais; a geração de resultados inesperados durante a combinação de dois ou mais métodos/funções.

O teste de integração, embora complementar, pode afetar diretamente o teste unitário e o custo da atividade de teste como um todo. Erros identificados durante o teste de integração podem acarretar em correções de unidades previamente testadas, implicando na necessidade de um novo ciclo de teste no nível unitário e de integração. Desta forma, o teste de integração também deve ser aplicado desde os ciclos iniciais de desenvolvimento, paralelamente ao teste unitário. Tal medida ajuda a mitigar a elevação inesperada de custos em fases avançadas do desenvolvimento. Além disso, o teste de

integração pode contribuir para a melhoria da qualidade geral dos próprios teste unitários, na medida que gera a oportunidade de revisão dos mesmo.

2.2.3 Teste de Sistema

O teste de sistema (ou teste fim a fim) é o teste aplicado sobre o software como um todo, ou seja, considerando-se todas as suas unidades integradas e funcionando. Neste tipo de teste, procuram-se por erros associados à integração do software sob teste com recursos aos quais esteja vinculado por exemplo, banco de dados, serviços ou plataforma de hardware. O objetivo desse tipo de teste é verificar se o funcionamento global do software atende às expectativas definidas na especificação do sistema. Desta forma, requisitos não funcionais como desempenho, estresse e segurança também são observados nesse nível de teste.

2.3 Etapas, Técnicas e Critérios de Teste

Além dos níveis de teste, a atividade de teste organiza-se em etapas. Em geral, essas etapas subdividem-se em: (i) planejamento, (ii) projeto de casos de teste, (iii) execução e (iv) análise dos resultados ((MYERS et al., 2004); (BEIZER, 1990); (PRESSMAN; MAXIM, 2014)).

Idealmente, para garantir a ausência de defeitos em um programa, o teste de todas as combinações de valores possíveis do domínio de entrada e saída é necessário (teste exaustivo). Todavia, esse tipo de teste é reconhecido como impraticável (FABBRI; VINCENZI; MALDONADO, 2007). Uma forma de se contornar essa situação é a utilização de técnicas de teste. As técnicas de teste definem o tipo de informação a ser utilizada na escolha de casos de teste. O intuito de cada técnica é aumentar as chances de se escolher casos de teste mais propícios a fazerem o software falhar, seguindo-se algum critério. Basicamente, as técnicas se dividem em três categorias: funcional, estrutural e baseada em defeitos.

De acordo com (ZHU; HALL; MAY, 1997), um critério de teste estabelece os requisitos de teste a serem verificados para avaliar a qualidade dos testes. Os requisitos de teste são propriedades derivadas do artefato sob teste. Esses requisitos podem estar associados ao domínio de entrada, à estrutura interna ou a um modelo representativo do artefato sob teste. Diz-se que um critério de teste é satisfeito quando os requisitos de teste definidos pelo critério são atendidos pelo conjunto de casos de teste (conjunto de teste). Um exemplo de critério seria definir que todas as instruções de um programa devem ser testadas. Neste exemplo, cada instrução seria um requisito de teste e a satisfação do critério se daria pelo exercício de todas as instruções do programa.

Um critério de teste pode ser entendido tanto como um critério de seleção quanto de adequação, havendo correspondência entre ambas perspectivas. Entende-se como um critério de seleção aquele que define um procedimento para selecionar o conjunto de teste. O critério de adequação por outro lado, define um procedimento para avaliar o conjunto de teste (SOUZA, 1996). A correspondência pode ser entendida da seguinte maneira (MALDONADO, 1991). :

> Dado o critério de adequação A, é possível estabelecer um critério de seleção S que define: "selecione um conjunto de teste que satisfaça o critério A".
>
> De maneira análoga:
>
> Dado o critério de seleção S, é possível estabelecer um critério de adequação A que define: "Um conjunto de teste é adequado se ele é gerado pelo procedimento S".

2.3.1 Técnica de Teste Funcional

Na técnica de teste funcional—também denominada teste caixa-preta ou *black-box testing*—o tipo de informação utilizada na escolha dos casos de teste é somente a especificação (BEIZER, 1990). O objetivo do teste funcional é verificar se a saída gerada pela execução da entrada fornecida corresponde ao resultado especificado. Desta forma, a maneira como o resultado é gerado não é relevante para a verificação da satisfação dos testes.

Algumas fraquezas na aplicação deste critério são: a dependência de uma definição e interpretação correta da especificação; a impossibilidade de quantificar o teste — por exemplo, no caso de um programa, o quanto cada região do código foi exercitada; a impossibilidade de testar explicitamente partes críticas da estrutura do componente testado (VINCENZI, 2004).

Os critérios da técnica funcional, em geral, subdividem ou delimitam os valores do domínio de entrada/saída visando a derivação de casos de teste com maiores chances de relevar defeitos (PRESSMAN; MAXIM, 2014). Dois critérios de teste funcionais são os critérios "Particionamento de Equivalência" e "Análise do Valor Limite".

Particionamento de Equivalência

No critério Particionamento de Equivalência, utiliza-se a especificação do componente de software a ser testado para dividir o domínio de entrada em intervalos—denominados classes de equivalência.

As classes de equivalência podem ser classificadas em válidas ou inválidas dependendo dos valores que ela representa na especificação—válidos ou inválidos. Cada

classe de equivalência representa um conjunto de valores que executa as mesmas funções do componente de software testado. Isso significa que independentemente do valor escolhido dentro de uma dada classe de equivalência, o componente testado deve se comportar de forma semelhante (MYERS et al., 2004).

A aplicação do critério Particionamento de Equivalência se dá em quatro etapas:

1. Identificação das classes de equivalência.
2. Criação de ao menos um caso de teste para cada classe de equivalência válida.
3. Criação de ao menos um caso de teste para cada classe de equivalência inválida.
4. Comparar o resultado obtido pela execução de cada caso de teste com seu respectivo resultado esperado.

O critério Particionamento de Equivalência é satisfeito quando pelo menos um caso de teste é exercitado para cada classe de equivalência estabelecida.

Análise do Valor Limite

O objetivo do critério Análise do Valor Limite é avaliar os valores no limite de cada classe de equivalência definida. Sendo assim, a verificação desse critério implica em testar o valor mínimo—bem como o seu antecessor imediato—e o valor máximo—bem como o seu sucessor imediato—nos intervalos representados por cada classe de equivalência. A motivação para o uso deste critério é que os erros geralmente acontecem nos limites dos domínios de entrada (PRESSMAN; MAXIM, 2014). Logo, o critério Análise do Valor Limite ajuda a verificar se as operações que funcionam para as classe de equivalência definidas, também mantêm um comportamento adequado nos limites desses intervalos. Esse critério pode auxiliar, por exemplo, na descoberta de erros oriundos de enganos cometidos pelo programador durante a definição de condições de parada para laços de repetição.

2.3.2 Técnica de Teste Estrutural

O teste estrutural, ou teste caixa-branca, difere-se do teste funcional por estabelecer os requisitos de teste baseados na estrutura do programa.

Os critérios da técnica estrutural, em geral, avaliam o exercício dos requisitos de teste sobre estruturas de grafos. No caso do teste de programas, o Grafo de Fluxo de Controle (GFC) é uma dessas possibilidades (RAPPS; WEYUKER, 1982). O GFC é um grafo direcionado no qual os nós representam blocos sequenciais de instruções do código. As arestas representam instruções de desvios entre os blocos de comandos sequenciais. Desta forma, o exercício de um nó representa a execução sequencial de todas as instruções que ele mapeia no código. De forma análoga, o exercício de uma aresta

significa a execução do desvio que ela representa. Em um GFC, um caminho é formado por uma sequência finita de nós $(n_1, n_2, ..., n_k)$, para $k \geq 2$, tal que existam arestas de n_1 para $n_i + 1$ e $(1 \leq i < k)$. Uma das vantagens do teste estrutural é que essa técnica permite verificar os caminhos exercitados na estrutura do componente de software testado. Esta é uma característica importante ao considerar-se que um componente de software que possui caminhos não exercitados tem a sua confiabilidade comprometida (RAPPS; WEYUKER, 1982). Sendo assim, a técnica estrutural é vista como complementar ao teste funcional (PRESSMAN; MAXIM, 2014). Uma das desvantagens associadas ao teste estrutural é que podem haver caminhos não executáveis—isto é, caminhos impossíveis de serem exercitados, independentemente do caso de teste fornecido[2]; (HOWDEN, 1987) (RAPPS; WEYUKER, 1985). A tarefa de descobrir caminhos não executáveis é difícil de ser totalmente automatizada, uma vez que depende de habilidade de inferência e raciocínio humano (WEYUKER, 1990).

A seguir, duas categorias de critérios de teste estruturais para programa são apresentadas: critérios baseados em fluxo de controle e critérios baseados em fluxo de dados.

Critérios Baseados em Fluxo de Controle

Neste tipo de critério, os requisitos de teste são derivados de informações do fluxo de controle do programa testado. Alguns critérios de fluxo de controle difundidos na literatura de teste são:

- Critério Todos-Nós: Neste critério, um conjunto de teste T é adequado ao critério Todos-Nós para o programa P se todos os nós do GFC de P são exercitados por T. Esse critério pode ser utilizado para resolver o problema de cobrir todas as instruções de um programa. Entretanto, é relativamente fácil construir um conjunto de teste que atinja a adequação desse critério. Essa característica acaba permitindo a formação de conjuntos de teste pouco eficazes isto é, incapazes de revelar até mesmo alguns tipos de erros mais simples (MYERS et al., 2004). Um exemplo trivial seria um caso de teste que exercita um determinado subconjunto de nós, mesmo quando esse subconjunto representa uma condição de execução incorreta.
- Critério Todas-Arestas: O objetivo do critério Todas-Aresta é garantir que todos os desvios do programa sob teste sejam exercitados ao menos uma vez. Desta forma, diz-se que um conjunto de teste T é adequado ao critério. Todas-Arestas para um programa P se todas as arestas do GFC de P são exercitadas por T.

[2]Caminhos não executáveis ocorrem devido a combinações contraditórias de condições lógicas—as quais podem ocorrer naturalmente ao longo do fluxo de execução—e que não viabilizam o exercício de todos os nós no caminho considerado.(HOWDEN, 1987)

- Critério Todos-Caminhos: O critério Todos-Caminhos busca garantir que todos os caminhos possíveis do GFC de um programa sejam exercitados ao menos uma vez. Diz-se que um conjunto de teste T é adequado ao critério Todos-Caminhos para um programa P se ele cobre todos os caminhos possíveis do GFC de P. Apesar de ser considerado um critério ideal, o critério Todos-Caminhos é infactível para a atividade rotineira de testes. Uma das razões é que a quantidade de caminhos possíveis pode ser grande ou infinita mesmo para GFC's considerados simples (MYERS et al., 2004).

Critérios Baseados em Fluxo de Dados

Neste tipo de critério, os requisitos de teste são derivados a partir das informações dos dados do programa. Em Rapps e Weyuker (1982) e Rapps e Weyuker (1985) é proposto um conjunto de critérios de fluxo de dados que avaliam se as definições das variáveis e os seus respectivos usos são feitos adequadamente ao longo da execução do programa. Para a verificação desse critério um Grafo Definição Uso (GDU)—uma extensão do GFC—é utilizado. Além da estrutura do GFC, o GDU contém anotações referentes à definição e/ou uso das variáveis existentes em cada nó. Uma definição de uma variável ocorre sempre que esta variável recebe um valor. O uso *de* uma variável pode ocorrer de duas maneiras distintas: (i) uso predicativo (*p-uso*) que se refere ao uso da variável na avaliação de uma condição; (ii) uso computacional (*c-uso*) que se refere ao uso da variável para computar um valor. Cada relação de definição/uso para uma dada variável no GDU é denominada *associação*.

De acordo com Rapps e Weyuker (1982) um caminho livre de definição para uma variável v de j até k é um caminho $(j, n_1, ..., n_m, k)$ com $m \geq 1$ no qual v é definido em j e v não é redefinido de n_1 até n_k—incluindo estes (RAPPS; WEYUKER, 1982).

Os principais critérios de fluxo de dados definidos por (RAPPS; WEYUKER, 1982) são:

- Critério Todas-Definições *(all-defs)*: A satisfação desse critério requer que cada definição de variável no GDU seja exercitada ao menos uma vez, independentemente se o uso é do tipo *c-uso* ou *p-uso*.
- Critério Todos-Usos *(all-uses)*: A satisfação desse critério requer que cada associação do tipo definição/*p-uso* e definição/*c-uso*, para toda variável no GDU, sejam exercitadas por meio de pelo menos um caminho livre de definição.
- Critério Todos-*Du*-Caminhos *(all-du-paths)*: A satisfação desse critério requer que cada associação do tipo definição/*p-uso* e definição/*c-uso*, para toda variável no GDU, sejam exercitadas por meio de todos os caminhos livres de definição e livres de laço que cubram essa associação.

Outras variações menos completas do critério Todos-Usos, também propostas por Rapps e Weyuker (1982) são os critérios: Todos-P-Usos, Todos-P-Usos/Alguns-C-Usos e Todos-C-Usos/Alguns-P-Usos.

Maldonado (1991) propôs ainda a família de critérios de fluxo de dados Potenciais-Usos. Os principais critérios da família são: Todos-Potenciais-DU-Caminhos, Todos-Potenciais-Usos/DU e Todos-Potenciais-Usos. A satisfação do critério Todos-Potenciais-Usos requer que uma variável definida em um nó k exercite ao menos um caminho livre de definição para todo nó e arco possível de ser alcançado a partir de k (MALDONADO, 1991). Diferentemente dos critérios definidos por Rapps e Weyuker (1982), nos critérios da família Potenciais-Uso, dada a definição de uma variável, não é necessário que ocorra um uso real dessa variável em um determinado nó para que essa associação seja testada. O potencial uso dessa variável no nó considerado, por meio de um caminho livre de definição, já é suficiente para o teste. Denomina-se *potencial-associação* a associação do tipo definição/potencial-uso.

2.3.3 Relação de Inclusão Entre Critérios Estruturais

Rapps e Weyuker (1985) definem a relação de inclusão entre critérios da técnica estrutural da seguinte forma: "Dados dois critérios C_1 e C_2, diz-se que C_1 inclui C_2 se para todo conjunto de teste T_1 C_1-adequado, T_1 é C_2-adequado e existe um T_2 C_2-adequado que não é C_1-adequado; C_1 e C_2 são equivalentes se para qualquer T C_1-adequado, T é C_2-adequado e vice-versa". A Figura 2.1 ilustra a relação de inclusão entre alguns critérios estruturais difundidos na literatura. Quanto mais alto está um critério na relação de hierarquia, mais casos de teste são exigidos para satisfazê-lo—o que consequentemente aumenta o custo de teste.

2.4 Técnica de Teste Baseada em Defeitos

No teste baseado em defeitos—denominado também de teste baseado em mutação ou teste baseado em defeitos— os requisitos de teste são derivados de informações sobre os tipos de erros comumente cometidos durante o processo de desenvolvimento e sobre os tipos de erros que se deseja revelar (DEMILLO; LIPTON; SAYWARD, 1978). Nesta técnica, busca-se inserir, propositalmente, pequenas alterações no código do programa sob teste. O objetivo da técnica baseada em defeitos é utilizar defeitos artificiais para promover a geração de casos de teste capazes de revelar os erros—tanto os artificiais quanto os naturais. A análise de mutantes é um dos principais critérios representantes da técnica baseada em erros (DEMILLO; LIPTON; SAYWARD, 1978).

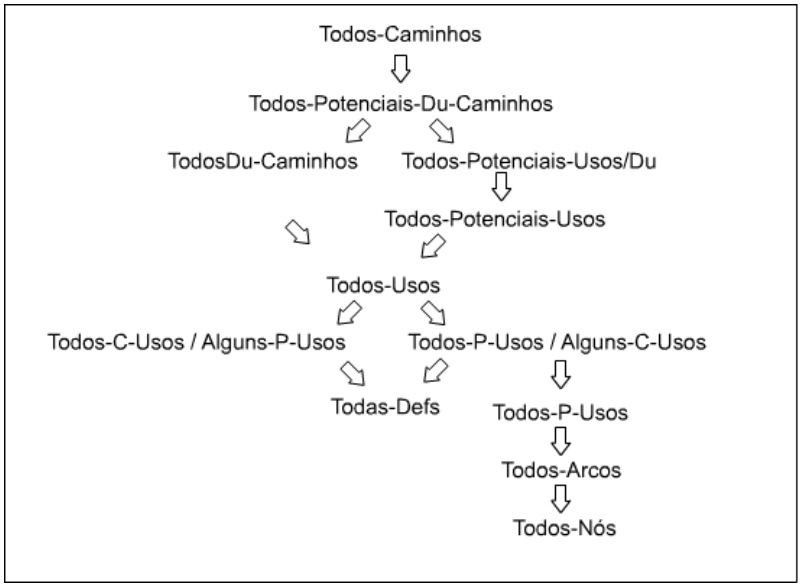

Figura 2.1: Ordem parcial entre os critérios Potenciais-Usos básicos, Critérios de fluxo de dados e de fluxo de controle (MALDONADO, 1991).

2.4.1 Critério Análise de Mutantes

O critério Análise de Mutantes faz a inserção de pequenas alterações sintáticas no programa sob teste por meio de operadores de mutação. Os operadores de mutação são regras aplicáveis sobre um programa, definidas com base em erros sintáticos que são comuns de serem cometidos. Os operadores de mutação definem como as alterações devem modificar o programa original. Os programas gerados pela aplicação dos operadores de mutação são denominados mutantes.

Na Análise de Mutantes o conjunto de teste a ser definido tem como objetivo demonstrar que o programa sob teste não apresenta os mesmos erros que seus respectivos mutantes.

O critério Análise de Mutantes baseia-se em duas hipóteses:

- *A hipótese do programador competente:* Essa hipótese assume que um programa escrito por um programador competente é um programa próximo de sua correção. Assumindo-se essa hipótese como verdadeira, a existência de defeitos nos programas ocorre devido a pequenos desvios sintáticos. Esses desvios embora não cheguem a causar erros sintáticos (e.g. erros em tempo de compilação) alteram a semântica do programa e conduzem o mesmo a um comportamento incorreto isto é, não especificado. Desta forma, a Análise de Mutantes revela os erros identificando os tipos de erros mais comuns; realizando pequenas transformações no programa sob teste; e promovendo a construção de um conjunto de teste pelo testador capaz de mostrar que as transformações aplicadas levam ao funcionamento incorreto dos mutantes (AGRAWAL, 1989).

- *A hipótese do efeito do acoplamento:* Essa hipótese assume que a revelação de erros complexos está relacionada à revelação de erros simples. Essa hipótese foi confirmada empiricamente em alguns estudos da literatura (OFFUTT; ROTHERMEL; ZAPF, 1993a), (ACREE et al., 1979).

Aplicação do Critério Análise de Mutantes

A aplicação do critério Análise de Mutantes sobre um programa P, considerando-se um conjunto de teste T é realizada em quatro etapas principais (VINCENZI, 1998):

1. **Geração do Conjunto de mutantes** M: O conjunto de mutantes M é gerados por meio da aplicação dos operadores de mutação sobre o programa P.
2. **Execução do programa** P: O programa P é executado com o conjunto de teste T e é verificado se o resultado gerado corresponde ao esperado. Caso alguma saída incorreta seja identificada, o defeito deve ser localizado, corrigido e o processo de teste deve retornar à primeira etapa. Em geral, cabe ao testador fazer o papel de oráculo, ou seja, verificar se a saída gerada corresponde à esperada (WEYUKER, 1982).
3. **Execução do conjunto de mutantes** M: O conjunto de teste T é aplicado a cada mutante m_i do conjunto M. Um mutante m_i é considerado morto quando é revelada a diferença de seu comportamento em relação a P, ou seja, quando um caso de teste t_i pertencente a T consegue revelar em m_i um resultado diferente do resultado obtido quando t_i é executado em P. Quando m_i permanece vivo, duas possibilidades ocorrem: (i) o conjunto de teste T não é capaz de revelar a mutação e precisa ser melhorado; ou (ii) o mutante (m_i) é equivalente ao programa P.
   Após a execução de T sobre M, o escore de mutação pode ser calculado. O escore de mutação informa a qualidade do conjunto T adotado. A fórmula para calcular o escore de mutação é:

$$ms(P,T) = \frac{DM(P,T)}{M(P) - EM(P)} \tag{2-1}$$

   Na qual,

   - $ms(P,T)$: escore da mutação;
   - $DM(P,T)$: número de mutantes mortos por T;
   - $M(P)$: número total de mutantes gerados;
   - $EM(P)$: número de mutantes equivalentes identificados.

   O escore de mutação é sempre um valor no intervalo [0,1]. Quanto mais próximo de 1 for o escore, mais adequado é o conjunto T.

4. **Análise dos mutantes vivos em M**:
   Esta etapa requer a capacidade de análise humana para ser realizada. Dependendo do escore de mutação alcançado, o testador decide se deve ou não continuar os testes. Em caso positivo, o testador deve determinar se cada mutante vivo m_k é ou não equivalente ao programa P. O problema de determinar se dois programas mutantes são equivalente é indecidível (BUDD, 1981). Logo, se faz necessária a utilização de heurísticas conforme em Simão e Maldonado (2000). Quando um mutante é identificado como equivalente ele deve ser descartado. Quando o conjunto de teste não é suficiente para distinguir os mutantes não-equivalentes do programa P o testador deve retornar à etapa 2, gerar um novo conjunto de teste e repetir o processo até que se alcance o escore de mutação desejado.

O custo de aplicação da Análise de Mutantes é o maior entrave para aplicação desse critério na prática. Em geral, um grande número de mutantes é gerado mesmo para sistemas pequenos. Isso se deve ao grande número de operadores de mutação existentes—cada um com a possibilidade de ser aplicado em inúmeros comandos distintos dentro de um mesmo programa. A execução do conjunto de teste sobre cada mutante, já eleva da carga de processamento computacional requerido. Além disso, existe o problema de determinar a equivalência dos mutantes vivos, em relação ao programa original—o que implica em um esforço intelectual humano e, consequentemente, tempo.

Alguns trabalhos ((ACREE et al., 1979); (MATHUR, 1991); (OFFUTT; ROTHERMEL; ZAPF, 1993b); (OFFUTT et al., 1996); (WONG et al., 1997); (BARBOSA; MALDONADO; VINCENZI, 2001)) propuseram formas de se tentar reduzir o conjunto de mutantes a serem gerados, mantendo-se a capacidade de revelar as discrepâncias do conjunto total em relação ao programa original. Desta forma, o conjunto de teste necessário para "matar" o subconjunto reduzido de mutantes deverá ser tão eficaz quanto aquele necessário para "matar" os mutantes do conjunto original. Todavia, essas soluções para redução do conjunto de operadores podem ser sensíveis a determinados contextos. Além disso, o problema da determinação de mutantes equivalentes continua existindo.

2.5 Considerações Finais

Neste capítulo, foram explicados os conceitos de teste de software adotados na escrita deste trabalho. Inicialmente, estabeleceu-se a terminologia empregada. Em seguida, foram descritos os três principais níveis de granularidade para definição de casos de teste: teste unitário, teste de integração e teste de sistema. Cada nível de teste representa uma forma diferente de se caracterizar as partes do software a serem testadas. Além dos níveis, foram abordadas as principais técnicas e critérios de teste. As técnicas referem-se

ao tipo de informação a ser considerada na derivação dos casos de teste—por exemplo, documentação, estrutura do artefato sob teste, principais tipos de erros cometidos pelos programadores. As principais técnicas consideradas no teste de programa são: técnica funcional, estrutural e baseada em erros. Por fim, foram descritos alguns dos principais critérios propostos na literatura para cada técnica de teste abordada.

CAPÍTULO 3

Aspectos Cognitivos

3.1 Considerações Iniciais

A cognição é um assunto interdisciplinar e seu estudo, em geral, é realizado dentro das ciências cognitivas. Muitas definições sobre o que é cognição podem ser encontradas na literatura (NEISSER, 1967; NORMAN, 2002; APA - American Psychological Association, 2017). Uma ideia comum entre essas definições é a de que a cognição corresponde aos processos mentais aplicados para construir e usar o conhecimento.

Neste capítulo são apresentados os fundamentos e terminologia sobre cognição aplicados neste trabalho. O tema cognição pode ser abordado sob diversas perspectivas dentro das ciências cognitivas (linguística, neurociência, antropologia, dentre outros). Nesta tese, a perspectiva considerada é a da Interação Homem-Computador, considerando-se a pertinência e adequação desta perspectiva ao escopo deste trabalho.

3.2 A Cognição e os Processos Cognitivos

Norman (1994) classificou a cognição em dois tipos genéricos distintos: *experiencial* e *reflexiva*. A cognição experiencial é aquela na qual desempenhamos tarefas rotineiras com eficácia e sem muito esforço aparente. Em geral, esse tipo de cognição requer envolvimento e um certo nível de habilidade pré-adquirido com a tarefa realizada. Alguns exemplos cotidianos de aplicação da cognição experiencial são: falar, andar de bicicleta, preparar a refeição diária etc. A cognição reflexiva por sua vez, demanda de forma mais explícita a abstração, comparação e a tomada de decisões. Trata-se da cognição orientada à criatividade e discernimento. Exemplos de aplicação da cognição reflexiva são: projetar, aprender, escrever um livro ou programar. Ainda de acordo com Norman (1994), ambos os tipos de cognição são necessários no cotidiano e cada um deles requer suportes tecnológicos próprios.

De acordo com Preece, Sharp e Rogers (2015) a cognição também pode ser entendida em termos dos vários processos específicos envolvidos. Alguns exemplos

destes processos específicos são: a atenção, a memória, o aprendizado, a comunicação, o raciocínio e a tomada de decisões.

A atenção é o processo que nos permite focar na informação de interesse dentre um conjunto de informações disponíveis. A facilidade no processo da atenção está relacionada a dois fatores principais: (i) definição de objetivos claros e (ii) saliência da informação no ambiente. Quando o objetivo sobre a informação pretendida é claro, a atenção consiste em relacionar esse objetivo com o ambiente disponível por exemplo, ao procurar saber as horas, o interessado provavelmente irá se ater ao relógio no seu próprio pulso ou celular. Quando os objetivos não são claros, a atenção passa a ser guiada pelas informações salientes durante a varredura do ambiente. Além disso, a apresentação da informação tem um papel crucial em tornar as informações de interesse mais ou menos explícitas à atenção do interessado por exemplo, um cliente de um restaurante que tenha somente uma vaga ideia do que deseja comer, pode prestar mais atenção aos pratos que estão ilustrados no cardápio do que naqueles que estão apenas textualmente descritos.

A memória é outro processo que compõe a cognição humana. É por meio deste processo que conseguimos armazenar o conhecimento para utilizá-lo posteriormente. Todavia, este processo não garante a armazenagem de toda informação recebida pelo indivíduo e está sujeita à filtragem. A filtragem da informação ocorre inicialmente por meio da codificação isto é, o acesso e interpretação da informação no ambiente. As circunstâncias em que a codificação ocorrem podem afetar a forma como o conhecimento é representado e recuperado da memória. Desta forma, por exemplo, quanto mais o indivíduo presta atenção, reflete e compara a informação adquirida com outros conhecimentos prévios, maiores as chances de lembrança. A memorização também pode ser afetada pelo contexto em que é codificada. A influência do contexto na codificação pode ser ilustrada pela dificuldade que por vezes temos em identificar um conhecido fora do ambiente em que estamos acostumado a encontrá-lo — por exemplo, um colega de serviço no supermercado. Por fim, uma outra característica que merece destaque no processo de memorização é a capacidade dos seres humanos de reconhecerem uma informação com mais facilidade do que se lembrarem delas. Um exemplo do aproveitamento dessa característica na área da computação são as interfaces gráficas dos sistemas operacionais (*Graphical User Interfaces ou GUI*). Com as *GUI*, os usuários, ao invés de terem que se lembrar de incontáveis comandos textuais, precisam apenas fazer uma varredura visual da tela do computador para reconhecerem elementos gráficos que representem a operação desejada.

No domínio da computação, o processo cognitivo do aprendizado pode fazer uso de uma aplicação de computador como meio, para se entender um determinado assunto, ou como fim, no qual a própria aplicação é o foco do aprendizado.

Para o primeiro caso, recursos como a web, multimídia e a realidade virtual e aumentada contribuem para a ampliação das possibilidades de representação e interação

com o tema a ser compreendido. Isso possibilita que os indivíduos explorem a informação de novas formas, enriquecendo a sua experiência durante a aquisição do conhecimento.

No caso da própria aplicação de computador ser o foco do aprendizado, Carroll (1990) demonstra que o aprendizado por exploração é considerado mais fácil do que aquele em que o usuário precisa recorrer à documentação (manual). Carroll (1990) também indica que a restrição inicial de operações possíveis em uma interface, facilita o processo de aprendizagem. Novamente, as interfaces gráficas *(GUI)* dos sistemas operacionais servem como bons exemplos: elas fornecem a possibilidade de que o usuário explore diferentes opções de interação e, em geral, dão pistas de como desfazer operações incorretas. Além disso, as operações triviais e essenciais para dominar o uso da aplicação, em geral, encontram-se explícitas nestas interfaces. Por outro lado, as operações mais avançadas encontram-se aninhadas ou "escondidas" em menus e submenus. O intuito dessa estratégia de organização da interface é encorajar o usuário iniciante a dominar a aplicação em seu uso mais corriqueiro, antes que ele possa explorar operações mais complexas — as quais podem confundi-lo ou desencorajá-lo nas etapas iniciais do aprendizado.

O processo cognitivo da comunicação do ser humano ocorre essencialmente por meio da fala, audição e escrita/leitura. A facilidade no uso desses mecanismos de comunicação varia conforme a pessoa, tarefa e o contexto. Por exemplo, alguns indivíduos consideram que ouvir é mais fácil que ler; a leitura por sua vez, permite que a informação seja reprocessada inúmeras vezes sem que seja reescrita — ao passo em que a fala exige uma repetição do interlocutor. Além disso, na audição a mensagem só pode ser capturada sequencialmente, o que torna esse mecanismo menos dinâmico do que a leitura por exemplo. Na computação esses mecanismos de comunicação são empregados para ampliar a interação com os sistemas de software, por exemplo: assistentes virtuais de voz como os softwares *Siri*, Cortana e Alexa fornecem novas possibilidades do usuário interagir com dispositivos móveis somente usando a fala e audição; no domínio da acessibilidade, interfaces especiais foram projetadas para facilitar a interação de pessoas com dificuldades cognitivas de comunicação (por exemplo, (KAMARUZAMAN et al., 2016) (BRITTO; PIZZOLATO, 2014) (MYNATT, 1997)).

Por fim, os processos de raciocínio e tomada de decisão são processos tipicamente relacionados à cognição reflexiva. Esses processos estão envolvidos no planejamento, escolha de alternativas e análise das consequências das tarefas desempenhadas conscientemente. Algumas estratégias frequentemente empregadas por esses processos são: comparação entre os diversos cenários e fontes disponíveis para a solução de um problema; resolução de problemas inéditos reutilizando-se o conhecimento prévio sobre problemas semelhantes; exploração de soluções por tentativa e erro. A habilidade no uso dessas estratégias está sujeito, ainda, ao nível de experiência do indivíduo como por exem-

plo, pessoas inexperientes podem recorrer com mais frequência à estratégia de tentativa e erro pela falta de conhecimento acumulado sobre um determinado assunto; por outro lado, o acumulo de conhecimento em pessoas experientes pode favorecer uma visão mais crítica e holística durante a solução de problemas isto é, considerar as consequências de uma determinada solução além do problema analisado.

Uma das estratégias para se projetar a interação em sistemas de software é emular as interações do usuário com o mundo físico no mundo digital. Essa estratégia considera que se a interação é bem sucedida no mundo real, não deveria haver motivos para que ela não seja também no mundo digital. Ainda que essa suposição seja válida para alguns casos — por exemplo, abrir um "arquivo", arrastar "documentos" indesejados para a "lixeira", ou "folhear as páginas" de um "livro" digital em um *tablet* — ela pode ser falha em outros contextos. Um caso clássico que exemplifica uma emulação problemática da interação no mundo virtual foi a interface *"Magic Cap"* para sistemas operacionais de *PDAs* (*Personal Digital Assistants*). Esta interface empregava metáforas literais de cômodos, prédios e vilarejos para interagir com o sistema operacional. Dependendo do local em que o usuário se encontrava, ele precisava navegar entre vários cômodos até que conseguisse chegar na "agência postal" para verificar seus e-mails. Neste caso, o compromisso de manter-se fiel à metáfora se sobrepunha à facilitação da interação, complicando desnecessariamente uma tarefa que deveria ser trivial para o usuário de qualquer ponto do sistema (GENTNER; NIELSEN, 1996). Em geral, a emulação falha quando subestima-se a complexidade da tarefa do mundo físico. Nestes casos, ao invés de enriquecer a experiência, a emulação acaba restringindo as possibilidades de interação e, consequentemente, o suporte à interação.

3.3 Modelos Mentais

Uma das abordagens utilizadas pela psicologia cognitiva para compreender como o ser humano interage com o mundo à sua volta são os modelos mentais. O conceito de modelo mental é utilizado para se referir às construções internas (representações mentais) que um indivíduo desenvolve com o objetivo de manipular, inferir e explicar suas experiências sobre algo (CRAIK, 1967) (SCHAEKEN; JOHNSON-LAIRD; D'YDEWALLE, 1996) (JOHNSON-LAIRD, 2001). Para tanto, o modelo mental desenvolvido pelo usuário engloba tanto construções internas sobre a forma como algo deve ser operado quanto sobre a forma como ele funciona. A profundidade com que a pessoa conhece o dispositivo com que interage influencia sobre o modelo mental desenvolvido. Por exemplo, um engenheiro de carro de corridas tem um modelo mental extenso a respeito de como a mecânica de um carro funciona, habilitando-o a fazer reparos/melhorias. O piloto por sua vez, tem um modelo mental bem desenvolvido sobre como operar o carro em diversas

condições de pista, porém um modelo mental mais restrito que o do engenheiro sobre como a mecânica do carro funciona. Uma consequência dessas diferenças é que, não raramente, as pessoas podem acabar aplicando modelos mentais errôneos nos problemas cotidianos (NORMAN, 2014). Por exemplo, ao utilizar o termostato de um aquecedor elétrico, é comum as pessoas ajustarem o aparelho a uma temperatura mais elevada que a desejada, na tentativa de fazer o ambiente aquecer mais rápido. Todavia, aquecedores elétricos trabalham com a velocidade de aquecimento constante, desativando quando a temperatura desejada é alcançada e reativando quando ela fica abaixo da temperatura definida. Ao operar dessa forma, o usuário muito provavelmente está empregando um modelo mental distinto da forma como o dispositivo realmente funciona — mas que provavelmente funcionaria em situações que ele julga análogas por exemplo, colocar mais lenha em uma lareira ou aumentar a intensidade da chama de um fogão a gás. Desta forma, um modelo mental errôneo pode levar à operação incorreta e à frustração na operação de um sistema.

Ao desenvolver um sistema, pretende-se que os usuários sejam capazes de desenvolverem modelos mentais consistentes com as expectativas definidas pelo projetista da interação. Uma das formas de se alcançar isso é por meio do ensino/treinamento do usuário. Todavia, essa abordagem normalmente requer tempo e disposição de ler a documentação do sistema — condições nem sempre disponíveis ou receptíveis pelo usuário. Sendo assim, a alternativa consiste em projetar interações que sejam transparentes e intuitivas durante o projeto da interação. Para tanto, o projetista da interação deve conhecer bem os usuários do sistema para que possa então, entender os desafios e oportunidades de interação na perspectiva daqueles que de fato irão interagir com o sistema.

3.4 Cognição Externa

A cognição externa refere-se a uma outra abordagem da psicologia cognitiva (SCAIFE; ROGERS, 1996) para entender como a manipulação de instrumentos do mundo físico auxiliam e ampliam os processo cognitivos dos seres humanos. Dependendo da atividade cognitiva desempenhada, diferentes formas de cognição externa se aplicam. A seguir são apresentadas algumas formas de cognição externa caracterizadas pela literatura.

3.4.1 Liberação da Carga de Memória e Computacional por Exteriorização

A liberação da carga de memória por exteriorização é uma forma de cognição externa aplicada frequentemente em nosso dia-a-dia. Nessa estratégia, o conhecimento é exteriorizado em representações do mundo físico. O objetivo é auxiliar na memorização

de tarefas consideradas difíceis de serem lembradas autonomamente. Calendários e diários são exemplos de artefatos empregados nessa forma de cognição externa. Eles possibilitam a externalização de eventos importantes que, a princípio, deveriam ser mantidos na memória do interessado. Desta maneira, esses artefatos funcionam como extensões da memória do usuário. Esse recurso auxilia o usuário a reduzir a sua "carga de memória" com eventos importantes, os quais, de outra forma, disputariam a necessidade de lembrança com os demais eventos de seu cotidiano.

Na liberação de carga computacional, o objetivo é facilitar a realização de um cálculo ou operação por meio de representação externa. Por exemplo, durante o cálculo de multiplicação com operandos de vários dígitos, as pessoas frequentemente precisam lembrar os resultados intermediários. Ao exteriorizar os resultados intermediários usando papel e caneta, o interessado pode substituir o armazenamento de informações na sua memória de curto prazo por uma operação de registro e busca no mundo físico. Isso contribui para que ele foque na realização dos demais passos envolvidos no cálculo — resgatando os valores intermediários quando necessário e diminuindo os riscos de se confundir com os passos já realizados.

3.4.2 Anotação e Rastreamento Cognitivo

A cognição também pode ser externalizada por meio da modificação (anotação) e manipulação (rastreamento cognitivo) de representações externas. O uso de lista de supermercado é um exemplo de externalização por anotação. Por meio desse recurso, a pessoa não somente libera a carga de memória com o registro de itens que precisam ser comprados. Ao riscar os itens da lista, ela também otimiza o controle sobre a modificação da tarefa, percebendo com rapidez os itens que foram selecionados (ou estão indisponíveis) e os que restam ser colocados no cesto de compras.

O rastreamento cognitivo é adequadamente ilustrado pelo jogo de cartas. Jogadores de cartas frequentemente rearranjam as cartas na mão de acordo com o naipe e a numeração das cartas. Essa externalização facilita o rastreamento de cartas inseridas e removidas ao longo de uma partida. A redução da necessidade de busca aleatória de cartas permite que o jogador dedique mais atenção à estratégia do jogo e aos demais jogadores.

3.5 Cognição Distribuída

O paradigma tradicional de estudo da cognição humana considerou por bastante tempo somente as representações internas na modelagem dos processos cognitivos humanos — isto é, somente as representações mentais próprias de um indivíduo eram consideradas na representação de problemas e soluções envolvendo a cognição. Essa abordagem

negligenciava a interação do indivíduo com o ambiente, os artefatos do mundo físico e a interação com os demais membros de uma atividade. Todavia, esta abordagem mostrou-se insuficiente ao longo do tempo, principalmente em contextos colaborativos. Mesmo considerando somente as representações internas, elementos do mundo físico acabavam sendo invocados — ainda que de forma velada — para preencher as lacunas em problemas envolvendo a modelagem das atividades cognitivas humanas (WALENSTEIN, 2002).

Considerando as deficiências do paradigma tradicional, Hutchins (1996) propôs o paradigma da Cognição Distribuída para modelar e entender tarefas envolvendo a cognição. Nesse paradigma, considera-se que propagação da informação — denominada de mudança no estado representacional — para construção e uso do conhecimento, não é feita de forma individual, mas composta pelos vários processos/representações das pessoas e artefatos envolvidos em um sistema cognitivo.

Um procedimento médico em uma sala de cirurgia exemplifica como a mudança no estado representacional ocorre em um sistema cognitivo: o oxímetro preso ao dedo de um paciente faz a leitura da oxigenação e do batimento cardíaco, os quais são exibidos no monitor cardíaco; a informação dos batimentos mostrada no monitor é lida pelo anestesista que aumenta ou diminui a sedação do paciente e informa ao cirurgião quando iniciar o procedimento. Nota-se que a informação do batimento cardíaco foi transmitida e transformada ao longo de todo o sistema cognitivo: primeiro nos sinais gerados pelo oxímetro, em seguida na imagem renderizada no monitor, após isso na representação mental sobre a informação lida pelo anestesista, depois na ação motora do anestesista na regulagem da sedação e, por fim, na comunicação do anestesista com o cirurgião.

De acordo com o paradigma da cognição distribuída a análise de um sistema cognitivo envolve, em geral (PREECE; SHARP; ROGERS, 2015):

- o entendimento sobre como as pessoas resolvem os problemas;
- a análise da comunicação verbal/não-verbal;
- a atenção aos canais de comunicação utilizados;
- o conhecimento sobre como a informação é compartilhada e acessada;
- a identificação de problemas e de soluções que emergem e são aplicadas nesse processo;
- a compreensão a respeito das formas de coordenação aplicadas entre as pessoas.

Hollan, Hutchins e Kirsh (2000) propuseram um *framework* para mapear as atividades de pesquisa envolvidas em estudos que seguem a abordagem da cognição distribuída (vide Figura 3.1). Este *framework* já foi adaptado e utilizado para mapear as atividades de pesquisa de suporte cognitivo na área da engenharia do conhecimento (ERNST, 2004). De acordo com o *framework*, o princípio da cognição distribuída deve ser aplicado para identificar os elementos centrais que envolvem a atividade em estudo isto é, pessoas,

comunicação, colaboração, artefatos, contexto e ações de descarregamento cognitivo praticadas no contexto considerado. Esses elementos servem para informar as classes de fenômenos que merecem observação. A partir disto, estudos que empregam técnicas de etnografia cognitiva (por exemplo, entrevista contextual, estudo observacional, *surveys* qualitativos etc.) são aplicados para observar, documentar e analisar o(s) fenômeno(s) de interesse. Em função do caráter observacional desses estudos, podem haver casos em que os dados coletados não sejam suficientes para permitir que inferências sejam feitas adequadamente. Nestes casos, os resultados do estudo etnográfico servem para informar novos estudos experimentais que possam enriquecer os dados. Por meio da sinergia entre essas três áreas de atividade do *framework*, informações prescritivas sobre o *design* dos materiais de trabalho (por exemplo, novos elementos ou recursos de suporte cognitivo) ganham forma. Essa iteração contudo, pode não ser tão trivial quanto parece. Isso porque os materiais de trabalho são eles próprios parte do local de trabalho (contexto). Logo, a inserção de um novo material de trabalho representa uma alteração na dinâmica do contexto em que ocorre a cognição distribuída. Desta forma, a própria inserção promove uma nova oportunidade de análise etnográfica, permitindo o teste e revisão da teoria que emerge durante este ciclo. Todavia, os autores ressaltam que deve ser guardada uma certa precedência à aplicação do princípio da cognição distribuída, já que é ela que informará as experiências, a observação etnográfica, o *design* dos materiais e do ambiente de trabalho.

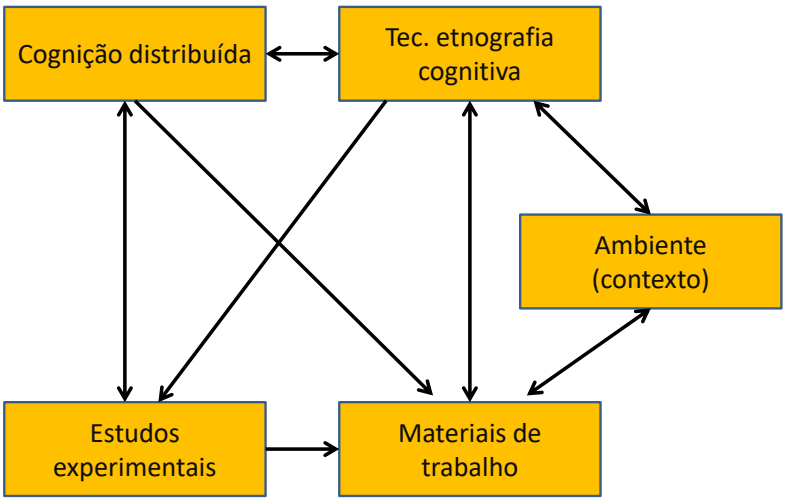

Figura 3.1: Mapa de atividades de pesquisa integradas (adaptado de Hollan, Hutchins e Kirsh (2000) e Ernst (2004))

Neste trabalho, também estabelecemos uma correspondência entre as atividades de pesquisa e o framework proposto por Hollan, Hutchins e Kirsh (2000). Nesse sentido,

utilizamos o princípio da cognição distribuída na identificação das demandas cognitivas de teste unitário (resumidas em nosso *framework* inicial definido no Capítulo 5). Com base nessas demandas, analisamos lacunas cognitivas nos materiais de trabalho isto é, nas ferramentas visuais existentes que apoiam o teste unitário (Capítulo 6). A partir do conhecimento obtido nessas duas atividades, projetamos um estudo qualitativo (Capítulo 7 que emprega técnicas de etnografia cognitiva para estudar a prática de revisão de testes unitários em uma abordagem centrada nos profissionais da atividade. A partir deste estudo — que também observa os princípios da cognição distribuída em sua análise — propusemos um *framework* de tarefas que demandam suporte cognitivo o qual informa, simultaneamente, novos estudo experimentais e propostas de alterações nas ferramentas de teste unitário e no contexto em que elas executam (materiais e ambiente de trabalho).

3.6 Considerações Finais

Neste capítulo, foram abordados os principais fundamentos sobre cognição relacionados ao contexto deste trabalho. Inicialmente, foram explicadas as diferenças entre os conceitos de cognição experiencial e reflexiva e exemplificadas tarefas que ilustram o uso desses dois tipos de cognição. Em seguida, foram abordados os processo específicos envolvidos na cognição (atenção, memória, aprendizado, comunicação, raciocínio e tomada de decisão).

Três abordagens utilizadas pela psicologia cognitiva para descrever a cognição do ser humano em sua interação com o mundo foram apresentadas — modelos mentais, cognição externa e cognição distribuída. A primeira e mais tradicional (modelos mentais), considera a representação da cognição somente por meio de construções internas, ou seja, só aquilo que acontece na mente do indivíduo é importante para modelar a cognição. A segunda (cognição externa) admite a capacidade de instrumentos do mundo físico ampliarem as capacidades cognitivas dos seres humanos. A terceira (cognição distribuída) incorpora elementos do mundo externo à mente do indivíduo (por exemplo, a colaboração entre os indivíduos e os artefatos do mundo físico) para explicar como ocorre a mudança do estado representacional em um sistema cognitivo.

Por fim, estabelecemos uma correspondência entre as atividades de pesquisa realizadas neste trabalho e a abordagem da cognição distribuída segundo o *framework* proposto por Hollan, Hutchins e Kirsh (2000).

CAPÍTULO 4

Mapeamento Sistemático sobre Ferramentas e Técnicas de Visualização para o Teste de Software

4.1 Considerações Iniciais

O estudo da visualização na área da computação é responsável pelo desenvolvimento e aplicação de modelos visuais que permitam explorar a informação a partir de representações gráficas dos dados e suas inter-relações. Isso é feito explorando-se as capacidades cognitivas da mente humana a partir de informações capturadas pelo sentido da visão (LUZZARDI et al., 2004). Além disso, os dados a serem modelados visualmente não requerem a existência prévia de uma representação geométrica associada (SPENCE, 2007). O propósito é ampliar o conhecimento obtido sobre os dados os quais, em sua representação original, seriam insuficientes ou tomariam um longo tempo para serem interpretados (CARD; MACKINLAY; SHNEIDERMAN, 1999a).

Neste capítulo, são apresentados detalhes de um mapeamento sistemático conduzido com o intuito de determinar como técnicas e ferramentas de visualização aplicadas ao teste de software têm sido abordadas na literatura. O processo aplicado no mapeamento foi alinhado às definições de Petersen et al. (2008) e Brereton et al. (2007).

Nas seções seguintes são apresentados os parâmetros do protocolo adotado na definição e planejamento da pesquisa, detalhes sobre a condução do mapeamento, resultados alcançados após a fase de extração, análise dos resultados e questões de validade do estudo de mapeamento.

4.2 Planejamento

O principal objetivo deste estudo foi a identificação e caracterização de estudos sobre técnicas e ferramentas de visualização para apoio ao teste de software. O planejamento de um mapeamento sistemático é composto pela definição do protocolo, estratégia

de pesquisa e critérios de inclusão/exclusão. O protocolo exibido na tabela 4.1 consiste na definição das questões de pesquisa, população, intervenção, controle, resultados e aplicação.

Tabela 4.1: Protocolo do mapeamento sistemático

Assunto	**Descrição**
Questões de Pesquisa	RQ.1. Como o número de publicações sobre ferramentas e técnicas de visualização, aplicadas ao teste de software, têm evoluído nas principais bibliotecas digitais das áreas de pesquisa envolvidas? RQ.2 - Qual é o perfil das ferramentas e técnicas de visualização, aplicadas ao teste de software, com relação a atributos de teste, visualização e treinamento? RQ.3 - Como as ferramentas e técnicas de visualização, que suportam a atividade de teste de software, têm sido avaliadas?
População	A população é composta de trabalhos científicos que propõem/avaliam estudos de visualização de informação aplicados à atividade de teste de software e que foram publicados em bibliotecas digitais.
Intervenção	A característica observada foi a aplicação de ferramentas e técnicas de visualização na atividade de teste de software. A observação de características é feita do ponto de vista de pesquisadores de engenharia de software.
Controle	Um total de seis artigos ((JONES; HARROLD; STASKO, 2002), (LI; HORGAN, 2003), (LAWRENCE et al., 2005a), (MAO; LU, 2007), (DALTON et al., 2009), (ENGSTRöM; RUNESON, 2013)) foi previamente definido pelos autores para ser usado como controle para a *string* de busca. Os resultados da busca serão considerados adequados se retornarem pelo menos metade desses artigos em cada uma das bibliotecas consideradas.
Resultados	O resultado esperado no estudo de mapeamento é a caracterização de como as ferramentas e as técnicas de visualizações encontram-se distribuídas sobre a atividade de teste. Isto deve considerar: as fases de teste apoiadas pela visualização; nível de teste considerado (unitário, integração, sistema); a distribuição dos trabalhos entre as técnicas de teste (funcional, estrutural, baseada em defeitos); como as propostas de visualização orientadas ao teste de software estão sendo avaliadas.
Aplicação	i) Colaborar com um melhor entendimento de como a atividade de teste está sendo atualmente apoiada pelas ferramentas e técnicas de visualização; ii) Prover dados para apoiar o desenvolvimento de novos trabalhos orientados à pesquisa das oportunidades identificadas; iii) Contribuir com engenheiros de software e pesquisas de visualização sobre quais são as soluções já estabelecidas relacionadas à atividade de teste de software.
Ferramentas e Instrumentação	As bibliotecas digitais adotadas no mapeamento sistemático foram: "*IEEE Xplore" (IEEE), "ACM" e "Science Direct"*. Para auxiliar a condução do processo, a ferramenta *StArt* (HERNANDES et al., 2012) foi utilizada. A ferramenta *StArt* segue os passos do processo que são geralmente adotados na maioria dos estudos secundários, os quais compreendem as fases de "Identificação", "Seleção" e "Extração".
Strings de Busca	ACM: *(Abstract:visuali*tion and Abstract:(technique or tool) and Abstract:(software or program) and Abstract:test*)* IEEE: *("Abstract":visuali*tion AND (tool OR technique) AND ((software OR program) AND (test*)))* Science Direct: *tak(visuali*tion AND (tool OR technique) AND ((software OR program) AND (test*)))*

4.2.1 Critérios de Inclusão e Exclusão

As seguintes restrições foram adotadas na etapa de identificação: (i) estudos não escritos em língua inglesa; (ii) estudos anteriores ao ano 2000. Essas restrições foram adotadas para tornar o mapeamento replicável por outros pesquisadores e para delimitar uma janela de tempo que fosse simultaneamente aceitável e compatível com os recursos de pesquisa disponíveis— tempo para realização e número de pesquisadores envolvidos.

Os seguintes critérios de inclusão (CI) e exclusão (CE) foram adotados nas fases de seleção e extração:

CI.1 Ferramenta de visualização aplicada ao teste de software.
CI.2 Técnica de visualização aplicada ao teste de software.
CI.3 Estudo experimental sobre ferramentas ou técnicas de visualização no contexto de teste de software.
CE.1 Trabalho não relacionado a ferramentas e técnicas de teste de software.
CE.2 Ferramenta ou técnica com enfoque secundário no teste de software.
CE.3 Não é um artigo de conferência, artigo de *journal*, capítulo, dissertação, tese, ou não se encontra disponível.

Para poder responder às questões de pesquisa do mapeamento, dados relevantes precisam ser coletados durante a leitura dos artigos. Estes dados são obtidos pela observação de critérios de classificação pré-definidos (CC). Os critérios de classificação representam propriedades objetivas e coerentes com os parâmetros “intervenção” e “resultados”, definidos no protocolo. Para alguns critérios estabelecidos, o valor “outros” foi criado para classificar trabalhos que não atendiam quaisquer dos valores pré-definidos mas atendiam a valores possivelmente existentes. O valor “não adequado” foi definido para classificar trabalhos que não atendem um determinado critério quando um valor possível não existe ou não pode ser inferido. O valor “múltiplo” foi usado para indicar casos em que pelo menos dois valores do referido critério de classificação são adequados para o julgamento do trabalho. Os critérios de classificação adotados são apresentado na tabela 4.2.

4.2.2 Condução do Mapeamento

Na fase de “Identificação” do mapeamento, as restrições definidas na Seção 4.2.1 foram aplicadas. Do total de 649 trabalhos retornados das três bibliotecas digitais (232 da *ACM*, 348 da *IEEE* e 69 da *Science Direct*), 76 artigos foram detectados como duplicados e um total de 573 trabalhos foram de fato considerados.

Na fase de “Seleção”, conduzimos um processo de revisão baseado nos títulos dos artigos e *abstracts* e eventualmente — quando isso não era suficiente — baseado

Tabela 4.2: Critérios de classificação

Critério	Rótulo	Propósito	Valores
CC.1	Nome	Registra o nome da ferramenta ou técnica proposta quando aplicável.	-
CC.2	Fonte	Registra onde o trabalho foi publicado.	(i) ACM; (ii) IEEE; (iii) Science Direct.
CC.3	Foco	Registra o enfoque do trabalho.	(i) Ferramentas; (ii) Técnicas; (iii) Múltiplo; (iv) Estudo experimental.
CC.4	Contexto de avaliação	Registra o contexto em que o trabalho foi avaliado	(i) Indústria; (ii) Academia; (iii) Múltiplo; (iv) Pobremente avaliado/não detalhado; (v) Não adequado.
CC.5	Nível de teste	Registra o nível de teste abordado pela proposta.	(i) Unitário; (ii) Integração; (iii) Sistema; (iv) Múltiplo; (v) Não adequado.
CC.6	Objeto de Análise	Registra o tipo de artefato de software representado ou mapeado pela visualização.	(i) Código; (ii) Modelo/Diagrama; (iii) Trace; (iv) GUI; (v) Chamada de procedimento/método; (vi) Threads; (vii) Múltiplo; (viii) Outros, (ix) Não adequado.
CC.7	Técnica de teste	Registra a técnica de teste caso a proposta apoie algum critério dessa técnica.	(i) Funcional; (ii) Estrutural; (iii) Baseado em defeitos; (iv) Múltiplo; (v) Nenhum critério; (vi) Não adequado.
CC.8	Substrato de visualização	Registra o tipo de substrato usado para representar a visualização.	(i) 2D; (ii) 3D; (iii) Múltiplo; (iv) Não adequado.
CC.9	Escopo no ciclo de vida	Registra a fase no ciclo de vida do desenvolvimento de software à qual a proposta melhor se aplica.	(i) Análise, (ii) Design, (iii) Construção, (iv) Implantação, (v) Múltiplo, (vi) Não adequado.
CC.10	Apoio ao treinamento	Registra se a ferramenta ou técnica proposta provê algum apoio ao treinamento ou aprendizado da visualização proposta.	(i) Sim; (ii) Não; (iii) Não adequado.
CC.11	Linguagem alvo/Plataforma	Registra o nome da linguagem ou plataforma de programação abordada pela proposta do trabalho.	-
CC.12	Notas Relevantes	Registra informações importantes sobre a proposta, identificadas durante a leitura dos trabalhos.	-
CC.13	Interação	Registra o tipo de interação usada para manipular a visualização.	(i) Cursor; (ii) Prompt; (iii) Touchscreen; (iv) Sensor(res); (v) Outros; (vi) Múltiplo; (vii) Não adequado.
CC.14	Fase de teste	Registra a fase de teste melhor atendida pela técnica/ferramenta de visualização.	(i) Planejamento; (ii) Implementação; (iii) Análise; (iv) Múltiplo; (v) Não adequado.
CC.15	Modelo de referência de visualização	Registra se uma ferramenta/técnica de visualização foi construída baseando-se em um modelo de referência de visualização.	(i) Sim; (ii) Não; (iii) Não adequado.

na leitura da introdução e/ou conclusão. Nessa etapa do estudo os trabalhos eram inicialmente lidos por um revisor, de forma a determinar quais artigos deveriam ser mantidos ou descartados. Depois, outro revisor fazia o seu julgamento considerando o mesmo conjunto de trabalhos. Artigos aceitos/rejeitados por ambos revisores eram diretamente aceitos/rejeitados para o próximo passo. Em caso de opiniões divergentes, o artigo era definido como aceito para ser lido na próxima fase. Adicionalmente, para casos em que dois ou mais trabalhos do mesmo autor, descrevendo a mesma técnica, eram identificados, apenas as publicações mais recentes eram consideradas. Após completar a análise de 573 artigos, 485 foram considerados rejeitados e 88 foram aceitos e passaram para a fase seguinte.

Na fase de "Extração", cada um dos 88 trabalhos identificados na fase anterior foram cuidadosamente lidos e classificados permanentemente, de acordo com os critérios de inclusão e exclusão. Dos 88 estudos, 3 foram considerados duplicados, 21 foram rejeitados e 64 foram aceitos. Os critérios de classificação também foram concomitantemente avaliados neste passo, após cada artigo aceito ser lido (ver Figura 4.1).

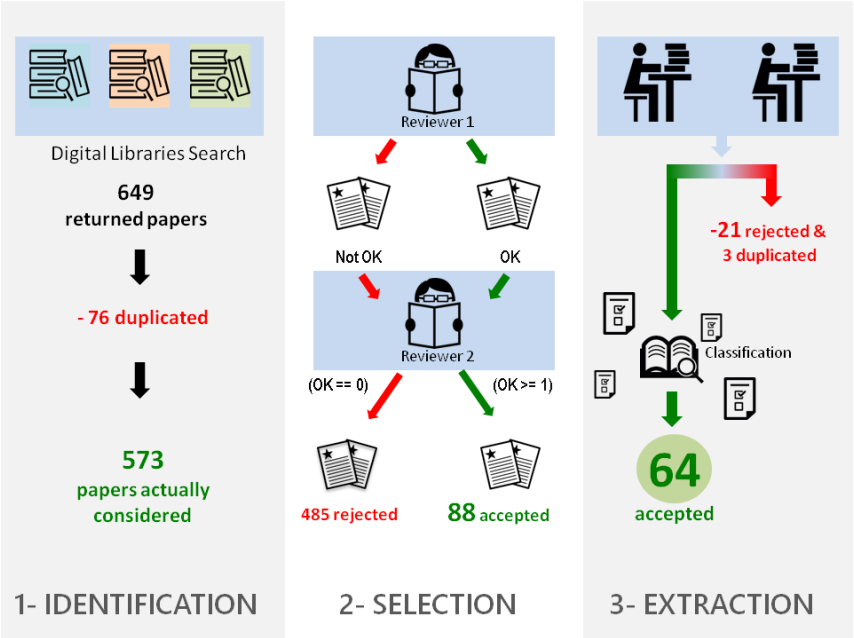

Figura 4.1: Infográfico resumindo o processo de mapeamento sistemático.

4.2.3 Resultados e Análise

A primeira questão de pesquisa *(RQ1)* pergunta: *"Como o número de publicações sobre ferramentas e técnicas de visualização, aplicadas ao teste de software, tem evoluído nas principais bibliotecas digitais das áreas de pesquisa envolvidas?"*

Na Figura 4.2 é ilustrada a oscilação das publicações antes da fase de seleção (linhas tracejadas) e após a fase de extração (linhas contínuas), por ano e por biblioteca.

A biblioteca que retornou mais publicações no estudo foi a *IEEE*, seguida sucessivamente pela *ACM* e *Science Direct*. Com relação aos resultados pós-extração, notamos que a mesma ordem de contribuição permanece, embora com diferenças menores entre as bases. A comparação entre as regiões de cristas e vales nas curvas da IEEE e ACM também permitem identificar o biênio 2007-2008 como aquele em que ocorreu a maior variação de publicações relevantes para o estudo (diferenças somadas de 13 publicações a favor da *IEEE* isto é, aproximadamente 20% do total de estudos relevantes). Em outros anos, entretanto, o número de publicações manteve-se proporcional entre as bibliotecas. A biblioteca *Science Direct* não exibiu um volume relevante de publicações relevantes ao estudo, não excedendo mais do que uma publicação relevante por ano e apresentando várias ocorrências de anos sem quaisquer publicações.

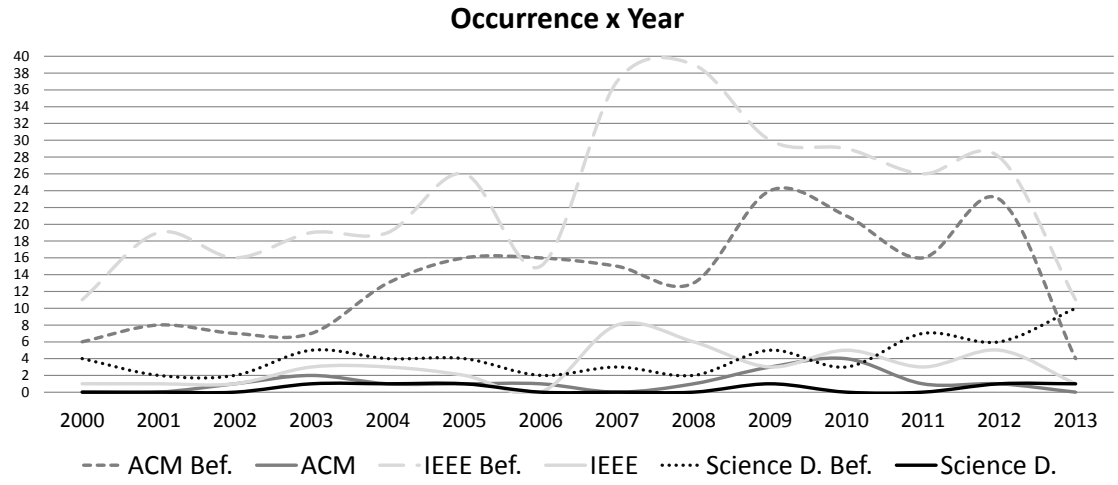

Figura 4.2: Ocorrências de publicações ao longo dos anos separadas por bibliotecas digitais antes da fase de seleção (sufixo "Bef.") e pós-extração, excluindo-se os trabalhos duplicados.

Teste de hipótese foi aplicado para checar se as diferenças entre as bibliotecas digitais na fase pós-extração, quanto ao número de publicações ao longo dos anos, eram significantes. O teste de *Shapiro-Wilk* foi aplicado para determinar se seria apropriado aplicar testes paramétricos ou não paramétricos, considerando os seguintes parâmetros para cada biblioteca digital:

- H_0 = Os valores foram amostrados de uma população que segue uma distribuição Gaussiana.
- H_1 = Os valores não foram amostrados de uma população que segue uma distribuição gaussiana.
- Nível de significância (α) = 0.05.

A significância estatística (*p-value*) para as ocorrências nas bibliotecas *ACM, IEEE e Science Direct* foram, respectivamente: $p = 0,005178$, $p = 0,181980$ e $p = 0,000092$. Uma vez que $p < \alpha$ para *ACM* e *Science Direct*, a hipótese nula é rejeitada em ambos os grupos, ao nível de 5%. Considerando que o teste de hipótese a ser aplicado deve considerar a distribuição das três bibliotecas, e duas delas não seguem uma distribuição normal, o teste não paramétrico *Kruskal-Wallis rank sum* foi aplicado. Os parâmetros foram estabelecidos da seguinte forma:

- H_0 = Em média, as três bibliotecas digitais têm o mesmo número de ocorrências relevantes (pós-extração) ao longo dos anos.
- H_1 = Em média, pelo menos uma das três bibliotecas digitais não têm o mesmo número de ocorrências relevantes (pós-extração) ao longo dos anos.
- Nível de significância (α) = 0.05.

O *p-value* para o teste é de 0.00042. Assim, $p < \alpha$ e rejeita-se a hipótese nula de que, em média, as três bibliotecas digitais têm o mesmo número de ocorrências relevantes (pós-extração) ao longo dos anos. As múltiplas comparações das bibliotecas digitais tomadas em pares, resultante do testes de Kruskal-Wallis *rank sum*, indicam que o teste foi significativo devido às diferenças observadas entre o par "IEEE - Science Direct"'. Desta forma, confirma-se que a Science Direct, significativamente, contribuiu em média com menos artigos que a IEEE, para o estudo de mapeamento no tópico investigado.

A segunda questão de pesquisa *(RQ2)* aborda: *"Qual é o perfil das ferramentas e técnicas de visualização, aplicadas ao teste de software, com relação a atributos de teste, visualização e treinamento?"*. Para responder essas questões, foram feitos os cruzamentos dos dados levantados por meio dos critérios de classificação, de acordo com três aspectos de interesse:

i *Aspectos de teste*: CC5 vs CC3 (Nível de teste vs. Foco), CC5 vs. CC7 (Nível de teste vs. técnica de teste), CC5 vs. CC6 (Nível de teste vs. Objeto de análise), CC5 vs.CC14 (Nível de teste vs. Fase de teste);

ii *Aspectos de visualização*: CC3 vs. CC8 (Foco vs. Substrato de visualização), CC8 vs. CC13 (Substrato de visualização vs. Interação), CC15 vs. CC8 (Modelo de referência de visualização vs. Substrato de visualização);

iii *Treinamento*: CC.3 vs. CC.10 (Foco vs. Apoio ao treinamento), CC10 vs. CC14 (Apoio ao treinamento vs Fase de teste), CC10 vs. CC5 (Apoio ao treinamento vs. Nível de teste), CC10 vs. CC7 (Apoio ao treinamento vs. técnica de teste);

A maior parte dos artigos (25/64) refere-se a propostas de FERRAMENTAS de visualização que apoiam a atividade de teste. Propostas que abordam tanto FERRAMENTAS quanto TÉCNICAS de visualização no apoio ao teste, correspondem a 19/64 dos trabalhos.

Propostas que abordam somente TÉCNICAS de visualização no apoio ao teste são 17/64. ESTUDOS EXPERIMENTAIS correspondem a 3/64.

Com relação aos aspectos de teste, o primeiro cruzamento (Figura 4.3) indica que a maior parte dos trabalhos apoia MÚLTIPLOS níveis de teste (32/64) e estão uniformemente distribuídos entre FERRAMENTAS de visualização (12/32), TÉCNICAS de visualização (11/32) e propostas MÚLTIPLAS — ferramenta mais técnica (9/32). O teste UNITÁRIO é majoritariamente adotado em artigos orientados a propostas MÚLTIPLAS (FERRAMENTAS e TÉCNICAS) ou FERRAMENTAS de teste e menos adotado em artigos voltados para TÉCNICAS e EXPERIMENTOS (6/12, 4/12, 1/12 e 1/12 respectivamente). Teste de SISTEMA e de INTEGRAÇÃO foram, nessa ordem, os dois níveis de teste com menos propostas de visualização (9/64 e 5/64). Artigos considerados NÃO ADEQUADOS quanto ao nível de teste correspondem a 6/64.

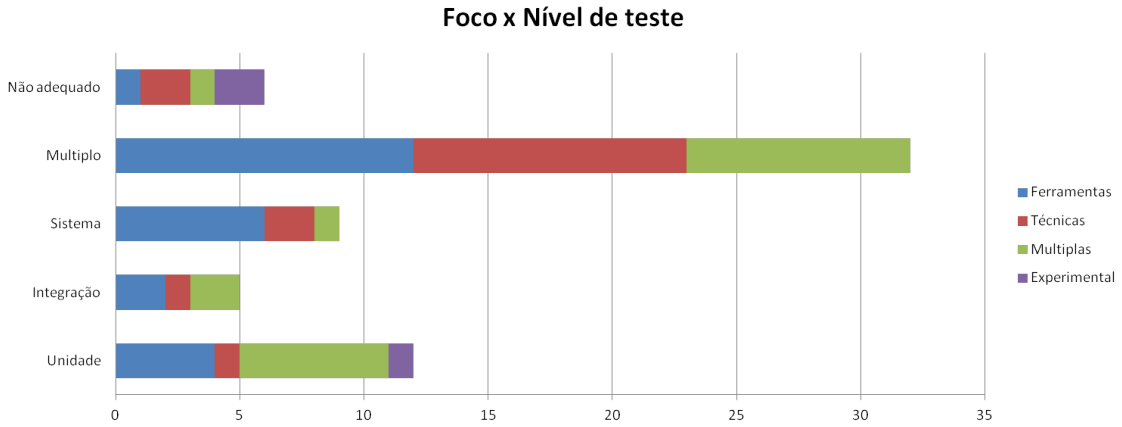

Figura 4.3: Cruzamento entre critérios Foco e Nível de teste.

Do segundo cruzamento (Figura 4.4) nota-se que a maioria das propostas de visualização (45/64) não apoia a aplicação de critérios de NENHUMA técnica de teste, seguido respectivamente por propostas que apoiam critérios da técnica ESTRUTURAL (10/64), FUNCIONAL (5/64) e BASEADA EM DEFEITOS (2/64). Nenhuma ocorrência de artigo suportando critérios de MÚLTIPLAS técnicas de teste foi encontrado. Nenhuma das propostas de visualização que apoia critérios de algum tipo de técnica de teste considera o nível de teste de integração. Quase todas as propostas que apoiam o nível de teste de SISTEMA não apoiam NENHUM critério de qualquer técnica de teste (8/9).

O terceiro cruzamento (Figura 4.5) mostra que a maioria das propostas de visualização faz o mapeamento visual do artefato CÓDIGO-FONTE (25/64). A maioria dessas propostas são dedicadas ao nível de teste UNITÁRIO (7/25) e MÚLTIPLOS níveis de teste (12/25). Os artefatos OUTROS, MÚLTIPLOS, THREADS, CHAMADA DE PROCEDI-

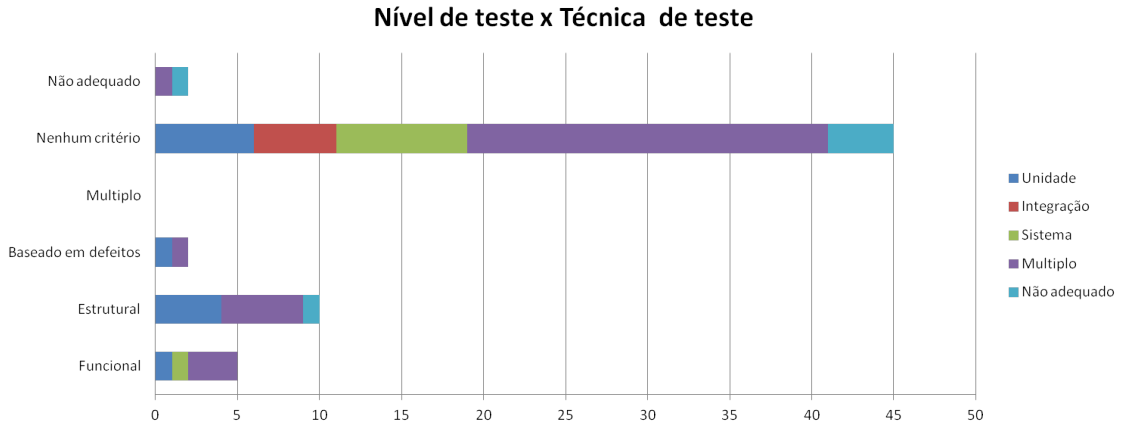

Figura 4.4: Cruzamento entre critérios Nível de tese e Técnica de teste.

MENTO, GUI, TRACE e MODELO/DIAGRAMA correspondem, respectivamente, a 11/64, 9/64, 4/64, 2/64, 1/64, 8/64 e 3/64 das ocorrências.

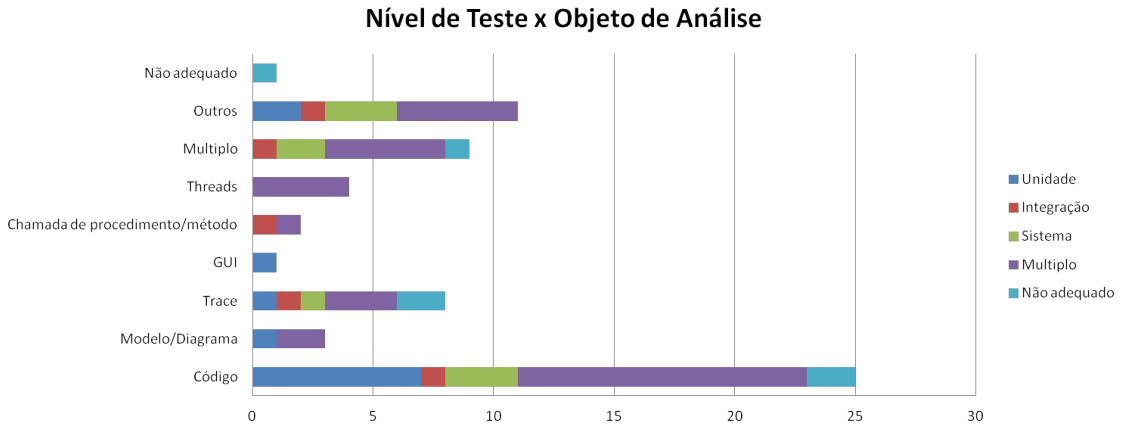

Figura 4.5: Cruzamento entre critérios Nível de teste e Objeto de análise.

No quarto cruzamento (Figura 4.6) observa-se que a maioria das propostas de visualização (29/64) não é dedicada a uma fase específica do teste de software (MÚLTIPLO), seguido por propostas que são orientadas às fases de PLANEJAMENTO (20/64), ANÁLISE (11/64) e IMPLEMENTAÇÃO (1/64) dos casos de teste. Artigos classificados como NÃO ADEQUADOS para qualquer fase de teste correspondem a 3/64 ocorrências. Com relação aos artigos relacionados a alguma fase específica do teste, aqueles relacionados ao PLANEJAMENTO dos testes se subdividem entre os níveis UNITÁRIO (3/20), INTEGRAÇÃO (1/20), SISTEMA (5/20), MÚLTIPLOS níveis (9/20) e NÃO ADEQUADOS ao critério nível de teste (2/20). O único trabalho encontrado relacionado à fase de IMPLEMENTAÇÃO do teste é relacionado ao teste de sistema. A maior parte dos trabalhos relacionados à fase de ANÁLISE dos testes está associada a MÚLTIPLOS níveis de teste (9/11).

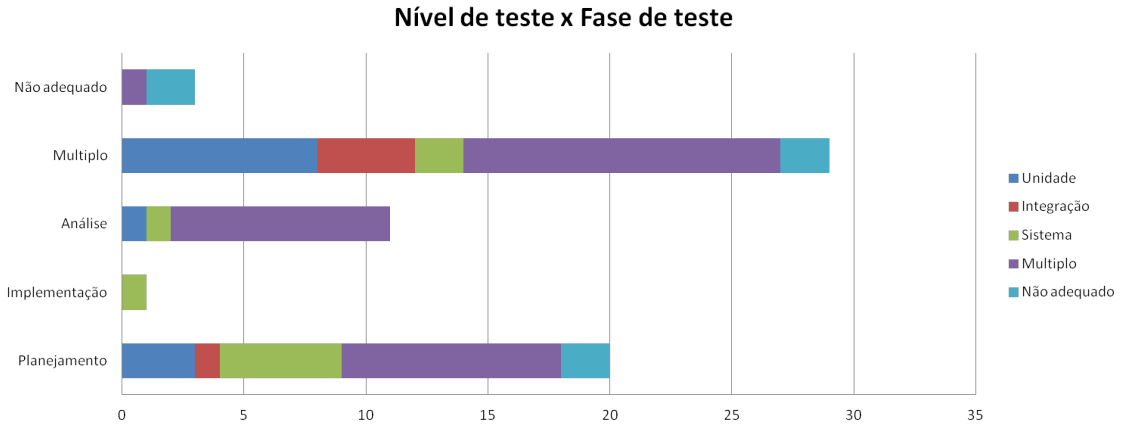

Figura 4.6: Cruzamento entre critérios Nível de teste e Fase de teste.

Com relação aos aspectos de visualização, o primeiro cruzamento (Figura 4.7) mostra que a maioria das propostas de visualização exploram o substrato BIDIMENSIONAL (58/64). Dentre estes, 23/58 propõem FERRAMENTAS, 16/58 TÉCNICAS, 16/58 MÚLTIPLAS e 3/58 correspondem a ESTUDOS EXPERIMENTAIS. Apenas 1/64 explora o substrato TRIDIMENSIONAL, 4/64 exploram ambos (MÚLTIPLOS) substratos e 1/64 trabalho foi classificado como NÃO ADEQUADO a ao critério de classificação substrato de visualização.

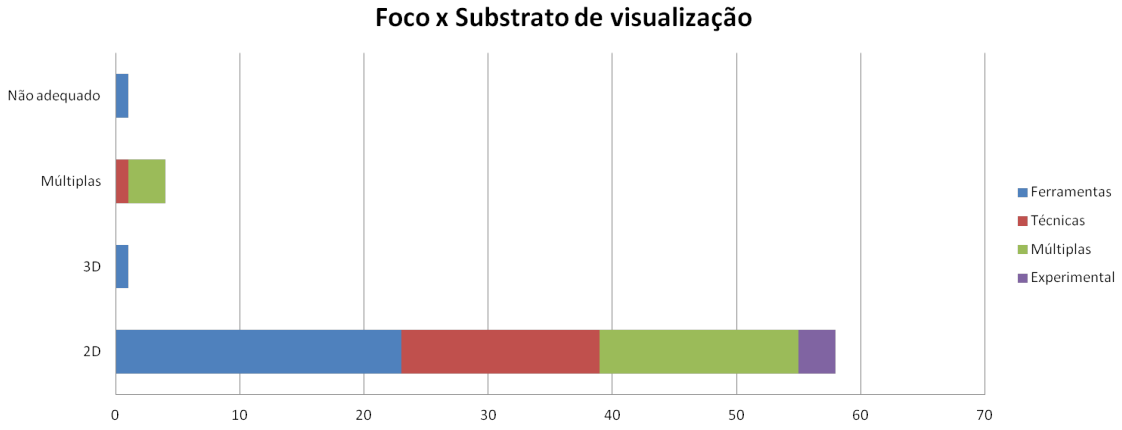

Figura 4.7: Cruzamento entre critérios Foco e Substrato de visualização.

No segundo cruzamento relacionado a aspectos de visualização (Figura 4.8), observa-se que dentre as propostas com interação via CURSOR, 56/60 propunham a interação com a visualização em um substrato BIDIMENSIONAL, 1/60 no substrato TRIDIMENSIONAL e 3/60 em ambos (MÚLTIPLOS) substratos — os demais trabalhos não apresentavam formas de interação com a visualização e foram classificados com o valor NÃO ADEQUADO.

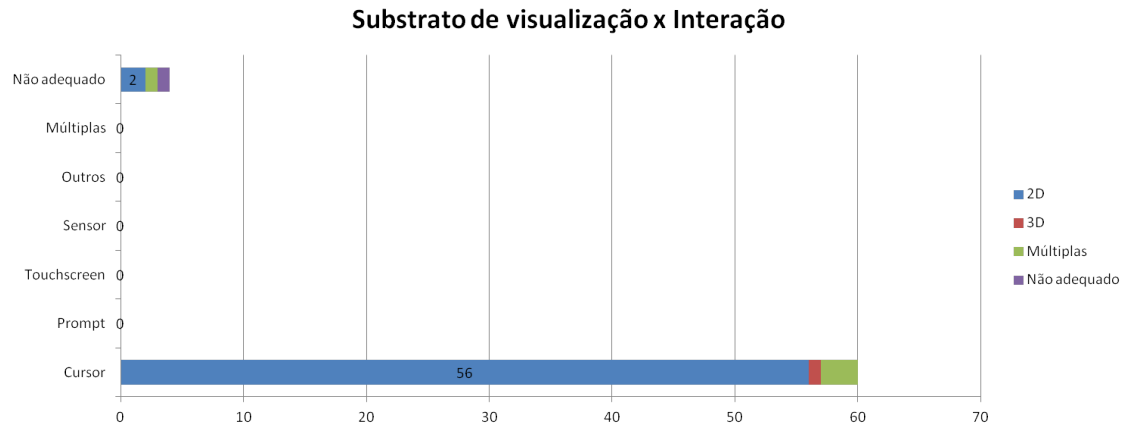

Figura 4.8: Cruzamento entre critérios Substrato de visualização e Interação.

Por fim, observou-se que nenhuma das propostas baseou a sua visualização em um modelo de referência.

Para o aspecto treinamento (iii), o primeiro cruzamento (Figura 4.9) revela que NENHUM apoio ao treinamento é provido em 59/64 das propostas de visualização. Dentre as propostas que registram algum tipo de suporte ao treinamento, as propostas de FERRAMENTAS e MÚLTIPLAS correspondem respectivamente a 4/5 e 1/5 ocorrências.

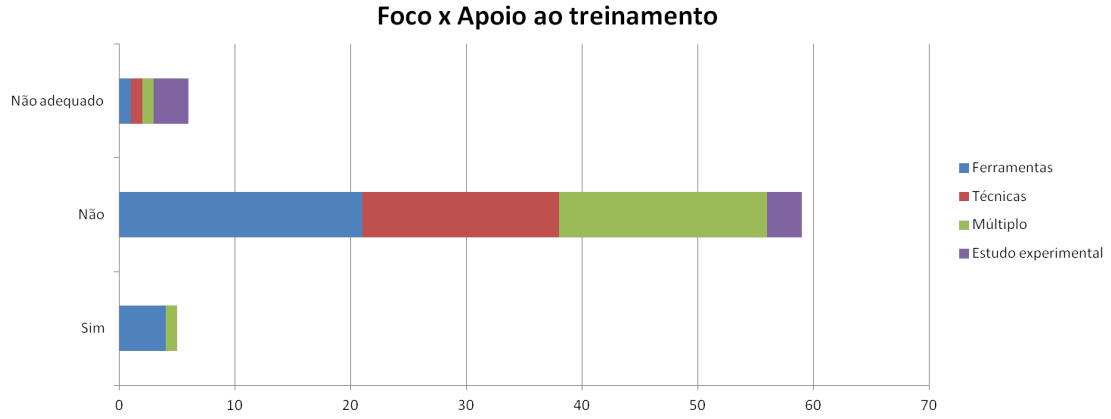

Figura 4.9: Cruzamento entre critérios Foco e Apoio ao treinamento.

O segundo cruzamento relacionado ao aspecto treinamento (Figura 4.10) permite observar que as poucas ocorrências de trabalho que apoiam o treinamento do usuário estão distribuídas entre as fases de teste de PLANEJAMENTO (2/5), ANÁLISE (1/5) ou MÚLTIPLAS (2/5).

Do terceiro cruzamento (Figura 4.11), observa-se que o apoio ao treinamento no uso da visualização ocorre na proporção de: 1/5 para o nível de INTEGRAÇÃO, 2/9 para o

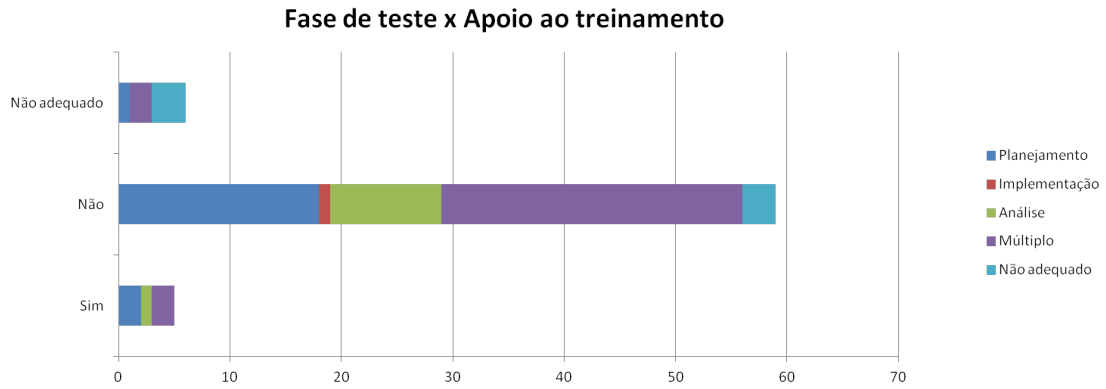

Figura 4.10: Cruzamento entre critérios Fase de teste e Apoio ao treinamento.

nível de SISTEMA e 2/34 para MÚLTIPLOS níveis de teste. Nenhum apoio ao treinamento foi identificado nos trabalhos orientados ao nível unitário.

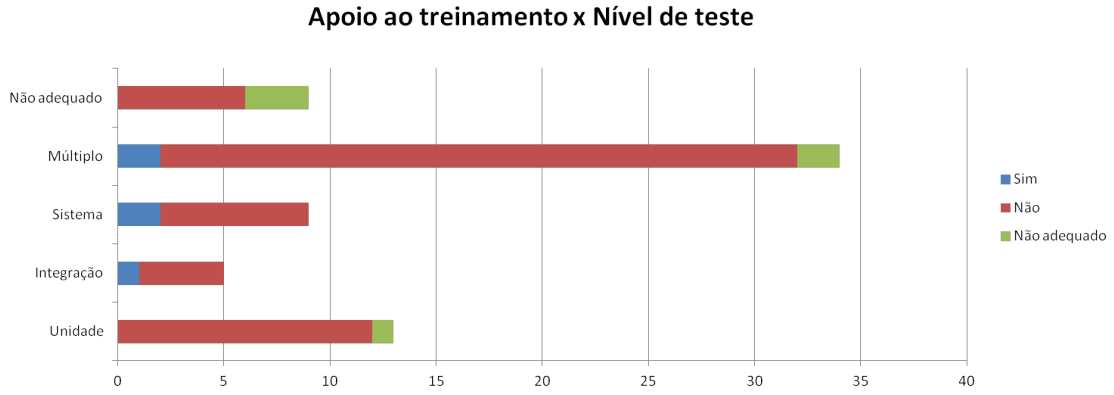

Figura 4.11: Cruzamento entre critérios Apoio ao treinamento e Nível de teste.

Pelo quarto cruzamento (Figura 4.12) observa-se que somente 1/5 das ocorrências na técnica FUNCIONAL e 4/47 das ocorrências que não são orientadas a qualquer critério de teste específico, oferecem suporte ao treinamento.

Observando-se os resultados obtidos pelos cruzamentos estabelecidos para cada aspecto, nossa interpretação e resposta para a *RQ*.2 é resumida a seguir:

No que diz respeito ao aspecto de teste, observa-se que o número de publicações cujas visualizações são dedicadas a apoiar a busca de erros nas dependências entre os módulos de software ("sistema" e "integração") ainda é moderado (14/64). Embora as propostas orientadas a "múltiplos" níveis de teste possam fornecer alguma facilidade nesse tipo de análise, é importante notar que muitas delas foram classificadas nesta categoria porque não apresentavam características que permitissem categorizá-las em

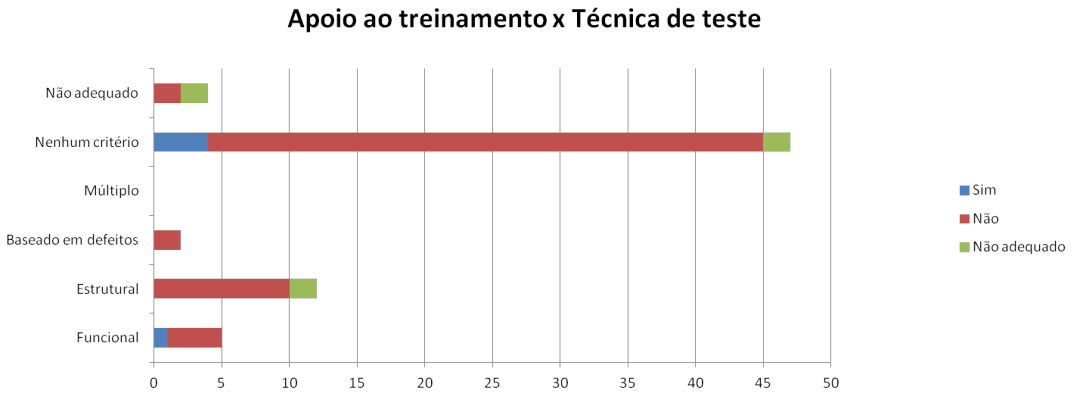

Figura 4.12: Cruzamento entre critérios Apoio ao treinamento e Técnica de teste.

um nível de teste específico. Em alguns casos, os artigos foram classificados como orientados a "múltiplos"níveis de teste, simplesmente porque permitiam visualizar, além dos módulos, alguma representação da interface entre os módulos, o que ainda era melhor do que a análise pura de informações textuais referentes a essas interfaces. No entanto, isso não foi suficiente para classificá-las como propostas dedicadas à detecção de erros decorrentes de interface ou sistema. Essa conclusão foi de certa maneira reveladora, uma vez que a visualização poderia ser empregada para facilitar esse tipo de análise para os testadores.

Os resultados dos critérios nível de teste e fase de teste permitem observar a generalidade dos trabalhos, isto é, atendendo majoritariamente ao valor "múltiplo"em ambos os critérios de classificação. Outro ponto é que existem mais propostas de visualização relacionadas à fase de "planejamento" dos testes do que trabalhos relacionados à fase de "análise" dos resultados. Combinando essa informação com o fato de que os trabalhos orientados a algum nível de teste também possuem mais ocorrências nas propostas de "planejamento"dos casos de teste do que de "análise"dos resultados, percebe-se que o uso da visualização como mecanismo para auxiliar a definição de casos de teste é uma questão mais investigada do que o uso da visualização para auxiliar a análise dos resultados gerados pela execução dos mesmos.

A ocorrência de um único artigo orientado à fase de "implementação" dos testes também é digna de nota. Considerando-se que este é o estágio que provavelmente requer maior esforço manual dos testadores para geração de artefatos de testes de software, nota-se uma lacuna na pesquisa de visualizações interativas para esta fase de testes que permitam melhorar a produtividade do testador.

Com relação ao aspecto visualização, as tradicionais interfaces "bidimensionais baseadas em interação via cursor" são características em quase todas as propostas de

visualização no tema. Um possível fator que pode estar relacionado à baixa adoção do substrato tridimensional é que as propostas são comumente orientadas a adequarem-se nos ambientes de desenvolvimento existentes, os quais são em geral bidimensionais e orientados à codificação. Considerando-se o advento das telas touchscreens como uma forma de interação alternativa nos últimos anos — e que nenhuma ocorrência nessa direção foi observada mesmo no substrato bidimensional para o tópico investigado — observa-se uma oportunidade de investigar as vantagens e desvantagens de explorar esse tipo de interação na exploração de visualizações orientadas ao teste de software.

Outro ponto a ser considerado é que não foi encontrada nenhuma proposta de visualização que declarasse estar baseada em um modelo de referência, conforme definição de Card, Mackinlay e Shneiderman (1999a) e Roberts (1999). De acordo com Robertson e Ferrari (1994) " [...] a tarefa de gerar visualizações padrões e robustas de dados em uma investigação exploratória ou direcionada, ou de automatizar a produção de tais padrões e ajudar o usuário a refiná-los, baseia-se em ter representações padrões que satisfaçam critérios estabelecidos ou especificados para interpretação ". Nesse sentido, um modelo de referência seria útil por viabilizar uma maneira adequada de tornar as propostas de visualização mais comparáveis. Além disso, o modelo de referência caracteriza os componentes da visualização em uma perspectiva relevante e comum, além de generalizar a descrição do processo de definição da visualização.

Em relação ao terceiro aspecto, quase todas as ferramentas ou técnicas de visualização propostas não fornecem apoio ao treinamento dos usuários em seus usos. Considerando que seria complicado ensinar o uso de uma técnica de visualização sem a existência de uma ferramenta associada para apoiá-la, a falta de suporte de treinamento nos trabalhos que propõem somente técnicas é considerado compreensível. No entanto, os trabalhos que propõem somente ferramentas ou ambas ferramentas e técnicas (múltiplos) também apresentaram poucos trabalhos que ofereciam "apoio ao treinamento" no uso da visualização. Destaca-se a importância da proposta apoiar o treinamento no uso da própria visualização, considerando que este pode ser um fator decisivo na adoção das ferramentas. Também é uma maneira de antecipar fatores inesperados e práticos que podem prejudicar o uso das visualizações (por exemplo, problemas relacionados à interação e metáforas confusas).

A terceira questão de pesquisa *(RQ.3)* aborda: *"Como as ferramentas e técnicas de visualização, que suportam a atividade de teste de software, foram avaliadas?"*

Para responder esta questão de pesquisa, foram considerados os cruzamentos dos seguintes critérios de classificação: CC.4 vs. CC.3 (contexto de avaliação vs. foco); CC.4 vs. CC.2 (contexto de avaliação vs. fonte); CC.4 vs CC.5 (contexto de avaliação vs. nível de teste) e CC.4 vs CC.7 (contexto de avaliação vs. técnica de teste).

Inicialmente, considerando apenas o critério contexto de avaliação, é possível observar que a maioria dos trabalhos foram classificados em POBREMENTE AVALIADOS/NÃO DETALHADOS (27/64). Considerando apenas os estudos avaliados adequadamente, 21/35 foram avaliados em ambiente ACADÊMICO, 13/35 na INDÚSTRIA e 1/35 em MÚLTIPLO contexto. Além disso, 2/64 documentos foram classificados como NÃO ADEQUADOS para o critério contexto de avaliação.

Do cruzamento dos critérios de classificação CONTEXTO DE AVALIAÇÃO e FOCO (Figura 4.13), observa-se que as ferramentas foram, em sua maioria, classificadas como POBREMENTE AVALIADAS/NÃO DETALHADAS (13/25) e as demais ocorrências foram avaliadas na INDÚSTRIA e na ACADEMIA na mesma proporção (6/25 cada). As propostas que contemplam ferramenta e técnica (MÚLTIPLAS) foram classificadas em sua maioria como avaliadas na ACADEMIA (9/19) e POBREMENTE AVALIADAS/NÃO DETALHADAS (7/19). Ainda nessa categoria, apenas 1/19 dos trabalhos foi avaliado na INDÚSTRIA e 2/19 classificado como NÃO ADEQUADOS para o critério CONTEXTO DE AVALIAÇÃO. Dentre as propostas exclusivamente orientadas a TÉCNICAS, 7/17 foram consideradas como POBREMENTE AVALIADAS/NÃO DETALHADAS, 5/17 avaliadas na INDÚSTRIA e 4/17 avaliadas na ACADEMIA. O único trabalho que reporta MÚLTIPLOS contextos de avaliações (academia e indústria) também foi registrado como uma proposta de TÉCNICA (1/17). ESTUDOS EXPERIMENTAIS foram aqueles que tiveram a menor ocorrência: 2/64 avaliados na ACADEMIA e 1/64 avaliado na INDÚSTRIA.

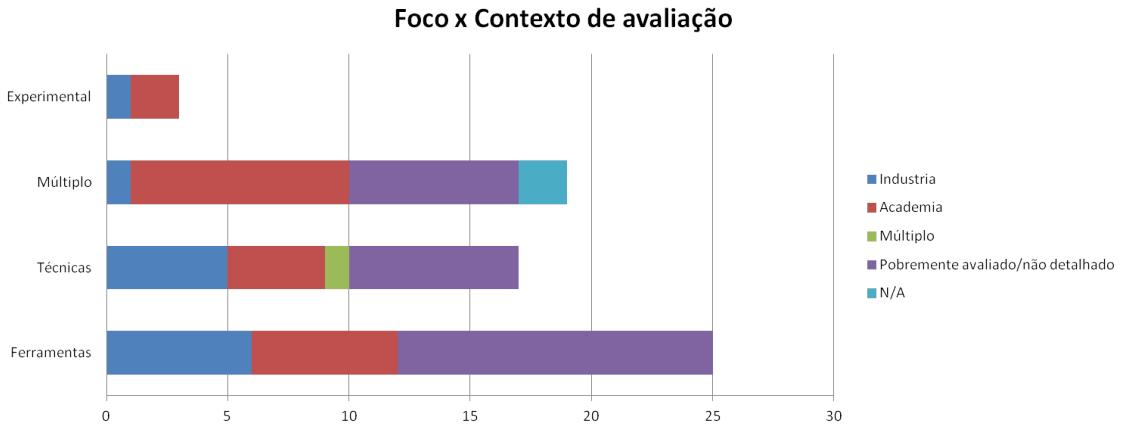

Figura 4.13: Cruzamento entre critérios Foco e Contexto de avaliação.

Cruzando-se os critérios CONTEXTO DE AVALIAÇÃO e FONTE (Figura 4.14), nota-se para os trabalhos publicados na IEEE que 18/42 foram avaliados na academia e 16/42 foram POBREMENTE AVALIADOS. A avaliação na INDÚSTRIA corresponde a 7/42 artigos. Por outro lado, os trabalhos publicados pela ACM foram avaliados na INDÚSTRIA, ACADEMIA ou POBREMENTE AVALIADOS/NÃO DETALHADOS respectivamente

na seguinte proporção: 5/16, 3/16 e 6/16. Na SCIENCE DIRECT, a maior parte dos trabalhos foram classificados em POBREMENTE AVALIADOS/NÃO DETALHADOS (5/6) e houve uma única ocorrência de trabalho avaliado na INDÚSTRIA. Os outros estudos (2/64), foram publicados pela ACM e IEEE e foram classificados, respectivamente, como NÃO ADEQUADOS quanto ao critério CONTEXTO DE AVALIAÇÃO.

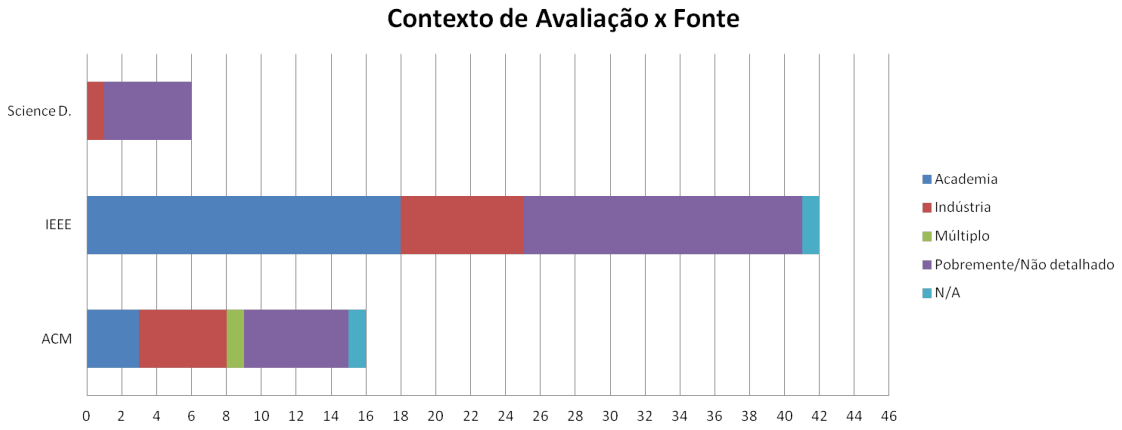

Figura 4.14: Cruzamento entre critérios Contexto de avaliação e Fonte.

Com relação ao cruzamento entre os critérios CONTEXTO DE AVALIAÇÃO e NÍVEL DE TESTE (Figura 4.15) nota-se que a maioria das propostas de visualização relacionadas ao teste UNITÁRIO foram avaliadas na ACADEMIA (7/12). Ainda para o nível UNITÁRIO, propostas avaliadas na INDÚSTRIA, e POBREMENTE AVALIADAS/NÃO DETALHADAS contabilizam, respectivamente, 3/12 e 2/12. Publicações relacionadas ao nível de teste SISTEMA avaliadas na INDÚSTRIA e POBREMENTE AVALIADAS/NÃO DETALHADAS contabilizam, respectivamente, 5/9 e 4/9 publicações. O nível de teste INTEGRAÇÃO é o nível de teste menos considerado nas propostas. Para este nível, apenas 3/5 dos estudos foram avaliados na ACADEMIA e 2/5 foram POBREMENTE AVALIADOS/NÃO DETALHADOS. A maioria das propostas foram classificadas como orientadas a (MÚLTIPLOS) níveis de teste (32/64), e os resultados para o critério contexto de avaliação nessa categoria foram: 17/32 para POBREMENTE AVALIADOS/NÃO DETALHADOS, 9/32 para avaliação na ACADEMIA, 4/32 para avaliação na INDÚSTRIA e 1/32 foi classificado como NÃO ADEQUADO para este critério. Poucos artigos foram classificados como NÃO ADEQUADOS com relação ao nível de teste (6/64). Dentre estes, 2/6 foram avaliados na ACADEMIA, 1/6 na INDÚSTRIA, 2/6 foram POBREMENTE AVALIADOS/NÃO DETALHADOS e 1/6 foi considerado NÃO ADEQUADO.

Para o último cruzamento considerado na terceira questão de pesquisa (CONTEXTO DE AVALIAÇÃO e TÉCNICA DE TESTE), os trabalhos que não foram endereçados a NENHUM critério de teste específico, se subdividem da seguinte forma:

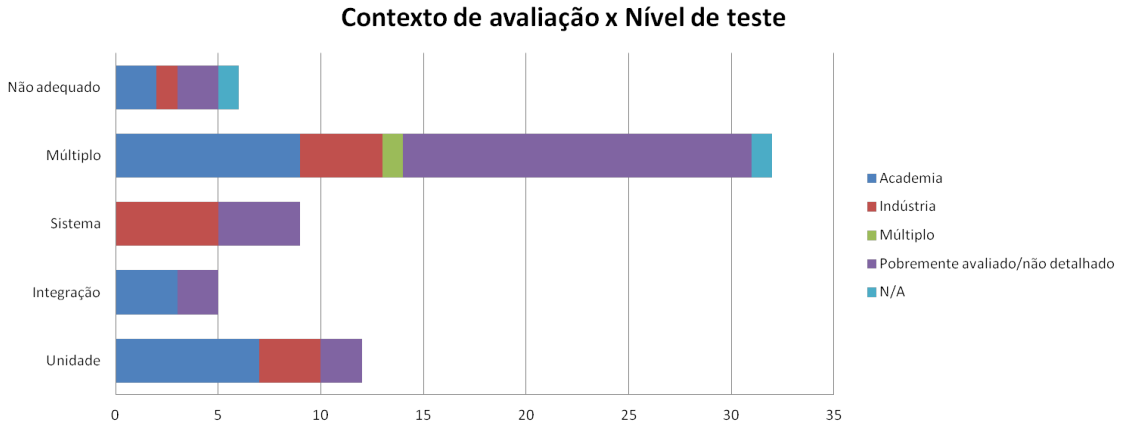

Figura 4.15: Cruzamento entre critérios Contexto de avaliação e Nível de teste.

19/45 foram POBREMENTE AVALIADOS/NÃO DETALHADOS, 14/45 foram avaliados na ACADEMIA, 10/45 foram avaliados na INDÚSTRIA e 1/45 foi avaliado em MÚLTIPLOS CONTEXTOS (academia e indústria). As propostas orientadas aos critérios ESTRUTURAIS, FUNCIONAIS e BASEADOS EM DEFEITOS contribuíram, respectivamente, com 10/64, 5/64 e 2/64 ocorrências. Para estes critérios, apenas um trabalho avaliado na INDÚSTRIA foi encontrado para cada categoria. Dentre as propostas que abordam critérios ESTRUTURAIS, 4/10 foram avaliados na ACADEMIA e 5/10 foram POBREMENTE AVALIADOS/NÃO DETALHADOS. Dentre as propostas que abordam critérios FUNCIONAIS 2/5 foram avaliados na ACADEMIA e 2/5 foram POBREMENTE AVALIADOS/NÃO DETALHADOS. Um artigo foi considerado NÃO ADEQUADO para ambos critérios de classificação. Um artigo foi considerado não adequado para o critério de classificação “técnica de teste” mas foi avaliado na ACADEMIA.

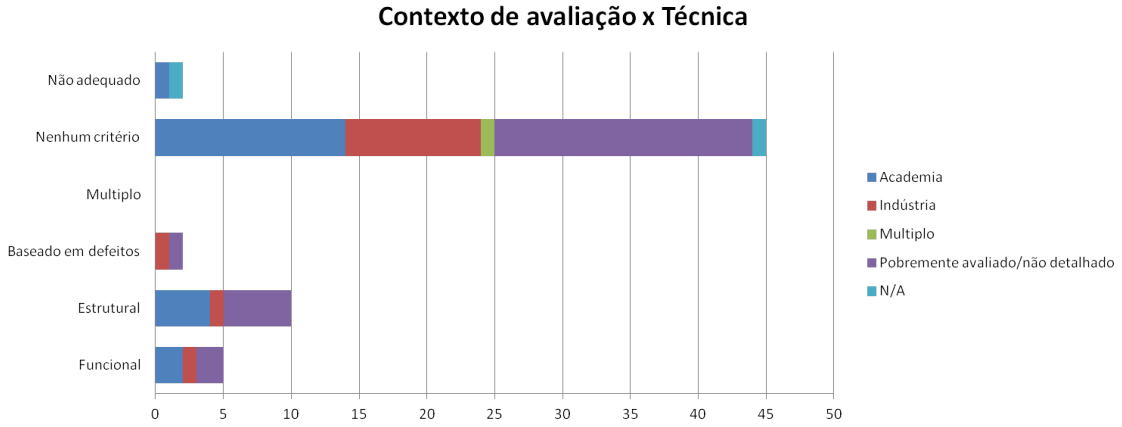

Figura 4.16: Cruzamento entre critérios Contexto de avaliação e Técnica de teste.

Observando-se os resultados obtidos de todos esses cruzamentos, nossa interpretação e resposta para a *RQ.3* é resumida a seguir:

A quantidade de propostas de ferramentas e de técnicas de visualização é, em geral, balanceada na área. Poucos trabalhos, no entanto, são conduzidos especificamente para fins experimentais (3/64). Mesmo os estudos não experimentais não possuem resultados de avaliação suficientes que demonstrem sua efetividade, considerando a alta taxa de detecção de trabalhos "pobremente avaliados/não detalhados". Na verdade, esta é a maior categoria para o critério contexto de avaliação. Uma vez que apenas trabalhos científicos foram considerados, havia uma esperança de que esta seria a categoria com o menor número de ocorrências ou, pelo menos, não seria a maior

A maior quantidade de propostas avaliadas na academia em relação à indústria foi um resultado esperado, considerando-se os altos custos e riscos geralmente envolvidos na avaliação em ambientes não acadêmicos. De cada biblioteca digital, pode-se notar que a IEEE, proporcionalmente e em valores absolutos, possui mais contribuições avaliadas na academia do que a ACM (42,86% vs. 18,75%). A ACM, por sua vez, possui proporcionalmente mais avaliações na indústria do que a IEEE (31,25% vs. 16,67%), embora os resultados sejam muito próximos quando considerados em valores absolutos (5 na ACM e 7 na IEEE). A Science Direct, além de menos trabalhos publicados, também demanda mais atenção quanto à avaliação dos trabalhos, considerando que apenas um documento selecionado apresentou avaliação.

Considerando-se apenas as propostas realmente orientadas a apoiar critérios de alguma técnica (estrutural, funcional ou baseada em defeitos), a maior parte foi "pobremente avaliada/não detalhada", mesmo considerando-se o baixo número de propostas em cada categoria. Este é um problema que chama a atenção, uma vez que a maioria dos critérios de teste já encontravam-se definidos mesmo antes dessas publicações. Logo, os trabalhos não deixam claro como as visualizações favorecem ou comprometem a aplicação desses critérios. Conforme discutido adiante na Seção 6.3.1, o uso de visualização na aplicação dos testes nem sempre implica em vantagens para os resultados da atividade, quando comparada a abordagens não visuais.

Por fim, nota-se que uma fração considerável dos trabalhos avaliados na academia é orientada ao teste de unidade (7/21) ou a múltiplos níveis de teste (9/21). Os trabalhos avaliados na indústria encontram-se em menor número e distribuídos quase que uniformemente entre propostas orientadas ao teste unitário (3/13), teste de sistema (5/13) e múltiplos níveis de teste (4/13). No entanto, nenhuma proposta que abordasse o teste de integração foi detectada entre aqueles avaliados na indústria.

4.3 Ameaças de Validade

Nesta seção, fazemos as considerações de validade do estudo de mapeamento.

A primeira consideração é em relação às bibliotecas digitais consideradas. Devido a restrições de tempo e no número de pesquisadores disponíveis para conduzir o processo, outras bibliotecas digitais relevantes na área de engenharia de software (BRERETON et al., 2007) não puderam ser consultadas.

A janela de tempo considerada também foi uma decisão tomada com base nas restrições de tempo e recursos para condução do estudo. Embora estejamos conscientes que alguns trabalhos importantes anteriores ao ano 2000 podem ter sido ignorados, acreditamos que os trabalhos anteriores ao ano 2000 que tiveram evolução ou impacto dentro da área, contaram com novas publicações dentro da janela de tempo considerada.

As *strings* de busca consideradas também podem não representar a melhor combinação de palavras-chaves possível, apesar de nossos esforços aplicados nessa etapa do estudo: observar palavras-chaves em artigos relacionados ao tema; tentar várias combinações antes de decidir a mais apropriada; checar a presença dos artigos de controle no retorno da busca feita com as *strings*.

Outro ponto são os critérios de classificação considerados. Eles foram definidos com base em características relevantes dos artigos de controle e com base no conhecimento prévio dos pesquisadores nas áreas de teste de software e visualização. Todavia, eles podem não ter sido suficientes em abranger todas as propriedades necessárias para caracterizar o tema. Independentemente de nossa confiança na correta caracterização do tema, nós entendemos que importantes características podem ter sido ignoradas.

Por fim, o processo de seleção e de classificação do estudo, sujeito à interpretação e julgamento humano, pode ser considerado uma importante fonte de viés. Infelizmente, não há forma de se evitar esse risco, considerando-se a natureza interpretativa desse tipo de estudo. Logo, tentamos minimizar esse tipo de risco descrevendo detalhadamente o processo empregado no estudo. De acordo com Wohlin (2012), quanto mais um estudo experimental é repetido, considerando-se diferentes contextos, mais informações são obtidas sobre o objeto de estudo, tornando os resultados mais significantes. Desta forma, acreditamos que a descrição detalhada do processo, habilitando a replicação do estudo de maneira próxima à condução original, ainda é a melhor prática para minimizar a probabilidade de ocorrência de vieses dessa natureza.

4.4 Considerações Finais

Neste capítulo foi descrito um estudo de mapeamento sistemático conduzido para entender como as ferramentas e técnicas de visualização têm sido propostas e aplicadas na melhoria da atividade de teste. Três questões de pesquisa orientaram essa investigação.

A primeira questão relaciona-se à evolução das publicações relacionadas ao tema, em três importantes bibliotecas de pesquisa para a área da computação. Com relação a essa questão de pesquisa, pode-se observar como o número de publicações oscilou na janela de tempo considerada. A *IEEE* corresponde à biblioteca digital com o maior número de publicações no tema ao longo dos anos, seguida respectivamente, pela *ACM e Science Direct*.

A segunda questão de pesquisa refere-se à caracterização do perfil das propostas, com relação aos aspectos de teste, visualização e treinamento. Para o aspecto de teste, alguns dos resultados de interesse são: (i) poucas propostas são orientadas a apoiar o teste de integração e sistema; (ii) a maior parte das propostas não são orientadas a um nível de teste específico; (iii) existe um maior enfoque das propostas de visualização em problemas relacionados ao planejamento dos testes do que na análise dos resultados de execução dos testes. Para o aspecto de visualização observou-se que a tradicional forma de representação/interação visual ”bidimensional-baseada-em-cursor” corresponde à abordagem empregada na maioria das propostas. A investigação do terceiro aspecto também revela que pouco apoio ao treinamento tem sido provido pelas ferramentas propostas nos trabalhos.

A terceira e última questão relaciona-se à avaliação apresentada nos trabalhos. Embora exista um certo equilíbrio entre o número de ferramentas e técnicas publicadas na área, poucos estudos experimentais foram observados. Além disso, um grande número de trabalhos não fornece informações suficientes de avaliação das ferramentas ou técnicas propostas. Poucos trabalhos são orientados a apoiarem a aplicação ou verificação de algum critério de teste específico. Mesmo dentre estes, apenas uma parte provê validação satisfatória isto é, capaz de mostrar os benefícios/riscos oferecidos pelo uso da visualização em relação às soluções que não usam o recurso visual proposto. Considerando-se os trabalhos avaliados em contexto acadêmico, a maioria das avaliações são orientadas a múltiplos níveis de teste ou ao teste de unidade. As avaliações no contexto da indústria apresentaram-se em menor número e distribuída uniformemente entre os diferentes níveis de teste, exceto pelo teste de integração, no qual nenhuma ocorrência pôde ser encontrada. Dentre as bibliotecas consideradas, *IEEE e ACM* são numericamente próximas quanto ao número de trabalhos avaliados na indústria. A *IEEE*, numericamente e proporcionalmente, é a biblioteca que provê mais contribuições avaliadas no contexto acadêmico. A

Science Direct teve, por sua vez, poucas contribuições nesse tema e apenas um dos artigos selecionados nesta biblioteca apresentou avaliação adequada da proposta.

Outro objetivo deste mapeamento foi habilitar a replicação/expansão do estudo. As referências para os artigos e artefatos gerados nesse estudo podem ser acessadas em (PRADO; VINCENZI; NASCIMENTO, 2014). A replicação deve ajudar a reforçar os resultados obtidos e estimular novos estudos no tema.

CAPÍTULO 5

Aspectos Humanos no Teste de Software Unitário

5.1 Considerações Iniciais

Neste capítulo é apresentado o problema da carência de pesquisas de ferramentas de teste que considerem aspectos humanos em sua definição. Para tanto, foram analisados estudos da literatura que apontam problemas reais envolvendo praticantes atuantes na área. Além disso, foram identificadas na literatura, evidências de problemas associados à falta de suporte cognitivo oferecido pelas ferramentas no nível de teste unitário. Como resultado, um framework conceitual foi proposto, com o intuito de organizar e direcionar os passos seguintes do desenvolvimento da pesquisa.

5.2 A Ausência do Testador na Pesquisa de Ferramentas de Teste de Software

A importância de se considerar o aspecto humano no teste de software, pode ser ilustrada na seguinte metáfora, inspirada no homem e sua condição natural de caçador:

> *Imagine que profissionais de teste sejam caçadores e que bugs sejam caças. Considere que as ferramentas de teste sejam armas ou armadilhas — e que as técnicas de teste fundamentam o uso desse armamento na forma de estratégias e táticas de caça. Para que possam ser bem sucedidos, os caçadores precisam apanhar o maior número de presas possível, em especial aquelas de maior valor. Um caçador que confie apenas em sua força natural ou habilidades humanas não será bem sucedido — armas especializadas e armadilhas são necessárias para ter sucesso. Mas se essas ferramentas forem difíceis de usar ou não forem projetadas para potencializar as habilidades do caçador, elas podem se tornar dificultosas, demandar muito tempo para dominar ou no pior caso, proverem falsas expectativas. Trocar pessoas por*

> *armas automatizadas é uma alternativa pouco adaptável e deficiente em encontrar presas mais ariscas e valiosas.*

No trabalho de Ammann e Offutt (2008) os autores destacaram com clareza a natureza imprevisível da ocorrência de *bugs* no software e a dificuldade natural de se controlar a ocorrência desses problemas. De acordo com os autores "[...] falhas no software são enganos de projeto. Elas não aparecem espontaneamente, mas ao invés disso existem como resultado de alguma decisão (desafortunada) cometida por um ser humano". Essa observação embora trivial, não se aplica somente à concepção corriqueira de software como produto fim — as suítes de teste, *frameworks* e outras ferramentas de desenvolvimento também são um tipo de software. Logo, o desenvolvimento e a operação destes tipos de softwares podem influenciar e serem influenciados por decisões humanas.

Em linhas gerais, o teste de software pode ser entendido como um tipo de análise dinâmica aplicada sobre um artefato de software em busca de erros. Ao longo dos anos, esforços de pesquisa têm almejado o desenvolvimento de ferramentas que suportem a aplicação de técnicas, critérios e heurísticas de teste. O objetivo maior tem sido transformar o problema da realização dos testes em uma solução factível e automática (ou semi-automática) para revelação de falhas. Esta abordagem parece ter encorajado a crença de que soluções significativas de teste surgiriam a partir de máquinas, para serem executadas por máquinas. Paralelamente, o papel fundamental do componente humano neste ciclo parece ter sido esquecido ou minimizado.

Orso e Rothermel (2014) analisaram o estado da arte no teste de software ao longo da última década e meia. Nesse estudo, eles sumarizaram inúmeros temas de pesquisa e apresentaram desafios a serem investigados. Os dados provieram de questões abertas feitas a cinquenta pesquisadores de teste. Embora os autores apresentem uma variedade de ideias na seção "desafios e oportunidades" de seu artigo, eles são categóricos em afirmar que não consideram "fatores humanos" e "transferência de tecnologia" como desafios de pesquisa, apesar de terem declarado que estes temas estavam entre os mencionados nas respostas de seus pares. Além disso, em uma subseção dedicada a "Estudos empíricos e suporte para os mesmos", Orso e Rothermel (2014) mencionam a necessidade de "estudos com usuários", a predominância de "automação de algoritmos e heurísticas" na pesquisa de teste e preocupações sobre ameaças de validade aos quais os resultados de pesquisa estão sujeitos quando caem nas "mãos de engenheiros". Eles também notaram a influência que contribuições advindas da indústria tiveram na pesquisa (uma transferência de tecnologia reversa). Eles mencionam o JUnit (BECK, 2004) — "um *framework* simples e direto" — como um exemplo dessa influência. Entretanto, a exclusão dos fatores humanos e da transferência de tecnologia como "desafios" na pesquisa em teste são inconsistentes com a discussão feita ao longo do artigo. Além disso, Kasurinen, Taipale e

Smolander (2010) demonstram que os resultados de pesquisa envolvendo automatização não têm sido adotados na prática, revelando uma desconexão entre indústria e pesquisa.

Mais exemplos dessa separação entre fatores humanos e pesquisa em teste podem ser observados em Daka e Fraser (2014). Nesse estudo, os autores conduziram um *survey* com testadores para investigar a prática atual de teste unitário. Eles descobriram que a principal motivação para a condução do teste é a própria convicção do testador, a qual excede as "exigências que os gestores impõem sobre os testadores". Por outro lado, apenas metade dos participantes do *survey* "atribuem um sentimento positivo à escrita de testes unitários". Estes resultados estão alinhados com os de Ng et al. (2004) e com os do *survey* realizado por Lee, Kang e Lee (2012). No estudo de (NG et al., 2004), a "dificuldade de utilizar [ferramentas]" foi identificada como a maior barreira na adoção de uma ferramenta, e consequentemente dos métodos de teste associados. No último, mais da metade dos participantes responderam que o teste de software é realizado com base em conhecimento pessoal e sem qualquer orientação. Como indicado nos resultados, apesar de ser uma atividade desagradável, a resiliência na caça aos *bugs* pode ser motivada pela crença genuína em seu propósito.

Embora o trabalho de Jia e Harman (2011) chame a atenção para problemas associados ao ser humano na atividade de teste, atribui-se pouca consideração à participação humana nas soluções de teste. No trabalho de Jia e Harman (2011) os autores revisam a literatura de teste de mutação e discutem direções de pesquisas promissoras. De acordo com os autores, a "barreira para a ampla aplicação do teste de mutação"consiste no "problema dos mutantes equivalentes". Além disso, o "problema do oráculo humano" — a verificação, feita por um humano, da saída do programa original para cada caso de teste gerado — é também citado como outra barreira. Por fim, os autores apontam que "mais ferramental é necessário para assegurar uma "adoção generalizada da indústria" e que "ferramentas práticas e automatizadas" de geração de caso de teste seriam um caminho promissor para a criação de soluções de teste. Ao longo do artigo, os autores discutiram conexões passadas e futuras entre a automação e o teste de mutação. Todavia, pouca ou nenhuma menção foi feita sobre a inserção do ser humano nesse ciclo como parte da solução. Considerando que o problema do mutante equivalente é reconhecidamente um problema indecidível e que a geração de caso de teste ainda requer um oráculo humano não mecanizado para análise da execução e saídas geradas, é difícil supor que a automação será uma panaceia no amplo espectro de domínios de desenvolvimento de software existentes.

Em suma, esses trabalhos mostram uma divisão entre pesquisa e prática e trazem elementos para reflexão. Os problemas identificados nesses trabalhos mostram que a omissão dos fatores humanos e da transferência de tecnologia observadas, por exemplo, no trabalho de Orso e Rothermel (2014), não é de nenhuma forma uma ocorrência isolada.

Pelo contrário, a comunidade de pesquisa de teste como um todo precisa clamar *mea culpa*: estamos atacando os problemas considerando os *stakeholders* principais, aqueles que lidam com o teste a maior parte do tempo? Se por um lado estamos pesquisando "armas" poderosas, por outro, elas estão satisfazendo ou facilitando o ato de caça aos *bugs*? Se queremos potencializar as habilidades dos testadores e tornar mais fácil seu papel crítico no teste de software, como podemos melhorar suas ferramentas de forma a serem mais ergonômicas e efetivas cognitivamente?

5.3 A Relação Entre Casos de Teste de Qualidade e o Testador

Retomando a metáfora utilizada no início do capítulo, a qualidade do disparo feito por um caçador é o que determina se um alvo é atingido ou não. Nesse sentido, um bom disparo pode ser classificado como aquele que é intencional, planejado e capaz de alcançar seu alvo. Para se tornar um bom caçador, um indivíduo precisa melhorar suas habilidades em produzir disparos de qualidade. Uma equivalência pode ser estabelecida com os casos de teste no teste de software: um caso de teste incorpora e traduz a intenção do testador. Ele é planejado de acordo com uma estratégia e é capaz de detectar erros dependendo se as assertivas passam ou não, isto é, se os casos de teste passam ou falham. Desta forma, focar a evolução das ferramentas na melhoria da qualidade dos casos de teste também implica em habilitar o testador a aprimorar sua capacidade de revelar falhas.

Conforme explicado no Capítulo 2, três principais níveis de teste são comumente considerados na definição dos casos de teste: teste unitário, teste de integração e teste de sistema.

Na prática, o teste de integração e de sistema diferem-se do teste unitário com relação a quem é atribuída a responsabilidade de gerar os casos de teste. Considerando-se que o teste unitário requer um conhecimento de fina granularidade do sistema, a sua realização por uma equipe de teste poderia tornar-se cara e ineficiente. Além disso, considerando-se o grande número de unidades a serem testadas, o ciclo de teste tornaria-se longo: (i) o desenvolvedor implementaria a unidade; (ii) o testador receberia e entenderia a unidade, implementaria o caso de teste e tentaria relevar erros no mesmo; (iii) uma vez revelado um erro, o testador entregaria o erro de volta para o desenvolvedor resolvê-lo. Ao invés disso, em geral atribui-se ao desenvolvedor a responsabilidade por desenvolver e testar as unidades — ou testá-las e desenvolvê-las no caso de *Test Driven Development* (BECK, 2002). Desta forma, a equipe de teste fica livre para se concentrar nos problemas de integração ou de sistema, enquanto os desenvolvedores aplicam o seu conhecimento minucioso na realização do teste de unidade. Por um lado, a transferência

da responsabilidade pelos testes de unidade para o desenvolvedor é vantajosa para os testadores e gestores. Por outro, ela arrisca divergir os recursos mentais do desenvolvedor das atividades inerentes ao desenvolvimento, sua responsabilidade primária.

5.4 Problemas Cognitivos na Atividade de Teste Unitário

Nesta seção, encontram-se resumidas as evidências de demandas cognitivas identificadas na literatura de teste unitário. Nós agrupamos as demandas identificadas em três dimensões:

- Problemas de Compreensão Testador-Artefatos
- Problemas de Direcionamento e Orientação ao Testador
- Problemas de Usabilidade e Interação com as Ferramentas

5.4.1 Problemas de Compreensão Testador-Artefatos

Daka e Fraser (2014) relataram os resultados de um *survey* sobre teste unitário conduzido com desenvolvedores de software de diferentes países. O propósito do estudo foi alinhar a pesquisa atual com práticas comuns existentes e identificar oportunidades de pesquisa vindas da indústria. Os resultados da segunda questão de pesquisa *(RQ2)* mostram que os desenvolvedores tendem a tratar os testes falhos como defeitos causados pelo próprio caso de teste com maior frequência do que como defeitos causados pelo código testado. Eles também reportaram *(RQ4)* que os dois principais usos da geração automatizada de teste pelos desenvolvedores são: (i) automação como complemento para o teste manual e (ii) descoberta de falhas do tipo "*crash*" isto é, falhas simples, que não exigem interpretação aprofundada da especificação para ser identificada. Estes resultados indicaram que tanto o teste manual quanto o teste automatizado são práticas atuais e não-exclusivas. Além disso, o teste manual se mostra dominante em áreas com erros menos óbvios (isto é, do tipo "*non-crash*"). Conforme mencionado por Leitner et al. (2007), embora a geração automatizada exija pouco esforço, ela não pode substituir o teste manual porque os desenvolvedores são *experts* em definir dados de entrada complexos e, portanto, em definir casos de teste efetivos.

As ferramentas de teste do tipo xUnit são reconhecidamente dominantes na atividade de teste unitário na atualidade. Um exemplo que se destaca é a JUnit. Além da ampla adoção na indústria, a JUnit é também um exemplo de transferência de tecnologia reversa, uma vez que sua adoção *a posteriori* pela academia influenciou e auxiliou a pesquisa de teste de software (LAM et al., 2012). Entretanto, conforme observado por Atkinson, Barth e Brenner (2010), ferramentas como a JUnit ainda requerem que os testes sejam escritos na forma de programas, o que é benéfico por sua integração natural com o código testado.

Infelizmente, isso também cria dependências entre os detalhes do código do programa ou da linguagem e a estratégia de testar a lógica, os dados e os resultados gerados. Um exemplo seria testar os dados de entrada de uma unidade que dependa de bibliotecas da linguagem com a qual o testador não tenha familiaridade: além de conhecer a unidade testada, o testador pode precisar saber instanciar e inicializar classes e dependências dessas bibliotecas, assim como saber interpretar seus comportamentos, para configurar uma pré-condição para a unidade a ser testada. O trabalho de Lappalainen et al. (2010) está comprometido com questões similares no campo educacional. Baseando-se em dificuldades identificadas em desenvolvedores novatos que aprendiam concomitantemente teste de unidade e *TDD*, eles propuseram uma ferramenta para tornar os casos de teste mais legíveis e fáceis de escrever. Observações similares são feitas por Daka e Fraser (2014) ao perguntarem aos participantes de sua pesquisa "de que forma o teste unitário poderia ser melhorado" *(RQ5)*, em particular "o que torna difícil consertar um teste unitário que falha". Os resultados de interesse foram que: "o código sob teste é difícil de entender", "o teste por si só é difícil de entender" e que "os testes refletem comportamentos não realísticos".

5.4.2 Problemas de Direcionamento e Orientação ao Testador

Runeson (2006) conduziu um *survey* para caracterizar o teste unitário conforme ele ocorre na prática. O *survey* incluiu uma rodada de discussão grupo focal com participantes e outra rodada com suas respectivas companhias. Um dos resultados revelou que não havia consenso quanto ao que constitui uma unidade sob teste. Apesar da confiança demonstrada pelas respostas nos questionários em relação a essa questão, a rodada de grupo focal revelou que não era incomum considerar a unidade como um grupo de unidades coesivas relacionadas ao invés de uma unidade indissociável. Quase uma década depois, Daka e Fraser (2014) revelaram que dois dos aspectos mais difíceis relacionados à escrita de novos testes corresponde a (i) identificação de qual código testar e (ii) ao isolamento da unidade sob teste. Similarmente, os resultados de Runeson (2006) e Daka e Fraser (2014) revelaram dificuldades na avaliação da unidade de teste. O primeiro indicou a necessidade de um critério claro e mensurável e o segundo revelou que a determinação sobre "o que checar em um teste" era uma dificuldade comum em escrever novos testes. Tais preocupações no *survey* de Runeson (2006) e Daka e Fraser (2014) estão relacionadas à incerteza sobre "o que" e "como" realizar a atividade. Essas preocupações podem ter sido influenciadas por muitas causas tais como a falta de treinamento adequado, a falta de padronização na atividade, diferentes rigores demandados entre projetos e o *background* da equipe. O fato é que, independentemente do motivo, o teste unitário pode ainda estar sendo subestimado pelos testadores devido à sua aparente simplicidade

comparado a outros níveis de teste. Nesse sentido, o provimento de alguma *orientação* poderia contribuir para mitigar tais problemas.

Mesmo quando testadores sabem como criar um teste e o que testar, muitas vezes eles não têm *pistas de referência* para confiar. Lee, Kang e Lee (2012) realizaram um *survey* sobre práticas correntes e descobriram que 50% dos participantes "realizavam os seus testes baseados no próprio conhecimento individual ou pessoal sem uma orientação padronizada". Isso corresponde de forma muito próxima ao que *Daka* e *Fraser* encontraram na sua terceira questão de pesquisa, a qual abordava como os testes unitários eram escritos pelos desenvolvedores. De acordo com os mesmos, "os desenvolvedores clamam escrever teste unitário sistematicamente e medir a cobertura de código, mas não têm uma prioridade clara sobre o que torna um teste individual bom". Runeson (2006) reportou que devido à falta de estratégias nas companhias, os casos de teste eram definidos usando o julgamento pessoal dos desenvolvedores, "levando a práticas variadas".

Lee, Kang e Lee (2012) revelaram que o uso de métodos associados a ferramentas no teste unitário é especialmente fraco, comparado a outros níveis de teste. De fato, os *frameworks* de teste unitário não exigem a aplicação de qualquer estratégia ou critério de teste em particular. Nesse sentido, eles são como uma folha de papel em branco na qual desenvolvedores podem definir os casos de teste da sua própria maneira. Além disso, Garousi e Zhi (2013) e Ng et al. (2004) concordam que o teste funcional é uma abordagem mais popular que o teste estrutural no teste de software como um todo. Considerando que testes funcionais se baseiam somente nas especificações para derivar e avaliar os requisitos de teste, esse tipo de teste é mais suscetível a interpretações e subjetividade do que outras técnicas. Além disso, o teste funcional "fim a fim"via interface gráfica requer bastante manutenção, o que implica em melhorias no teste unitário subjacente (FOWLER, 2015). Logo, é razoável questionar se aplicação indiscriminada de teste funcional no nível unitário estaria levando a práticas improdutivas, tais como a geração de um alto número de casos de teste de baixa eficácia.

5.4.3 Problemas de Usabilidade e Interação com a Ferramenta

As duas últimas oportunidades de investigação relacionadas ao suporte cognitivo para o teste são a aparente falta de facilidade no uso e apreciação das ferramentas de teste. Runeson (2006) verificou a dificuldade em motivar os desenvolvedores, como uma das fraquezas do teste unitário. Daka e Fraser (2014) corroboram a idéia, revelando que apenas metade dos desenvolvedores que eles pesquisaram, atribuiram um sentimento positivo à escrita do teste unitário. Eles também indicaram que a melhoria das ferramentas é um caminho para a resolução de tais problemas. Kasurinen, Taipale e Smolander (2009) revelaram em seu estudo, dedicado ao entendimento de problemas de teste de software

na prática, que a usabilidade e aplicabilidade das ferramentas de teste foi identificada pelas companhias pesquisadas como um dos fatores que afetam a eficiência da atividade de teste. Em particular, grandes companhias investigadas no estudo relataram que falsos positivos gerados pelo uso de ferramentas de teste complicadas e defeituosas adicionavam um *overhead* adicional na atividade. Wiklund et al. (2014) apontaram dificuldades relacionadas à configuração e uso do ambiente de desenvolvimento integrado *(IDE)* — Eclipse em particular — como uma barreira comumente enfrentada por testadores durante a criação dos seus códigos de teste em ambientes de teste automatizados. Adicionalmente, Lee, Kang e Lee (2012) clamam que o relacionamento entre as ferramentas de teste e as atividades suportadas deve se tornar mais claro.

Delahaye e Bousquet (2015) definiram algumas *guidelines* para a comparação e seleção de ferramentas de engenharia de software. Eles escolheram a técnica de mutação para estudo de caso. Nesse estudo, eles segregaram o critério de usabilidade em quatro categorias: compatibilidade, *interface*, documentação e gestão de mutantes sobreviventes. Na categoria *interface*, eles destacaram a dificuldade encontrada pelos testadores na operação das ferramentas. Dentre as oito ferramentas de mutação estudadas, apenas duas possuíam uma *interface* gráfica *(GUI)* adequada. Adicionalmente, a maioria delas não se integravam com *IDEs* populares. Na "documentação" eles procuraram por descrições claras de operações e processos executados pelas ferramentas. Apenas três das oito ferramentas analisadas foram ranqueadas como tendo uma documentação "muito boa", enquanto cinco foram ranqueadas no nível mais baixo devido à baixa qualidade dos relatórios de mutantes sobreviventes gerados.

5.5 Sintetizando as Necessidades do Testador em um *Framework* de Pesquisa

O suporte cognitivo para o teste pode ajudar a melhorar as dificuldades primárias encontradas no teste unitário. O colorimento de um caminho coberto em um grafo de fluxo de controle em uma ferramenta de teste estrutural é um exemplo simples de suporte cognitivo existente. Esse recurso ajuda o testador a manter o foco no que ainda resta ser coberto no código e o ajuda a lembrar qual parte já foi coberta. A barra de *status* verde/vermelha do JUnit é outro exemplo de suporte cognitivo que ajuda o testador a evitar ter que rastrear se os casos de testes mudaram o estado de suas assertivas, entre diferentes execuções de teste. Os problemas de teste unitário discutidos na Seção 5.4 indicam a exigência de um esforço mental não trivial do testador, apesar de qualquer assistência das soluções de teste automatizadas existentes. Com base nessa evidência, um *framework* foi proposto (Figura 5.1) combinando as três dimensões cognitivas discutidas.

O *framework* indica direções iniciais de investigação, para a evolução de ferramentas de teste unitário.

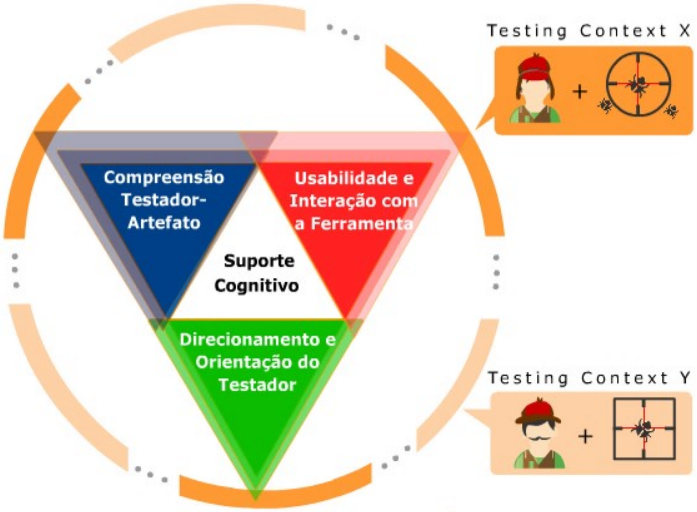

Figura 5.1: Representação gráfica do framework contendo as três dimensões de demandas cognitivas a serem satisfeita pelas ferramentas de teste unitário

- *Compreensão Testador-Artefato:* Testadores precisam raciocinar sobre casos de teste, unidades sob teste e relatórios de teste. Eles empregam esforço na compreensão destes artefatos, tanto tomados individualmente quanto em conjunto. Diferentes formas de reformular as informações dos artefatos podem contribuir para diminuir o custo associado a esta sobrecarga. Uma vez que estes problemas estão relacionados à compreensão do código de teste ou do código de programa, uma forma de abordar isso seria usando as contribuições providas pela literatura da psicologia da programação. Isso inclui a análise de modelos de como os humanos leem código e teorias sobre como o conhecimento sobre o código é construído e usado (DETIENNE, 2001).
- *Direcionamento e Orientação do Testador:* Testadores podem experimentar confusão ou desorientação, custando-lhes tempo e motivação, os quais por sua vez podem ocasionar em queda de rendimento e comprometimento com a atividade de teste. As ferramentas devem prestar assistência aos testadores no provimento de: direcionamento ao testador no seu fluxo de trabalho pessoal; ajuda ao testador a tomar decisões sob incertezas ou problemas ambíguos; ajustar o escopo das tarefas para ajudar o testador a focar nos objetivos da tarefa. Exemplos incluem mas não estão limitados a: eleger, ranquear, identificar e resumir tarefas de teste e artefatos. Isso envolve primeiro, entendimento sobre quais estratégias os humanos aplicam em tais

tarefas, tornando as teorias de tomada de decisão (SIMON et al., 1986) uma escolha natural.

- *Usabilidade e Interação com a Ferramenta:* Os testadores podem enfrentar dificuldades com a interface e outros desafios de usabilidade durante a instalação e operação das ferramentas. Isso contribui com o desprazer do testador com a realização dos testes (DAKA; FRASER, 2014), (RUNESON, 2006). Alguns exemplos incluem: disponibilidade de auto-instalação e auto-configuração das ferramentas; visibilidade das operações da ferramenta; *feedback* visual claro; operações alternativas da *interface*; documentação detalhada e atualizada. Vários princípios e diretrizes estão disponíveis na literatura de *IHC* para abordar tais problemas, conforme apresentado por Stone et al. (2005) e (NORMAN, 2013).

As dimensões cognitivas do *framework* podem ser relacionadas aos dois tipos gerais de cognição propostos por Norman (1994) — experiencial ou reflexiva. Considera-se que as dimensões "Compreensão Testador-Artefato" e "Direcionamento e Orientação do Testador" por exemplo, estejam mais orientadas à cognição reflexiva. Nesse sentido, elas são dimensões relacionadas aos passos e resultados intermediários produzidos durante a tarefa de teste, isto é, elas são orientadas ao suporte de atividades complexas e que demandam um raciocínio mais cauteloso do testador. A dimensão "Usabilidade e Interação com a Ferramenta", por outro lado, está mais próxima da cognição experimental. Logo, envolve aspectos auxiliares e ergonômicos, dando importância a como as características da ferramenta contribuem para uma experiência cognitiva mais prática e confortável durante a atividade de teste.

5.6 Considerações Finais

Neste capítulo, chama-se a atenção para o problema de como a pesquisa em teste de software não tem considerado a perspectiva do profissional de testes na evolução das ferramentas de teste. Ao ignorar ou minimizar a influência do testador na proposição de ferramentas nós, pesquisadores, estamos deixando em segundo plano aspectos importantes e úteis para esses profissionais.

As necessidades dos profissionais de teste unitário, sumarizadas a partir dos trabalhos identificados na literatura e descritas na Seção 5.4, consistem em demandas cognitivas genéricas que precisam ser investigadas na evolução das ferramentas. Essas demandas cognitivas foram condensadas em um *framework* composto de três dimensões cognitivas, conforme apresentado na Seção 5.5: Compreensão Testador-Artefato, Direcionamento e Orientação do Testador e Usabilidade e Interação com a Ferramenta.

Este capítulo destacou duas contribuições importantes deste trabalho. Primeiro, como ponto de partida para a investigação de melhorias para o teste unitário, ele desloca

o foco principal de pesquisa da automação em favor de um olhar voltado para as necessidades cognitivas do testador, uma mudança na perspectiva atual da literatura. A segunda contribuição é uma proposta de *framework* conceitual para o teste unitário, o qual caracteriza problemas cognitivos identificados nesse nível de teste. Nesse sentido, o *framework* indica direções de pesquisa a serem investigadas nos passos subsequentes do trabalho.

CAPÍTULO 6

Caracterização do Suporte Cognitivo Provido pelas Ferramentas de Teste Unitário

6.1 Considerações Iniciais

Conforme discutido nos capítulos anteriores o suporte cognitivo corresponde ao apoio que uma ferramenta presta aos usuários em suas tarefas mentais (WALENSTEIN, 2002). Considerando isso, quais podem ser as consequências se essas tarefas do usuário não forem adequadamente facilitadas por uma ferramenta existente? O que acontece quando estes usuários são na verdade, profissionais envolvidos no processo de garantia de qualidade de um produto tão complexo quanto o software? Quais as possíveis causas e consequências desse problema no teste unitário?

Antes de despender recursos e esforços propondo qualquer novo artefato para ajudar os testadores em sua caça aos *bugs*, é necessário inspecionar o vasto arsenal de ferramentas disponíveis e avaliar suas características. Neste capítulo, o domínio de ferramentas que aplicam a visualização no teste unitário foi eleito para tal análise, considerando-se o potencial natural da visualização para auxiliar humanos em problemas que envolvem a cognição. Para este propósito, investigou-se: como essas ferramentas foram planejadas; quais são as características e funcionalidades oferecidas; como elas têm sido avaliadas quanto à sua utilidade para auxiliar os profissionais de teste. Considerando-se o interesse no teste unitário, uma estratégia em três etapas foi aplicada: inicialmente, as ferramentas de teste unitário foram selecionadas a partir da literatura de visualização aplicada a testes; em seguida, o *framework* cognitivo proposto no Capítulo 5 foi aplicado para derivar, relacionar e analisar as visualizações. Por fim, foram sumarizadas e apresentadas as lacunas de suporte cognitivo e oportunidades identificadas nessas ferramentas.

6.2 Seleção das Ferramentas de Visualização Para o Teste Unitário

A visualização pode amplificar a cognição de diferentes formas. Card, Mackinlay e Shneiderman (1999b) classificaram seis categorias majoritárias: (i) aumentando os recursos de apoio ao processamento e memorização do usuário; (ii) reduzindo a busca por informação; (iii) melhorando o reconhecimento de padrões; (iv) habilitando operações de inferência perceptual; (v) monitoramento perceptual e (iv) codificação de informação em um meio manipulável. Dois conceitos da área de visualização precisam ser entendidos, de maneira a determinar se uma dada visualização é útil para o usuário: *expressividade*, a qual corresponde à capacidade da visualização de transmitir toda (e somente) a informação de interesse para o usuário e *efetividade*, a qual corresponde a quão claro o usuário entende a informação representada (MACKINLAY, 1986) (BESHERS; FEINER, 1993). Considerando estas definições, uma pergunta fundamental que deve ser feita ao propor qualquer nova visualização é: essa solução é *expressiva* e *efetiva* para o público alvo? Ambos os atributos estão intimamente relacionados ao suporte cognitivo que será provido. De acordo com (KOSSLYN, 1990), nós, humanos, visualizamos os objetos envolvidos em problemas diários, de forma a responder questões e encontrar soluções para os mesmos, ou seja, representamos mentalmente as imagens visuais associadas ao problema sendo solucionado. Agora, imagine as consequências se a visualização proposta para resolver um certo problema trouxer mais ou menos informação do que o necessário (problema de *expressividade* ou for mais difícil de entender (problema de efetividade) do que o dado bruto que é representado (o código fonte por exemplo)?

Uma analogia similar poderia ser feita na área de teste de software: “Como alguém descobre a ocorrência de um *bug* do tipo A em um programa do tipo B, usando um conjunto de teste C?”. Considerando duas visualizações diferentes para ajudar a resolver o problema, como determinar a mais *efetiva*? Ambas são adequadamente *expressivas*? Uma visualização pode ser mais vantajosa que a outra em facilitar essa tarefa de revelação de *bugs*? Como compará-las? Estes são problemas não-triviais que podem afetar o comportamento do usuário no uso das visualizações e deveriam ser considerados no planejamento e validação das propostas.

Algumas boas práticas são encorajadas pela literatura de visualização para orientar o processo de definição da visualização. Uma alternativa é a adoção de uma técnica de visualização que tenha sido projetada e avaliada para um problema similar. Outra alternativa é a definição de uma nova visualização baseada em um modelo de referência. Um modelo de referência pode ser entendido como um guia para mapear os dados brutos (programa ou código do caso de teste) em estruturas visuais e seus

mecanismos de interação. Além disso, um modelo de referência serve para discutir, comparar e contrastar visualizações (CARD; MACKINLAY; SHNEIDERMAN, 1999b).

Além dos aspectos gerais mencionados anteriormente, também é necessário prestar atenção às questões de suporte cognitivo relacionadas ao problema específico abordado pela visualização — no caso deste trabalho, as tarefas ou artefatos de teste unitário. No *framework* apresentado no Capítulo 5, três dimensões foram propostas para orientar a pesquisa de suporte cognitivo nas ferramentas de teste unitário: (i) a dimensão "Compreensão Testador Artefato" a qual aborda fatores que afetam os testadores nas tarefas de abstrair e entender os artefatos de teste unitário; (ii) a dimensão "Direcionamento e Orientação do Testador" a qual aborda fatores que influenciam a decisão dos usuários nas tarefas de teste unitário; (iii) "Usabilidade e Interação com a Ferramentas", a qual aborda aspectos de usabilidade na interface da ferramenta. Considerando essas preocupações, definimos uma estratégia para avançar na caracterização das necessidades e oportunidades de melhoria do suporte cognitivo para o teste unitário no domínio de visualização, conforme a seguir:

1. Dentre os trabalhos que aplicam visualização no teste de software, selecionar aqueles relacionados ao teste unitário.
2. Ler, analisar e inter-relacionar os aspectos dos trabalhos selecionados de acordo com as dimensões de suporte cognitivo definidas no Capítulo 5
3. Discutir as lacunas e oportunidades relacionadas ao planejamento e validação das visualizações, de uma perspectiva de suporte cognitivo.

Os detalhes dos passos 1 e 2 são apresentados nas subseções seguintes. O passo 3 é apresentado na Seção 6.3.

6.2.1 Seleção das Ferramentas que Aplicam Visualização

Os artigos selecionados para análise foram obtidos a partir do estudo de mapeamento sistemático descrito no capítulo 4. O objetivo deste mapeamento foi identificar trabalhos sobre técnicas e ferramentas de visualização aplicados à atividade de teste de software em geral.

Do total de 64 artigos obtidos na fase final do mapeamento, 11 são propostas de ferramentas no nível de teste unitário e 1 refere-se a um estudo experimental envolvendo ferramentas nesse nível (ver Tabela 6.1). Dos artigos relacionados às ferramentas, os trabalhos de Muto, Okano e Kusumoto (2011a) e Grechanik, Xie e Fu (2009) são evoluções de Muto, Okano e Kusumoto (2011b) e Conroy et al. (2007), respectivamente. Por simplicidade, nesses casos os artigos mais antigos foram considerados conjuntamente com as suas respectivas evoluções.

Tabela 6.1: Trabalhos que propõem ferramentas ou estudos experimentais usando visualização no teste unitário

Título	**Referências**
Assisting in fault localization using visual programming constructs	Karam e Abdallah (2005)
Restructuring unit tests with TestSurgeon	Estefo (2012)
Improvement of a Visualization Technique for the Passage Rate of Unit Testing and Static Checking and Its Evaluation	Muto, Okano e Kusumoto (2011a)
A Visualization Technique for the Passage Rates of Unit Testing and Static Checking with Caller-Callee Relationships	Muto, Okano e Kusumoto (2011b)
Creating GUI Testing Tools Using Accessibility Technologies	Grechanik, Xie e Fu (2009)
Automatic Test Generation From GUI Applications For Testing Web Services	Conroy et al. (2007)
How well do professional developers test with code coverage visualizations? An empirical study	Lawrence et al. (2005b)
Test Blueprints - Exposing Side Effects in Execution Traces to Support Writing Unit Tests	Lienhard et al. (2008)
MaVis: Feature-Based Defects Visualization in Software Testing	Wang, Zhang e Zhou (2012)
Representing Unit Test Data for Large Scale Software Development	Cottam, Hursey e Lumsdaine (2008)
Dynamic Fault Visualization Tool for Fault-based Testing and Prioritization	Daniel e Sim (2012)
ITVT: An image testing and visualization tool for image processing tasks	Romero et al. (2010)

6.2.2 Aplicação do *Framework*

A tarefa de projetar uma ferramenta que aborde requisitos de suporte cognitivo deve ser centrada no usuário. Como consequência, para avaliar o suporte cognitivo oferecido por uma ferramenta já desenvolvida, o foco da análise precisa estar nos caminhos de interação entre a ferramenta e o usuário, ao invés de somente na ferramenta. Durante essa análise, não tivemos a presunção de detectar todos os problemas de suporte cognitivo possivelmente existentes nas ferramentas analisadas. Na verdade, tal avaliação demandaria pelo menos um projeto de pesquisa dedicado a cada ferramenta, além da participação tanto dos projetistas das ferramentas quanto dos respectivos grupos de interesse. Tal avaliação está além do escopo deste trabalho. Ao invés disso, este trabalho conscientemente limita-se em mostrar as características de suporte cognitivo e problemas identificáveis nas ferramentas, baseando-se nas dimensões indicadas pelo *framework* descrito no capítulo 5.

O *framework* proposto no capítulo 5 emergiu de uma revisão de vários estudos na literatura envolvendo a participação de usuários da prática na atividade de teste unitário. Consequentemente, considera-se que a derivação das propostas de visualização do passo 1 nas três dimensões do *framework* é um passo importante em direção à caracterização de como o suporte cognitivo está sendo abordado nessas ferramentas. O resultado dessa derivação também serve para arranjar os aspectos de suporte cognitivo das ferramentas por similaridade e facilitar suas comparações. O seguinte processo foi adotado para aplicar esta derivação: para cada artigo considerado: (i) o artigo foi completamente lido, (ii) a visualização proposta foi entendida no contexto do problema abordado no artigo e (iii) as seguintes questões de pesquisa foram respondidas para cada dimensão correspondente do *framework*:

Dimensão Compreensão Testador Artefato: (RQ1) Como a visualização aborda o problema de reformular os artefatos de teste unitário para facilitar sua compreensão?

Dimensão Direcionamento e Orientação ao Testador: (RQ2) Como a visualização aborda o problema de orientar o testador nas tarefas de teste unitário?

Dimensão Usabilidade e Interação com a ferramenta: (RQ3) Como questões de usabilidade são abordadas nas ferramentas de visualização?

6.3 Resultados e Análise

Esta seção está organizada por tipo de visualização abordada. Para cada tipo, os resultados são discutidos para cada questão de pesquisa definida com base nas dimensões do *framework* (ver Seção 6.2.2).

6.3.1 Visualizações para o Teste Estrutural

Compreensão Testador Artefato

No trabalho de Muto, Okano e Kusumoto (2011a) a visualização reformula os artefatos de teste unitário representando a importância das unidades sob teste, a taxa com que os casos de teste passam e os resultados de análise estática usando uma combinação de dígrafo e gráfico de pizza. Os nós são classes e as arestas indicam os relacionamentos entre o método invocador e invocado. Cada nó é também um gráfico de pizza (veja Figura 6.1) que representa tanto os resultados do teste unitário ou da taxa de passagem da análise estática da classe correspondente. A espessura (peso) das arestas indicam a frequência com que o método de uma classe invoca os outros. O peso de um dado nó corresponde à soma dos pesos das arestas chegando neste nó. Quanto mais pesado o nó (mais importante), maior a posição que ele assume no *layout* do grafo.

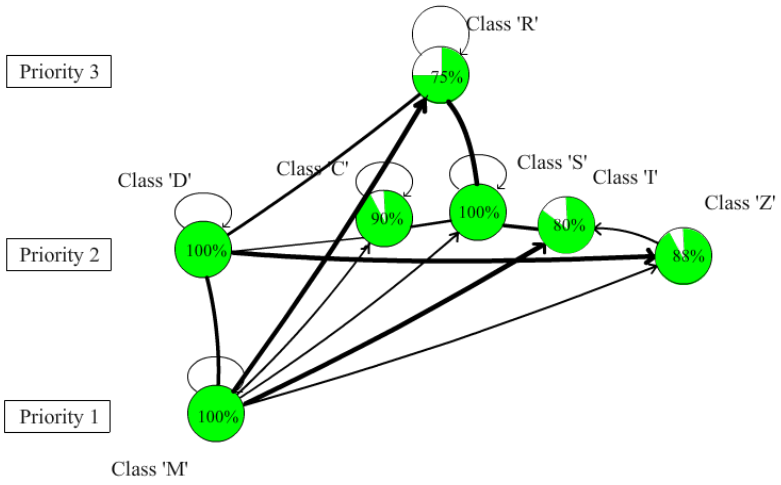

Figura 6.1: Exemplo do modelo de visualização de Muto et al. (MUTO; OKANO; KUSUMOTO, 2011a)((adaptado de Muto, Okano e Kusumoto (2011a))

A visualização proposta por Karam e Abdallah (2005) representa as funções a serem testadas (unidades) como grafo de fluxo de controle (GFC) e grafo de fluxo de dados (GFD). A cobertura dos atributos de programa (comandos, interações de fluxo de controle ou de dados) são mapeados visualmente por atributos coloridos. Por exemplo, os nós (comandos do programa) cobertos apenas por casos de testes que passam são coloridos em verde; os nós atravessados apenas por casos de teste falhos são cobertos em vermelho; os nós atravessados tanto por casos de teste que passam quanto por casos de teste que falham são coloridos em amarelo. Comandos, estruturas de controle, *loops* e associações definição-uso de variáveis são representadas por símbolos e customização sobre a representação tradicional de grafos.

A visualização proposta por Estefo (2012) (veja Figura 6.2) representa as diferenças entre pares de perfis de execução de casos de teste usando uma técnica de visualização chamada "visão polimétrica" (LANZA; DUCASSE, 2003). Cada visão (chamada de *blueprint*) gerada pela ferramenta mostra as diferenças entre um par de execuções de caso de teste para classes e métodos invocados pelos mesmos. No diagrama de *blueprint*, classes são caixas circundantes, métodos são caixas circundadas e arestas representam chamadas entre os métodos. As cores azul e vermelho foram atribuídas para cada caso de teste comparado. Caixas circundadas somente vermelhas ou somente azuis são somente executadas pelo caso de teste com a cor correspondente. Caixas amarelas são executadas por ambas. A altura e largura das caixas indicaram respectivamente, o número de vezes que os casos de teste azuis e vermelhos executaram o método.

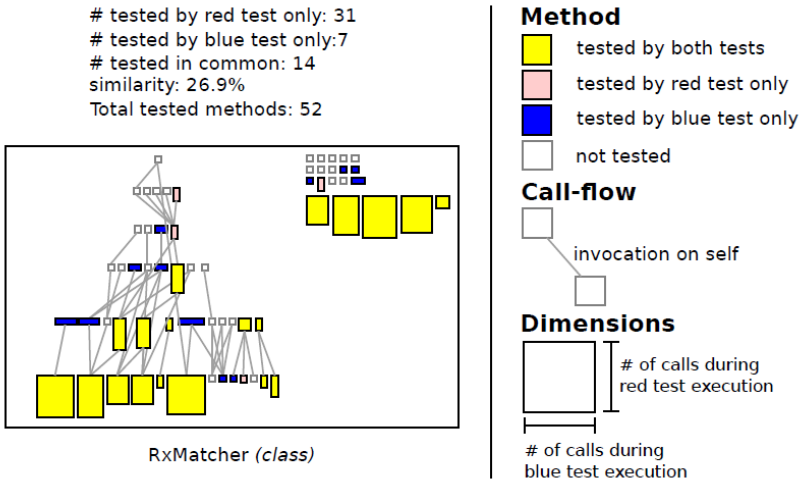

Figura 6.2: Blueprint de Estefo (2012) mostrando as diferenças entre um par de execuções de casos de teste (reusado com permissão de Estefo (2012))

Direcionamento e Orientação do Testador

No trabalho de Muto, Okano e Kusumoto (2011a) o algoritmo usado para traçar o dígrafo — bem como os comentários dos participantes relatados na seção experimento — indicam que a visualização ajuda a restringir o escopo das classes que precisam ser priorizadas na busca por *bugs*.

De acordo com Karam e Abdallah (2005) o colorimento do grafo proposto "permite que o usuário foque apenas nos atributos de programa e concentre-se mais claramente neles" isto é, eles tentam restringir o foco dos usuários a subconjuntos de interesse nos requisitos de teste considerados.

As cores e as formas das caixas representadas nos *blueprints* da visualização de Estefo (ESTEFO, 2012) servem como referência para os usuários no agrupamento e seleção de casos de teste que são potencialmente similares, considerando o impacto desses casos de teste em cada método. A identificação de casos de teste similares ajuda nas tarefas de refatoramento.

Interação com a Ferramenta e Usabilidade

Nenhum dos trabalhos proveram informação a respeito da usabilidade das ferramentas.

Análise

Embora a visualização de Muto, Okano e Kusumoto (2011a) mostre-se útil para revelar as discrepâncias entre o teste unitário e resultados de análise estática, foi identificado um risco potencial à expressividade da visualização do gráfico de pizza utilizado.

Em seu trabalho, a taxa de testes unitários que passam é calculada usando os resultados da cobertura de comandos da classe. Entretanto, a cobertura de comandos é reconhecida como um critério fácil de satisfazer, mesmo com testes fracos (INOZEMTSEVA; HOLMES, 2014). Considerando um caso em que o gráfico de pizza está 100% preenchido tanto para os resultados de teste unitário quanto para os de análise estática, este nó pode representar uma classe sem qualquer erro. Mas ela também pode representar uma classe com erros testada por casos de teste e anotações de análise estática de baixa qualidade que permanecem resultando em cobertura de 100%. Ambos correspondem a significados divergentes com o mesmo mapeamento visual. Similarmente, a visualização de Karam e Abdallah (2005) parece suscetível a problemas de expressividade. Os atributos de programa atravessados tanto pelos casos de teste que passam quanto pelos que falham são coloridos em amarelo. Entretanto, o artigo de Karam e Abdallah (2005) não menciona a diferenciação da coloração amarela para casos em que diferentes percentuais de casos de teste que passam e falha ocorrem sobre o mesmo atributo de programa exercitado. Isso implica que por exemplo, quando 10% do conjunto de teste passar e 90% falhar atravessando um certo atributo do programa, isso teria o mesmo mapeamento visual (para o atributo de programa considerado) do que quando 90% do conjunto de teste passar e 10% falhar. Uma variação na tonalidade da coloração amarela por exemplo, seria uma maneira possível de distinguir os casos exemplificados e adicionaria informações importante para o testador, por exemplo, rastrear o quão próximo ele está de mudar um atributo do programa de amarelo para verde ou vermelho.

A cor amarela nas caixas da visualização proposta por Estefo (2012) não apresenta tal problema de expressividade porque cada *blueprint* apenas corresponde a um par de casos de teste.

A intenção de Muto, Okano e Kusumoto (2011a) e Karam e Abdallah (2005) em restringir o foco do usuário para fatores importantes na estrutura do grafo pode ser melhor explicada por uma heurística de tomada de decisão denominada “Eliminação por aspectos” (TVERSKY, 1972). Esta estratégia pode ser aplicada quando uma pessoa tem que tomar decisões considerando um grande número de escolhas. Ao invés de manipular mentalmente e ponderar todos os aspectos de cada alternativa, ele foca em um atributo por vez como um critério mínimo. Isto habilita a pessoa a excluir iterativamente as opções que não satisfazem este critério e priorizar escolhas. Alternativamente, a visualização de Estefo Estefo (2012) usa as cores das caixas e a similaridade de suas formas para dar suporte aos testadores no agrupamento e reconhecimento de métodos que são similarmente afetados pelos pares de casos de teste. Tal exploração do reconhecimento das formas visuais e do agrupamento por similaridade pode ser melhor explicada pelas variáveis retinais de Bertin (2010) e princípios de Ellis (1999). Entretanto, os três trabalhos apresentam informação limitada sobre o projeto e avaliação das visualizações

com relação ao auxilio à cognição dos usuários. Para garantir um suporte cognitivo adequado, seria importante, por exemplo: realizar um estudo com o usuário *a priori* (adotando métodos como entrevista, *surveys*, análise de grupo focal) para entender como os usuários lidavam anteriormente com os problemas nas suas tarefas diárias de teste unitário; qual saída destes estudos com o usuário motivaram a definição ou customização das características da visualização de uma forma particular; caso a tomada de decisão e as estratégias de reconhecimento mencionadas forem intencionalmente reforçadas pelas visualizações, quais aspectos observados nos estudos com os usuários motivaram a sua implementação na forma proposta; para determinar a extensão à qual suas visualizações adequadamente beneficiam os usuários e quais são os possíveis riscos com relação à interpretação.

Lawrence et al. (2005b) — o único estudo selecionado completamente dedicado à experimentação — encontrou alguns resultados importantes com relação à influência da visualização no teste estrutural sobre o desempenho e comportamento dos usuários. Um dos resultados do experimento por exemplo, não mostrou diferença significativa no número de falhas encontradas entre o grupo tratamento (usando a visualização da cobertura de blocos) e o grupo controle (não usando a visualização para o mesmo critério). Um segundo resultado demonstrou que o número de casos de teste gerados não foi afetado pela adoção da visualização, embora a variabilidade na quantidade de casos de teste escritos tenha sido reduzida no grupo tratamento. Por fim, outro resultado sugere que a visualização pode afetar o comportamento do usuário ao encorajá-los a superestimar a eficácia *footnote*Usado da mesma forma que no trabalho original isto é, para indicar a capacidade de encontrar erros. dos seus testes. De acordo com os autores, isto se deve possivelmente ao *feedback* visual positivo obtido quando a adequação do critério era alcançada. Considerando os dois últimos resultados, a influência da visualização para o teste estrutural sobre os usuários são divergentes: ela pode orientar o comportamento dos testadores de forma negativa por exemplo, induzindo-os a parar os testes precocemente por conta do efeito da superestimação. Entretanto, a visualização pode ser útil para propósitos de padronização uma vez que ela reduz a variabilidade no número de casos de teste gerados.

6.3.2 Visualização para Teste de Sistemas Legados

Compreensão Testador Artefato

A visualização proposta por Conroy et al. (2007) ajuda no teste de serviços web gerados a partir de sistemas legados com *interface* gráfica, os quais são abreviados no artigo dos autores como "*GAPs*". A visualização proposta por Conroy et al. (2007) provê uma forma de ligar visualmente os elementos da interface gráfica do sistema legado à

interface do *web service* correspondente, servindo como uma conexão para transferir valores de entrada/saída entre os mesmos (veja Figura 6.3). O usuário define os elementos da *interface* gráfica de interesse usando ações *drag-and-drop* da tela do *GAP* para a ferramenta de visualização. Depois disso, o usuário pode ligar o *GAP* e o *Web Service* desenhando setas entre os *getters/setters*, que representam os elementos na interface gráfica do *GAP* na ferramenta, e a *interface* do *webservice* alvo. Este mecanismo ajuda na captura de dados úteis para a definição do conjunto de teste inicial.

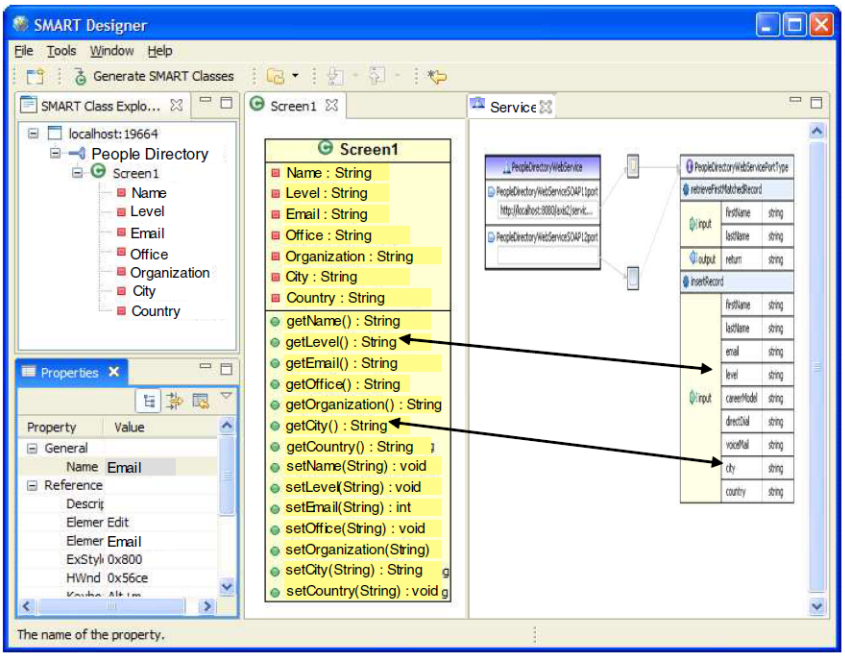

Figura 6.3: Visualização de Conroy et al. (2007) (reusado com permissão de Conroy et al. (CONROY et al., 2007))

A visualização de Lienhard et al. (2008) ajuda os usuários na identificação de partes relevantes do código legado para o propósito de teste (unidades de execução). A visualização é composta de duas partes principais: (i) o *trace* de execução e (ii) o *blueprint* do teste (veja Figura 6.4). O trace de execução usa uma visão bi-dimensional para representar as execuções dos métodos ao longo do tempo para um exercício do programa. A visualização adota cores preta e cinza para mostrar se os métodos (representados por retângulos verticais) já foram cobertos por um teste, ou não. A posição do retângulo indica sua ocorrência no tempo e o tamanho vertical indica o tempo gasto executando. A parte do *blueprint* do teste usa uma visualização de grafo — similar ao diagrama de objetos *UML* — para representar os métodos selecionados como uma unidade de execução candidata pelo usuário. Ele é usado para representar objetos que já existiram antes do início da unidade de execução (nós com rótulos em fonte regular); objetos instanciados pela unidade de execução (nós com rótulos em fonte negrito); referências entre objetos que foram acessados enquanto a unidade de execução rodava (uma seta preta entre os

objetos); referências entre o objeto que já existia antes da unidade de execução rodar (uma seta cinza entre os objetos) e os efeitos colaterais produzidos (uma janela *pop-up* mostrada quando o cursor é movido sobre os objetos).

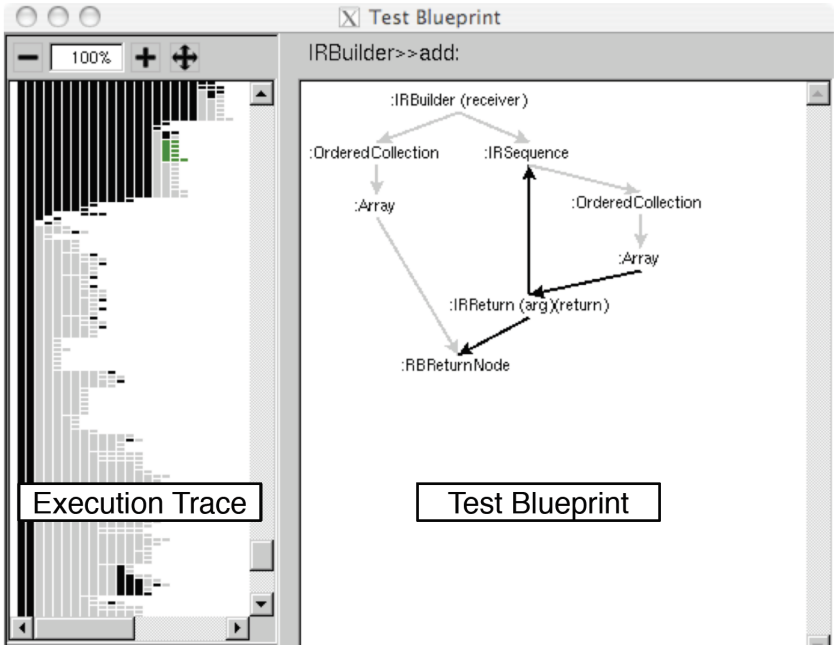

Figura 6.4: Visualização de (LIENHARD et al., 2008) mostrando o trace de execução e a parte o blueprint do teste (reusado com permissão de Lienhard et al. (2008))

Direcionamento e Orientação do Testador

No trabalho de Conroy et al. (2007) a ligação visual da interface do sistema legado ao *web service* correspondente é um mecanismo para habilitar o reuso do conhecimento do usuário sobre o sistema legado na derivação de importantes valores de entradas/saída. Estes valores capturados pela ferramenta e convertidos em unidades de caso de teste orientam o testador no estabelecimento da *baseline* inicial de casos de teste.

A visualização da parte do trace de execução no trabalho Lienhard et al. (2008) tenta facilitar a tarefa de eleger uma unidade de execução ao selecionar algumas referências para este propósito — por exemplo, as cores cinza e preta dos retângulos indicando sua cobertura e a ocorrência no tempo, indicada pela posição do retângulo. Ela também ajuda o usuário a restringir seu escopo de problemas ao prover mecanismos de interação tais como a filtração da quantidade de informação na visualização do trace de execução (por exemplo, mostrar somente métodos de uma classe ou pacote em particular). A visualização da parte do *blueprint* de teste provê referências para o desenvolvedor nas tarefas de: (i) implementação do *fixture* do caso de teste; (ii) execução da unidade sob teste e (iii) escrita das assertivas. Para a implementação da *fixture* por exemplo, os nós com rótulos que não estão em negrito são objetos que precisam ser criados.

Usabilidade e Interação com a Ferramenta

Nenhum dos trabalhos proveu informação com relação à usabilidade da ferramenta.

Análise

Conroy et al. (2007) afirma que eles avaliaram sua proposta por meios de um estudo de viabilidade. Eles dizem que suas experiências confirmam os benefícios da ferramenta mas eles não dão informações relacionadas à participação de voluntários. Desta forma, não pudemos obter detalhes relacionados à *efetividade* ou *expressividade* da visualização com usuários. Em contraste, o trabalho de Lienhard et al. (2008) tem um número limitado de participantes mas tenta responder questões importantes relacionadas à efetividade da visualização (por exemplo, "Quão direto é o uso (de uma ferramenta de visualização?") e sua expressividade (por exemplo, "O *blueprint* do teste comunica concisamente a informação requerida?").

Seria pertinente investigar a extensão pela qual a visualização de Conroy et al. (2007) dá suporte ao testador na aplicação da estratégia de "Satisfabilidade" (SIMON, 1956). Esta estratégia é geralmente aplicada quando os recursos são limitados e a pessoa precisa encontrar o *trade-off* entre a seleção das escolhas que satisfazem os requerimentos — neste caso, os casos de teste unitário relacionados às operações usadas frequentemente e obtidas pela ferramenta — e a minimização de possíveis perdas por exemplo, tempo, pessoas ou outros recursos necessários para entender o código legado. Lienhard et al. (2008) também não avaliaram as estratégias de tomada de decisão na sua avaliação. Entretanto, eles mostraram preocupação com os resultados afetados por tais decisões. Por exemplo, em seu segundo estudo de caso eles tentam determinar "Como os testes escritos usando a sua abordagem diferenciam-se do teste convencional escrito por um *expert*". Um de seus resultados demonstrou que algumas assertivas importantes foram descobertas por causa da visualização e que elas haviam sido ignoradas previamente.

6.3.3 Visualização Ligando as Unidades de Teste ao Escopo do Sistema

Compreensão Testador Artefato

A visualização proposta por Wang, Zhang e Zhou (2012) representa um "modelo de teste baseado em características" isto é, um modelo que representa o sistema sob teste, seus casos de teste e defeitos. A visualização na ferramenta consiste de duas partes: "árvore da estrutura de sistema" e "gráfico de resultados". A parte da árvore de estrutura do sistema representa o modelo de teste baseado em características. Um nó

folha representa uma característica, a qual corresponde a um elemento atômico do sistema para propósito de teste (uma unidade). Nós internos representam módulos do sistema — um conjunto de características ou recursivamente, um conjunto de módulos. Os nós são coloridos de vermelho e verde e expressam a taxa de defeitos dividido pelos casos de teste de cada característica — coloração mais verde para taxas baixas e mais vermelha para taxas altas. A parte do gráfico de resultado é um histograma usado para representar a "complexidade de cobertura", uma métrica usada pelos autores para representar o número de casos de teste e características para cada tipo de complexidade de característica. Consequentemente, usuários precisam pré-classificar as características em um dos três tipos de complexidades definidas, ou seja, alta, média e baixa, baseado na complexidade da implementação e na lógica de negócio.

A visualização proposta por Cottam, Hursey e Lumsdaine (2008) representa os resultados do teste unitário da ferramenta de teste de *MPI "MTT"* de forma agregada, para habilitar uma interpretação de alto nível pela comunidade *MPI*. A visualização comprime os dados da suíte de teste de um espaço N-dimensional para uma grade de representação bidimensional. O eixo "x" representa a arquitetura da suíte de teste e informações do sistema operacional e o eixo "y" representa o número de bits e a família do compilador. A cor de fundo de cada célula da grade é usada para mapear o estado da suíte de teste "trivial", ou seja, uma suíte de teste cuja falha implica na falha dos outros testes. Símbolos no primeiro plano da célula da grade mapeiam visualmente os resultados de cada suíte de teste não trivial. Estes símbolos são também gráficos de pizza, usados para indicar a taxa com que as suítes de teste não triviais passam nos testes.

Direcionamento e Orientação do Testador

A visualização de Wang, Zhang e Zhou (2012) orienta os usuários na busca por *bugs* ao prover simultaneamente uma visão global e refinada do sistema sob teste. Mecanismos de interação são providos para expandir e retrair nós, habilitando o usuário a abrir ou restringir seu foco nos nós considerados. As cores dos nós indicam a seriedade dos problemas nas características representadas pelos nós. O histograma indica se os casos de teste correntes são suficientes para cada tipo de complexidade de característica, ajudando os usuários a decidirem se devem adicionar mais casos de teste para cada tipo de complexidade ou não.

A visualização de Cottam, Hursey e Lumsdaine (2008) foi validada usando somente uma combinação específica de dados de teste disponíveis do *MTT*. Entretanto, os autores indicam que as visualizações podem ser adaptadas para representar uma variedade de combinações de dados. Considerando a variedade de combinações de dados que podem ser representadas, a visualização propostas por Cottam, Hursey e Lumsdaine (2008)

habilita os usuários a monitorarem os resultados do teste unitário e de cobertura do projeto *MPI* de forma continua, acurada e compacta.

Usabilidade e Interação com a Ferramenta

Nenhum dos trabalhos provê informação relacionada à usabilidade da ferramenta.

Analise

O trabalho de Wang, Zhang e Zhou (2012) não mostra informação sobre o envolvimento dos usuários na fase de planejamento da visualização. Além disso, o seu trabalho não reporta qualquer tipo de validação.

O trabalho de Cottam, Hursey e Lumsdaine (2008) é uma demonstração abrangente de práticas para o suporte adequado da cognição do usuário por meio da visualização. Primeiro, o público alvo foi considerado desde os passo iniciais do design. Isso envolveu: entender como a informação era previamente obtida dos dados crus; classificar os usuários em grupos; entender os tipos de informações necessárias para cada grupo e priorizá-las. A opinião dos usuários foi decisiva para várias escolhas feitas ao longo do processo de mapear visualmente os dados, tais como: a metáfora visual aplicada; cores e tonalidade de cores adotadas; a codificação dupla dos dados para resolver casos particulares onde a compreensão da informação poderia ser afetada (veja Figura 6.5). Os autores também avaliaram a visualização com relação à sua expressividade e efetividade para os usuários alvo. Essa avaliação incluía questões para: (i) comparar a interpretação das informações pelos usuários usando a visualização *versus* os modos anteriores à visualização; (ii) descobrir *insights* dos usuários facilitados pelas visualizações; (iii) determinar a capacidade dos usuários em alcançar seus objetivos usando a visualização; (iv) checar as melhorias na representação de dados e assistência ao usuário. Considerando a variedade de informações que as visualizações podem prover e os objetivos aos quais elas podem servir, a avaliação de uma ou mais estratégias de tomada de decisão poderia se tornar uma tarefa dispendiosa e inviável. Ao invés disso, os autores pediram aos usuários uma lista de *insights* providos pela ferramenta de visualização, os quais não podiam ser notados anteriormente. Os autores também encorajaram o criticismo dos usuários sobre suas visualizações.

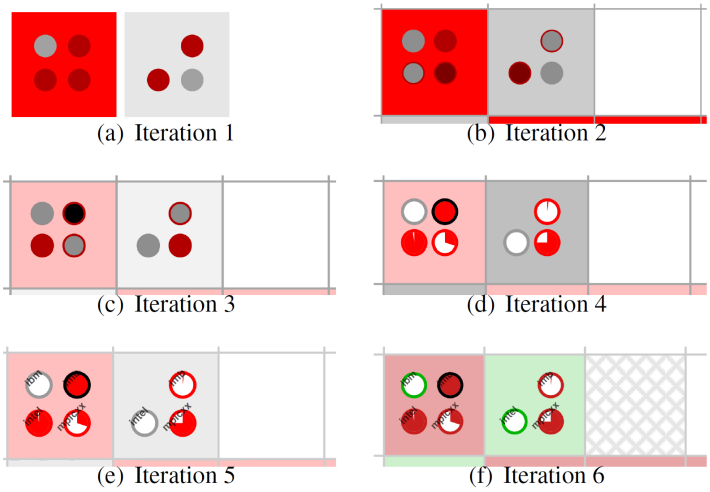

Figura 6.5: As várias iterações mostrando a evolução da codificação da suíte de teste em uma visualização do tipo grade de células (COTTAM; HURSEY; LUMSDAINE, 2008) (reusado com permissão de Cottam et. al (COTTAM; HURSEY; LUMSDAINE, 2008))

6.3.4 Visualizações Miscelâneas

Compreensão Testador Artefato

A visualização proposta por Daniel e Sim (2012) facilita a tarefa de selecionar entradas de teste que matam mutantes de expressões de álgebra booleana (as unidades sob teste) ao priorizar as entradas de teste de acordo com a sua aptidão para matar uma maior quantidade de mutantes por vez. Desta forma, uma entrada de teste A que mata mais mutantes do que uma entrada de teste B é *rankeada* primeiro. As entradas de teste são representadas por uma célula colorida em uma matriz. Cada célula é colorida usando uma cor em escala de cinza 8-bits. Caixas mais escuras representam entradas de teste que podem matar mais mutantes por vez. O usuário pode executar as entradas de teste clicando nas caixas as quais se tornariam vermelhas depois disso. As caixas são automaticamente atualizadas depois da execução de cada entrada de teste. A visualização também provê uma visão alternativa que indica os erros que podem ser detectados pela entrada de teste selecionada.

O trabalho de Romero et al. (2010) propõe uma ferramenta para facilitar a execução e análise do teste de unidade dos programas que usam imagem como dado de entrada. Sua visualização reformula os artefatos de unidade de teste, tornando visível e rastreável as imagens (dados de entrada) e as modificações que são feitas sobre as mesmas pelo programa sob teste, em uma taxa de atualização periódica. Isso permite que o usuário monitore os resultados da execução dos testes visualmente, ao invés de confiar apenas nas assertivas dos resultados.

Direcionamento e Orientação do Testador

Na visualização de Daniel e Sim (2012) a representação de células coloridas e a atualização dinâmica serve como guia para o usuário não-*expert*, de forma que ele possa progressivamente selecionar um pequeno subconjunto de casos de teste que mata todos (ou quase todos) os mutantes. A informação provida na janela de visão alternativa serve como uma referência para ajudar o usuário na identificação de erros, caso esses erros sejam detectados pelo entrada de teste no futuro.

De acordo com Romero et al. (2010), sua ferramenta de visualização ajuda o usuário a encontrar e entender erros produzidos por programas de manipulação de imagens enquanto casos de teste unitários são executados sobre os mesmos. Mecanismos tais como a capacidade de mostrar desenhos gerados pelo programa enquanto as imagens são modificadas (caso o programa seja instrumentado para tal), são habilitados por um *canvas*. Além disso a ferramenta de visualização habilita mecanismos de interação tais como zoom para melhor análise das modificações de imagem.

Usabilidade e Interação com a Ferramenta

Nenhum dos trabalhos provê informação relacionada à usabilidade da ferramenta.

Análise

Nenhum dos trabalhos menciona o envolvimento do público alvo no processo de planejamento da ferramenta de visualização. Não há relatórios sobre estudos anteriores com potenciais usuários tais como entrevistas, *surveys*, ou questionários para elicitar requisitos para as visualizações propostas. O contexto no qual essas propostas são aplicadas é muito específico. Como consequência, o suporte cognitivo provido por tais visualizações deveria ser customizado para atender as necessidades e habilidades específicas de seus usuários.

6.4 Considerações finais

Este capítulo cobriu vários aspectos dos trabalhos de visualização de teste unitário que podem influenciar o suporte cognitivo fornecido ao usuário.

Em geral, nota-se a falta de participação dos usuários no projeto e avaliação das visualizações. Somente um trabalho em particular mostrou a importância do envolvimento compreensivo do usuário para o estabelecimento de um suporte cognitivo adequado.

Também foram identificadas estratégias de tomada de decisão subjacentes e princípios de reconhecimento visual que poderiam ser melhor explorados e avaliados com o usuário.

Nenhum dos trabalhos analisados contém informação sobre a usabilidade das ferramentas de visualização. Atribui-se a isso dois possíveis fatores: (i) algumas ferramentas encontravam-se em estágio de protótipo no momento da publicação dos trabalhos e (ii) a participação escassa de usuários nas validações das propostas, quando existente. Todavia, ressalva-se que a usabilidade está diretamente relacionada à experiência do usuário com a ferramenta. Logo, a percepção do usuário sobre como a visualização funciona em suas mãos pode influenciar sua decisão em adotá-la ou não para suas necessidades práticas.

Com relação às visualizações para o teste unitário estrutural, foram identificados riscos relacionados à expressividade que poderiam afetar a interpretação dos usuários. Além disso, um dos trabalhos envolvendo experimentação mostrou a influência que a visualização pode ter sobre o comportamento dos usuários na atividade de teste estrutural.

As lacunas e oportunidades identificadas neste capítulo mostram que a tarefa de construir visualizações confiáveis para o teste unitário requer preocupações que estão fortemente focadas na participação e comportamento do usuário. Essa tarefa não pode ser restringida a questões como o arcabouço teórico de teste já existente, mecânica das ferramentas, resultados de saída ou aspectos estéticos por exemplo.

Por fim, novos estudos podem somar-se ao conhecimento resumido neste capítulo. Por exemplo, mais pesquisa como o trabalho de Lawrence et al. (2005b) poderia ajudar a melhorar o entendimento das barreiras estabelecidas pelas visualizações de teste unitário sobre o comportamento dos usuários para diferentes técnicas de teste. Percebe-se também uma demanda urgente pela condução de mais estudos envolvendo o público alvo na definição das visualizações. Isso se aplica aos estágios iniciais e intermediários — quando questões práticas como a expressividade podem ser avaliadas — e aos passos finais, quando o impacto das visualizações no comportamento dos usuário precisa ser melhor compreendido.

Finalmente, é importante destacar que as análises e críticas feitas neste capítulo não representam uma crítica ao mérito de qualquer um dos trabalhos analisados. De fato, todos eles são fundamentados na teoria de teste de software existente e representam avanços e contribuições relevantes na área. Mas é importante enfatizar que ao lidar com suporte cognitivo, os aspectos humanos precisam ser transversais no processo de pesquisa. Como consequência, os potenciais usuários das ferramentas precisam estar na linha de frente, acompanhando e sendo acompanhados pelos pesquisadores em decisões que afetam a prática da(s) atividade(s) suportada(s) pela ferramenta. Não um passo atrás, ou postegados a passos pós-construção das ferramentas. Por outro lado, nosso trabalho mostra o grande potencial existente nessas ferramentas e como seus autores —

e os pesquisadores de futuras ferramentas de teste unitário — poderiam se beneficiar dessas observações e colocar suas ferramentas um passo mais próximo da transferência tecnológica.

CAPÍTULO 7

Um Framework de Suporte Cognitivo para a Revisão de Testes Unitários

7.1 Considerações Iniciais

Atualmente, não há teoria sobre como os profissionais realizam a atividade de teste unitário. Além disso, o suporte que as ferramentas atuais provêm é geralmente limitado à eliminação do esforço manual repetitivo, por exemplo: geração de dados de teste, verificação de assertivas, checagem de cobertura etc. Conforme observado no Capítulo 5, as necessidades dos profissionais de teste unitário vão além de questões relacionadas à substituição de esforço manual repetitivo. Adicionalmente, no Capítulo 6, são apresentadas evidências de que embora a comunidade de pesquisa tenha proposto ferramentas com o intuito de facilitar a atividade de teste unitário, as soluções propostas possuem deficiências que não permitem o provimento de suporte cognitivo apropriado aos usuários. Considerando isso, dois atributos tornaram-se claros durante a evolução desta pesquisa: (i) o fenômeno de interesse ainda não era claramente compreendido na área; (ii) sua pesquisa requeria uma abordagem de investigação que pudesse explorar adequadamente a experiência de seres humanos, pela perspectiva daqueles que vivem esse problema. Considerando isso, uma abordagem qualitativa foi adotada nesta pesquisa. De acordo com Creswell (2013a) "na pesquisa qualitativa, o interesse é explorar o conjunto de fatores complexos em torno do fenômeno central e apresentar as variadas perspectivas ou significados que os participantes detêm". Neste trabalho, o "fenômeno central" corresponde à prática de revisão de testes unitários. O "conjunto de fatores complexos" são as tarefas relacionadas aos problemas recorrentes dessa prática. Por fim, "a perspectiva variada que os participantes detêm" são as peças de informação que permitem a definição de um *framework* de suporte cognitivo empregando uma abordagem centrada no usuário.

Neste capítulo são apresentados os detalhes do estudo qualitativo realizado para a definição deste novo *framework*. O *framework* representa um modelo teórico, o qual

deverá informar novos estudos e o projeto de ferramentas que melhorem o suporte cognitivo provido aos testadores na prática de revisão dos testes unitários.

7.2 Análise de Conteúdo Qualitativo

A escolha por uma abordagem de pesquisa qualitativa foi motivada pela necessidade de explorar o tópico de pesquisa e entender o fenômeno de interesse a partir da perspectiva dos profissionais de teste. Muito pouco é escrito ou conhecido sobre as experiências desses praticantes nos testes unitários e seus desafios.

As abordagens qualitativas não envolvem a formulação e o teste de hipóteses que são comumente vistos em abordagens quantitativas, isto é, predições sobre relações esperadas entre variáveis. Em vez disso, os pesquisadores utilizam questões de pesquisa amplas e gerais, de modo a não limitar a investigação (CRESWELL, 2013b). Para atingir esse objetivo, os pesquisadores utilizam a literatura e os estudos existentes para: (i) incorporar resultados de outros estudos que são tangenciais à investigação; (ii) comparar afirmações de observações dos participantes sobre o tema; (iii) descobrir problemas que não foram estudados; e (iv) compreender controvérsias metodológicas e teóricas no campo de estudo (FLICK, 2014; CRESWELL, 2013c; CORBIN; STRAUSS, 2014). Alinhada a esses princípios, a revisão da literatura realizada nos Capítulos 4 e 5 situa o estudo descrito no presente capítulo dentro de um contexto mais amplo mas sem impor limites à interpretação das perspectivas dos participantes (CRESWELL, 2013c; CORBIN; STRAUSS, 2014).

A pesquisa qualitativa é apropriada quando um problema de pesquisa precisa ser explorado, de forma a entender um grupo de pessoas, identificar variáveis que não são conhecidas (ou facilmente mensuráveis) e para trazer à luz da investigação, "vozes" que encontram-se silenciadas. Ela é apropriada para contextos nos quais a teoria é escassa ou inexistente, e um entendimento complexo e detalhado do fenômeno é necessário. Os pesquisadores tentam dar sentido a esses fenômenos a partir da perspectiva do participante. As descobertas de um estudo qualitativo são ricamente descritivas. Elas são representadas geralmente na forma de categorias, temas, tipologias, hipóteses tentativas ou teorias substanciais ao invés de derivação de postulados ou rejeição/aceitação de hipóteses (tal como na pesquisa positivista) (CRESWELL, 2013a) (CRESWELL, 2007) (MERRIAM, 2002) (MERRIAM, 2002). Em contraste à pesquisa quantitativa, a pesquisa qualitativa raramente define uma hipótese explicita antecipadamente. Na verdade, as hipóteses são geradas e amadurecidas ao longo do tempo — com a coleta dos dados e o processo de análise — ou elas emergem como parte dos resultados do estudo. (JR; UNRAU, 2010) (HENDERSON; CARTER, 2009).

Neste trabalho o método "*Qualitative Content Analysis*" (QCA) é adotado para analisar as respostas dos praticantes. QCA é um método estabelecido, usado para examinar e extrair significado de uma coleção de dados textuais. Ele adere ao paradigma naturalista por desenvolver e revisar o entendimento de um fenômeno iterativamente — em contraste à abordagem positivista de checar conclusões previas de teorias (BROD; TESLER; CHRISTENSEN, 2009). Suas características fundamentais incluem a codificação sistemática dos dados, bem como a interpretação e derivação de categorias ou temas, para elucidar um fenômeno de interesse em um contexto social. Os dados textuais podem vir de diferentes fontes tais como dados narrativos, *surveys* de questões abertas, entrevistas, grupos focais e outros tipos de mídias textuais. Embora possa-se apontar similaridades entre o QCA e *grounded theory* (GT), existem diferenças entre os mesmos, tanto para o processo adotado como para os resultados gerados. A distinção entre ambos é importante do ponto de vista metodológico e está em consonância com a discussão levantada por Stol, Ralph e Fitzgerald (2016) com relação ao mal uso de *grounded theory* — nomeado pelos autores como "*grounded theory à la carte*".

Grounded theory, ou "teoria fundamentada", é uma metodologia completa com uma forte fundamentação filosófica. Na prática, ela usualmente requer um processo bem mais lento, opera indutivamente, supõe fontes de dados variadas e pode ser aplicada sob pelo menos três paradigmas distintos, cada um com suas nuances (ADOLPH; HALL; KRUCHTEN, 2011). Conforme indicado, o seu objetivo final é construir uma teoria que seja bastante enraizada no processo adotado desde os estágios iniciais, em particular durante o desenho do estudo e a coleta de dados. Algumas das características distintas são a habilidade de observar, analisar e ajustar as questões enquanto as respostas são obtidas (amostragem teórica) (GLASER; STRAUSS, 1999).

O trabalho de Stol, Ralph e Fitzgerald (2016), além de bastante informativo quanto à aplicação de *grounded theory*, discute o uso inadequado desta denominação por muitas publicações na área de engenharia de software. Eles explicam a alta complexidade envolvida na execução adequada de estudos do tipo *grounded theory* e as nuâncias envolvidas em suas três principais variantes. Ao longo do artigo, eles mostram como algumas publicações não têm atendido aos critérios para serem designadas como estudos desse tipo. Além disso, provêm dicas úteis sobre o que deve ser considerado nas diferentes variantes da metodologia e na apresentação dos resultados de pesquisa.

Em contraste, QCA é um método para interpretar uma coleção de dados textuais. Este método é reconhecido por suas propriedades de reduzir os dados, ser sistemático e flexível (FLICK, 2013). QCA segue um processo bem mais direto em comparação com *grounded theory*. O método QCA aceita tanto a codificação indutiva ("convencional") quanto a dedutiva ("direcionada") para o processo de desenvolvimento de categorias. Permite também, a interpretação do conteúdo latente ou manifesto dos dados textuais

isto é, codificação do significado implícito ou explícito dos dados textuais (HSIEH; SHANNON, 2005). O método não tem por objetivo alcançar o desenvolvimento total de uma teoria — como pretendido pelo GT — mas revelar categorias de dados que possam ajudar a responder as questões de pesquisa propostas.

Apesar do fato de QCA ser um método qualitativo, ele foi parcialmente inspirado no método "*Quantitative Content Analysis*" e preserva alguns benefícios do método, tal como: (i) determinar quais aspectos da comunicação devem ser considerados durante o processo de análise — por exemplo, experiências de comunicação, contexto onde o texto foi produzido, *background* pessoal dos participantes; (ii) adoção de um processo pré-estabelecido para analisar o material e fragmentá-lo em unidades de análise; (iii) condução de uma análise cuidadosamente orientada a categorias por meio de sucessivas iterações; (iv) adoção de um critério de confiabilidade para tornar os resultados confiáveis e comparáveis (MAYRING, 2014).

7.3 Contextualização dos Participantes e o Mecanismo de Coleta de Dados

Os diferentes níveis de teste demandam diferentes tipos de análise dos profissionais de teste. Por exemplo, no teste unitário, os praticantes precisam focar sua atenção nas unidades do software individualmente e em seus conteúdos internos, enquanto que no teste de integração eles devem prestar atenção às interfaces desses elementos e como elas orquestram sua comunicação. A responsabilidade de desenvolver o teste unitário é comumente atribuída aos desenvolvedores (programadores). A premissa de que os desenvolvedores têm um conhecimento detalhado das unidades sob teste motiva essa escolha. Isto porque a compreensão das unidades contribui para abreviar o ciclo teste, reportação dos *bugs*, depuração, reparo, e re-teste. Desta forma, este estudo admite tanto profissionais exclusivamente dedicados às tarefas de garantia da qualidade, quanto desenvolvedores que atuem nesse tipo de atividade. O único requisito é que os voluntários tenham experiência prévia com o teste unitário.

Uma das preocupações iniciais no estudo foi maximizar o número de participantes e respostas válidas obtidas. Esta estratégia permite a análise de um conjunto de dados robusto e capaz de revelar informações primordiais ao estudo. Como consequência, foi preciso estabelecer um balanceamento entre a liberdade dada aos voluntários para ingressarem na pesquisa e a extensão de informações que se poderia capturar e interpretar dos mesmos. Um *survey* online que pudesse ser livremente aceito e respondido — onde e quando os voluntários quisessem — mostrou-se adequado para a participação voluntária almejada. Entretanto, isso também tem implicações para o nível de interpretação adotado

no processo de codificação, uma vez que não seria possível capturar outros aspectos tais como as linguagens corporais dos participantes ou as entonações em suas falas. Logo, o principal aspecto de comunicação considerado durante a análise foi a informação textual provida nas respostas dos participantes. Além disso, questões fechadas foram usadas para obter informações sobre o perfil dos participantes e para descobrir suas perspectivas sobre os problemas abordados.

7.3.1 Elaboração do *Survey*

A criação e aplicação de um *survey* é um processo laborioso. Os pesquisadores estão frequentemente envolvidos na investigação por um longo período, e inevitavelmente desenvolvem considerações particulares sobre o assunto. Por outro lado, eles precisam desenvolver um questionário que seja fácil de ser respondido por participantes que não têm o mesmo *background*. Nesta subseção, é partilhada parte da experiência acumulada nessa importante parte da pesquisa. Estas informações fornecem uma visão clara dos esforços aplicados na elaboração do *survey*. Além disso, as informações podem ser úteis a pesquisadores interessados em antecipar algumas das questões comumente enfrentadas em estudos similares.

A primeira — e possivelmente mais difícil — questão foi o desejo de acelerar o processo de ajuste fino das questões do *survey*. Voluntários qualificados são, em geral, difíceis de encontrar, preciosos para se perder e nós não queremos perder a oportunidade de fazer todas — e somente — as perguntas corretas. Como consequência, foi preciso ajustar as questões do *survey* de forma que elas fossem suficientemente abertas para permitir que os participantes se sentissem livres para discorrerem sobre suas respostas. Por outro lado, as questões deviam ser restritas o suficiente para ajudá-los a não divagarem. Logo, a preocupação inicial foi definir esse balanceamento antecipadamente para cada questão proposta. Entretanto, depois de rodar o primeiro piloto do *survey*, observamos que tais preocupações estavam estreitando nossa criatividade para gerar boas perguntas. Depois de alguma auto-crítica, percebemos que precisávamos nos permitir maior criatividade. Desta forma, decidimos que a prioridade naquele momento deveria estar somente em criar questões relevantes isto é, questões que, de fato, abordassem o problema de interesse. O número de questões, seus tipos (abertas ou fechadas), a precisão do vocabulário adotado ou qualquer outra preocupação estrutural não deveria ser relevante naquele momento. Ao invés disso, as questões deveriam parecer interessantes e prender a atenção dos voluntários. Esta abordagem habilitou a elaboração de novas questões e o exercício de novas rodadas de teste do *survey*. Felizmente, as resposta obtidas após as modificações mostraram-se promissoras: enquanto certamente seria preciso remover e refinar algumas questões, pela primeira vez os participantes estavam claramente entendendo as perguntas

feitas e fornecendo respostas ricas, não ambíguas e que fossem relevantes aos interesses do estudo.

A segunda preocupação foi determinar o papel das perguntas fechadas na pesquisa. Perguntas fechadas são rápidas de responder, os pesquisadores podem pensar antecipadamente sobre as alternativas de respostas, e os participantes têm apenas que escolher qual delas encaixa-se melhor em suas opiniões. Entretanto, tal facilidade esconde um problema: este tipo de questão frequentemente diz mais sobre a quantificação de um fenômeno do que sobre suas características. O objetivo final deste estudo não era entender como as opiniões dos participantes estavam distribuídas nas várias facetas de um fenômeno previamente entendido. Ao invés disso, desejava-se compreender um fenômeno inexplorado, baseado na perspectiva dos participantes. Como consequência, as questões deveriam favorecer que fatos novos pudessem emergir dos dados coletados. Com esse objetivo em mente, foi definido que as questões fechadas seriam somente aplicadas para caracterizar as informações demográficas gerais dos voluntários envolvidos e a percepção coletiva desses voluntários sobre os problemas investigados. Os motivos para essa medida são dois: (i) habilitar pesquisas futuras relacionadas a esta, com participantes que tenham perfil semelhante — por exemplo, para o propósito de comparação ou triangulação; (ii) obter uma ideia geral do ponto de vista dos participantes nas questões antes da realização da análise qualitativa propriamente.

Um outra preocupação foi manter os voluntários engajados ao longo do processo de responder o *survey*. Esta questão foi especialmente importante neste estudo porque o questionário era online e os voluntários eram livres para encerrar sua participação quando desejassem. Também, uma queda na qualidade das respostas enquanto os participantes preenchiam o *survey*, poderia arriscar a qualidade da fase de codificação — e sua confiabilidade como consequência. Depois de vários refinamentos (por exemplo, integrando questões similares, tornando-as mais concisas e coerentes) o questionário passou a ter 24 questões em sua versão final — metade aberta, metade fechada. Adicionalmente, tornamos as perguntas mais claras, concisas e aplicamos algumas adaptações estruturais. O *survey* foi dividido em um máximo de quatro páginas. As três primeiras páginas continham as questões centrais. A última era dedicada apenas a comentários/sugestões e informações de contato — as quais não eram fundamentais para a análise. As questões fechadas também foram interpoladas com as questões abertas sempre que possível, ao invés de dividi-las em dois grupos distintos. A estratégia subjacente era reduzir a fadiga para ambos os tipos de questões. Além disso, desejava-se oferecer a oportunidade dos voluntários fazerem pausas e elaborarem as respostas abertas subsequentes, ao invés de se manterem no fluxo. As questões do *survey* podem ser consultadas no Apêndice: B.

7.4 Questões de Pesquisa

A atividade de teste unitário pode envolver diferentes sub-atividades — denominadas de práticas — por exemplo, o processo criativo para a formulação de casos de teste; as estratégias para escolher e aplicar diferentes técnicas de teste (estrutural, baseada em erros) sobre as unidades; os métodos para reportar os resultados de teste de unidade; as tarefas de colaboração entre os profissionais para o gerenciamento do teste unitário em um projeto; as estratégias para integrar os testes unitários em uma base de código em um projeto etc. Além disso, existem múltiplas ferramentas associadas com cada uma dessas práticas, tornando impraticável abordar todas as práticas de uma vez.

Neste estudo, a identificação de *flaky tests* é um dos problemas que se desejava explorar. Um *flaky test* é um teste que pode passar ou falhar de forma imprevisível, tornando este teste não confiável. Esses tipos de testes podem originar de enganos cometidos durante a geração dos testes. Além disso, o testador não consegue prever, perceber ou confiar em seus resultados com facilidade. O problema de *flaky tests* tem recebido atenção considerável da comunidade de pesquisa (LUO et al., 2014), (GYORI et al., 2015), (ELOUSSI, 2015). Logo, um dos interesses do estudo foi saber se a ferramenta poderia alavancar as habilidades do praticante na identificação de *flaky tests*, após estes testes unitários terem sido criados.

Este estudo focou-se e limitou-se à exploração da prática de revisão de testes unitários. A revisão de testes unitários inclui qualquer tarefa realizada pelos praticantes que envolva checar ou verificar códigos, resultados e propriedades de casos de teste unitários existentes.

A elaboração das questões de pesquisa é fundamental para o planejamento de estudos qualitativos. As seguintes questões e sub-questões de pesquisa foram formuladas para orientar este estudo:

RQ1. Quais são as tarefas de revisão de teste de unidade que requerem suporte cognitivo de ferramentas, para auxiliar os praticantes de testes?

> RQ1.1 Quais são as tarefas (se houver) que exigem suporte das ferramentas para a compreensão de artefatos de teste de unidade?
>
> RQ1.2 Quais são as tarefas (se houver) que exigem suporte das ferramentas para orientação e direcionamento do testador durante a revisão do teste de unidade?
>
> RQ1.3 Quais são as características das ferramentas (se houver) que podem ser melhoradas para suportar as interações do testador com a ferramenta durante a revisão do teste de unidade?
>
> RQ1.4 Quais são as tarefas (se houver) que requerem suporte das ferramentas para a identificação de *flaky tests* durante a revisão do teste de unidade?

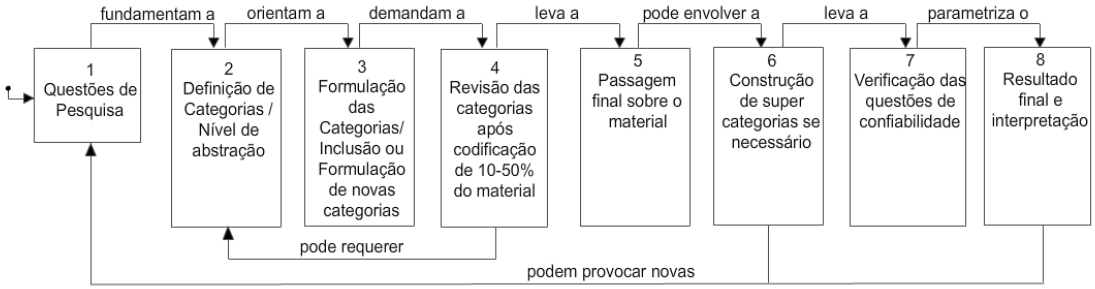

Figura 7.1: Modelo de processo para o desenvolvimento indutivo de categorias (redesenhado e adaptado de Mayring (MAYRING, 2014))

7.5 Planejamento e Processo de Codificação

O processo de codificação no QCA não é guiado por um processo linear e exato tal como o teste de hipótese de uma abordagem quantitativa-positivista. Diferentemente dos métodos quantitativos, o objetivo de um estudo qualitativo como QCA é revelar a partir dos dados, quais são as dimensões que compõem o fenômeno estudado — e não entender como o dado é distribuído nas dimensões de um fenômeno previamente conhecido. Ao se usar QCA, os caminhos seguidos durante a análise são influenciados pela experiência do grupo de pesquisa com métodos qualitativos, a auto-crítica mantida durante o processo de codificação e a qualidade das respostas obtidas. Ainda assim, QCA é mais sistemático que "*grounded theory*" e as práticas que são válidas para "*grounded theory*" por vezes não são adequadas ao QCA. Considerando isso, adotamos um conjunto de passos (Figura 7.1) inspirado no modelo para desenvolvimento indutivo de categorias de Mayring (2014).

Conforme definido no Capítulo 1, o propósito principal deste estudo foi revelar necessidades de suporte cognitivo considerando a perspectiva dos praticantes. Isso implica que as categorias deveriam emergir diretamente das respostas dos participantes, de maneira indutiva. Entretanto, conforme observado por Mayring (2014), mesmo em uma abordagem indutiva "o nível ou os temas das categorias a serem desenvolvidas precisam ser definidos previamente. Precisa haver um critério para o processo de seleção na formação das categorias. O critério é um elemento dedutivo e estabelecido dentro de considerações teóricas sobre o assunto e os objetivos da análise. Nesse sentido, o *framework* definido no Capítulo 5 estabeleceu os fundamentos para o desenvolvimento indutivo das categorias.

No primeiro passo do processo, foi elaborada uma questão de pesquisa central associada a quatro sub-questões (veja a Seção 7.4). A elaboração das questões de pesquisa

seguiram as recomendações de Creswell (2013a) e Miles, Huberman e Saldana (2013) as quais incluem:

- A definição da questão central de forma ampla e exploratória;
- O foco no conceito sobre o qual mais se deseja saber;
- O uso de verbos exploratórios nas questões de pesquisa;
- A especificação sobre quem são os participantes;
- A associação de sub-questões com o propósito de clarificar e especificar a questão central;

As dimensões cognitivas do *framework* descrito no Capítulo 5 ajudaram na formulação das sub-questões. As sub-questões foram definidas de maneira a destacar aspectos importantes do estudo e simultaneamente, deixar o questionamento em aberto. Desta forma, elas são mais comparáveis a "lentes" que ajudam a lembrar e observar características importantes do problema pesquisado, do que a "limitações" que poderiam restringir a análise.

No segundo passo, foram observadas algumas recomendações da literatura para estruturar os critérios de codificação (*coding-frame*):

- As codificações a serem incluídas em uma determinada categoria deveriam ter "significados e conotações similares" (WEBER, 1990);
- Uma categoria emergente precisa ser parte de um fenômeno endereçado nas questões de pesquisa (MAYRING, 2014);
- As subcategorias incluídas em uma dada categoria deveriam ser "mutualmente exclusivas e exaustivas"' (CROWLEY; DELFICO, 2013).

Além disso, foram estabelecidos os seguintes critérios de *definição de categoria* e de *nível de abstração* para o sistema de categorias:

Definição de categorias — tarefas relacionadas a: dificuldades envolvendo decisões sobre artefatos de teste unitário; problemas envolvendo o raciocínio sobre artefatos de teste unitário; necessidade de lidar com questões de revisão de teste unitário fora do ambiente de teste; atrito (*friction*) no uso das interfaces gráficas dos ambientes de teste; problemas relacionados à ocorrência de *flaky test*.

Nível de abstração — dificuldades relacionada a experiências concretas e pessoais; questões que poderiam ou deveriam estar sujeitas ao suporte de ferramentas; problemas que afetam o teste no nível unitário, mesmo que não exclusivamente.

7.5.1 Condução da Codificação

O terceiro passo é o estágio no qual a codificação é iniciada. Neste estudo, foram analisados 58 formulários válidos. Do total de 12 questões abertas em cada formulário, as

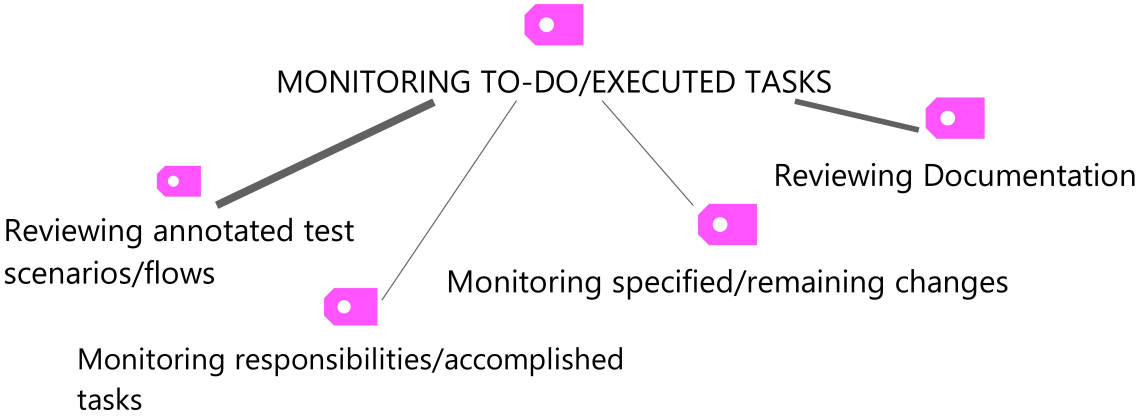

Figura 7.2: Uma ramificação do sistema de categorias final, contendo uma categoria de alto-nível e 4 categorias inclusas. Os rótulos das categorias foram mantido na língua original (inglês) por terem sido gerados automaticamente a partir dos dados da codificação.

10 primeiras foram consideradas para o propósito de codificação. As 2 últimas questões apenas coletaram informações pessoais dos participantes. O conjunto de respostas foi totalmente lido, assim como os resultados das questões fechadas, antes do processo de codificação ser iniciado. Essa abordagem foi aplicada para obtermos imersão no material coletado. Definiu-se que a unidade de codificação seria qualquer elemento semântico claramente definido no texto — seja ela uma frase única ou uma sequência de orações. A opção por essa forma dinâmica de segmentação foi feita por entendermos que cada participante elaborou suas respostas em um nível de detalhamento distinto.

Nós consideramos o conjunto de respostas de cada questão por vez, antes de proceder às questões seguintes. Isto permitiu contrastar as diferenças entre as perspectivas dos participantes nos problemas levantados por cada questão. Também ajudou a obter uma percepção real dos fatores comuns presentes em cada questão abordada, e foi especialmente útil nas iterações iniciais — quando as primeiras categorias emergiram. Esta estratégia está alinhada com a ideia de "construir o *coding-frame*[1] para um 'pedaço' [do material] depois de outro" (SCHREIER, 2013).

Uma ramificação do diagrama do sistema de categorias é utilizada para ilustrar como parte do processo de codificação evoluiu (ver Figura 7.2). Inicialmente, alguns segmentos codificados conduziram indutivamente às categorias representadas pelos nós folhas — "*Reviewing annotated test scenarios/flows", "Monitoring responsibilities/accomplished tasks", "Monitoring specified/remaining changes" and "Reviewing documentation*". Entretanto, estas categorias não emergiram todas de uma vez. Nós ite-

[1]Neste trabalho, o termo "*coding-frame*" corresponde ao termo "sistema de categorias", seguindo a terminologia de (SCHREIER, 2013)

rativamente e colaborativamente revisamos cada nova categoria proposta. Nós verificávamos repetidamente se um segmento codificado se encaixava em alguma das categorias existentes em todo o sistema de categoria. Caso contrário, consultávamos os critérios pré-definidos de *definição de categorias* e *nível de abstração* para garantir que a nova categoria seria apropriada para o sistema de categorias. Se notássemos que as categorias estavam intimamente relacionadas, considerávamos se deveríamos criar uma categoria de nível superior — neste exemplo a categoria "*Monitoring to-do/executed tasks*" — a qual pudesse incluí-las. Isto também requeria verificar se as categorias a serem inclusas tinham alcançado algum nível de consistência; se elas eram de fato coesas em relação à categoria de nível superior proposta; e se elas eram mutualmente exclusivas.

O material codificado foi revisto (passo 4) depois que uma quantidade substancial de categorias emergiram na primeira tentativa de codificação. Nesse ponto, notamos que algumas categorias precisavam ser corrigidas devido a uma sobreposição de conceitos — estávamos inconscientemente interpretando duas ideias distintas como uma única. Mayring (2014) recomenda que a *definição da categoria* e o *nível de abstração* sejam revistos quando conceitos ou categorias sobrepostas forem detectados. Caso eles precisem de modificação, a análise do material deve ser refeita desde o início. Desta maneira, ajustamos o nível de abstração, descartamos o conjunto de categorias obtidas até esse ponto e reiniciamos o processo. Esta decisão foi útil para restabelecer nossa confiança na obtenção de um conjunto de categorias consistente. Além disso, ajudou a reforçar a importância de rever a formação das categorias durante a codificação. Tomamos algumas medidas adicionais para mitigar as chances de ter o mesmo problema de sobreposição de conceitos novamente: (i) esperamos algumas semanas para recomeçar o processo; (ii) analisamos cada questão em uma ordem diferente daquela da primeira tentativa; (iii) randomizamos o conjunto de respostas para cada pergunta. Esta estratégia nos ajudou a apagar o processo de codificação anterior de nossas mentes — ou pelo menos torná-lo menos vívido.

O processo de codificação foi repetidamente revisado e checado (passos 5 e 6) até alcançar-se saturação isto é, quando as categorias emergentes esgotaram-se e não havia nada para adicionar ou refinar no sistema de categorias. A Figura 7.3 mostra o sistema de categorias que emergiu da análise. Uma cópia das respostas codificadas encontra-se disponível em (PRADO, 2017).

Os passos 7 e 8 são discutidos adiante na Seção 7.7.1, dedicada aos resultados do estudo e considerações de confiabilidade.

7.6 Perfil e Perspectiva dos Participantes

A análise dos resultados das questões objetivas (CQs) forneceu um panorama de como os participantes da pesquisa, tomados em grupo, enfrentam alguns problemas

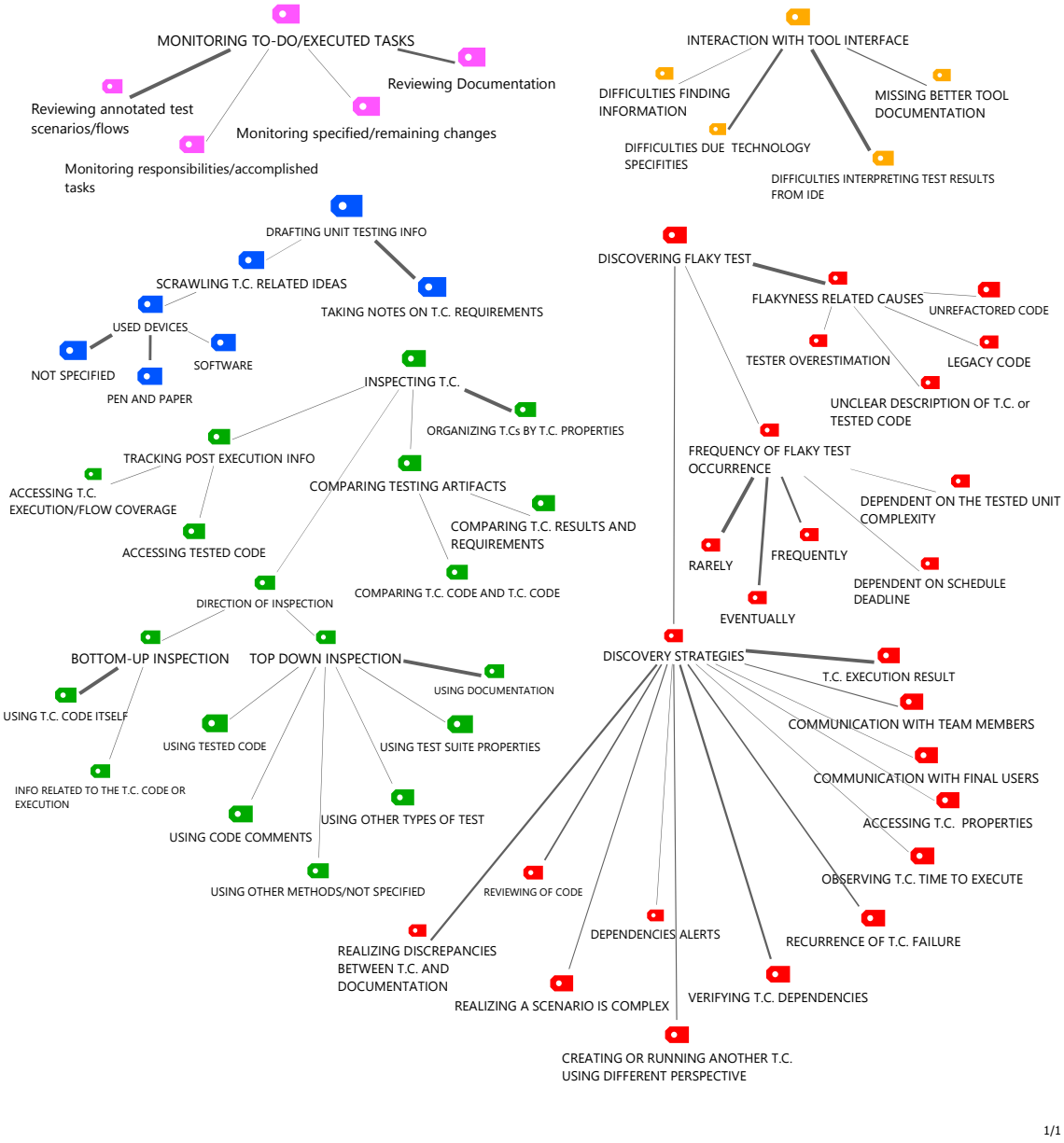

Figura 7.3: O sistema de categorias finalizado. Quanto mais espessa a aresta, mais segmentos codificados existem na categoria filha.

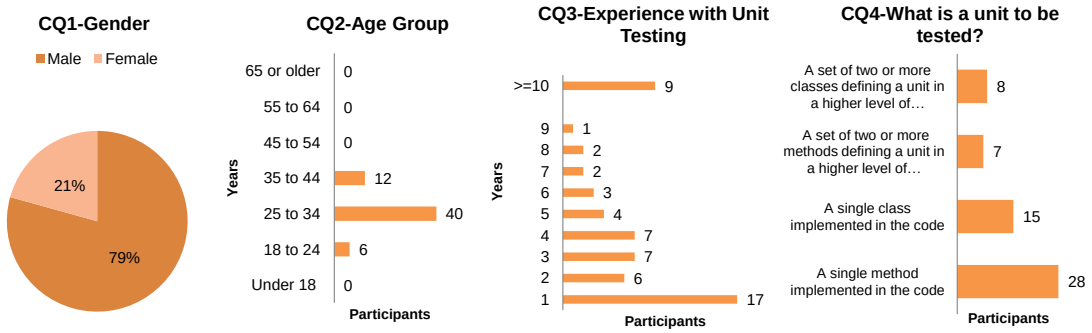

Figura 7.4: Gráfico com os dados demográficos dos voluntários que participaram deste estudo. Os gráficos foram mantidos na língua original (inglês) por terem sido gerados automaticamente a partir dos dados fornecidos pelos participantes.

relacionados às questões abertas. Essa análise, somada a uma leitura cuidadosa de todas as resposta abertas, nos ajudou a ganhar imersão no material obtido antes de proceder ao processo de codificação. Conforme observado anteriormente, o único critério adotado de amostragem dos participantes foi que eles tivessem experiência prévia com o teste unitário. Entretanto, estudos futuros que desejem expandir ou triangular com os resultados deste estudo podem querer selecionar participantes que partilhem de uma opinião similar à dos voluntários deste trabalho. Nesse sentido, os resultados das respostas objetivas podem ser reusadas como critério para escolha de participantes que sejam comparáveis aos deste estudo. Apresentamos os resultados de nossas questões objetivas nas subseções seguintes.

7.6.1 Informações Demográficas dos Participantes

Duas abordagens diferentes foram utilizadas para recrutar participantes para este estudo. Na primeira abordagem, convites foram postados em uma rede social profissional e em fóruns online de desenvolvedores. Na segunda, enviamos e-mails diretamente aos nossos pares no setor de tecnologia, de forma que eles pudessem encaminhar as cartas convites para potenciais candidatos em seus círculos. Ao final, um total de 44 respostas foi recebido na primeira abordagem e 17 na segunda. Entretanto, três respostas obtidas pela segunda abordagem foram consideradas inválidas e descartadas — totalizando 58 respostas válidas.

O gráfico na Figura 7.4 mostra os dados demográficos dos participantes. A maioria deles são do sexo masculino (79%). A maioria dos participantes (68.9%) têm entre 25 e 34 anos de idade e nenhum dos participantes tem mais que 44 anos de idade. Aproximadamente um terço dos voluntários (29.3%) têm um ano de experiência com o teste unitário. O número de participantes com maior experiência diminuiu à medida que o

total de anos de experiência com o teste unitário aumentou. Como não foi possível prever o número máximo de anos de experiência dos voluntários, simplificamos essa questão não distinguindo os participantes com 10 anos ou mais de experiência (15.5%). Também quisemos saber como os voluntários geralmente definem uma unidade a ser testada. A maioria dos participantes escolheram uma definição mais simplista, definindo a unidade a ser testada como um único método ou classe implementada no código (total de 74.1%) ao invés de defini-la como um conjunto desses elementos.

7.6.2 Perspectiva dos Participantes Sobre os Problemas

O problema de motivar os participantes na atividade de teste unitário foi previamente reportado na literatura por Runeson (2006) e Daka e Fraser (2014). Similarmente, neste estudo quisemos saber como os participantes percebiam suas motivações durante a revisão de seus próprios testes. O gráfico ilustrado na Figura 7.5–CQ5 mostra que a revisão dos testes unitários não é percebida nem como causa de entusiasmo, nem como uma prática tediosa pelos participantes. Ao cruzar os dados de CQ5 com os dados de CQ6 –CQ5 x CQ6), o gráfico revela que, em geral, os participantes têm uma opinião uniformemente distribuída sobre o quanto a interface gráfica da ferramenta contribui com sua motivação. Os participantes que responderam "muito motivados" tiveram uma opinião mais direta em isentar ou culpar a experiência com a interface gráfica pela sua motivação. Embora não causem surpresa, as observações anteriores trazem alguma confiança com relação à confiabilidade das respostas restantes. As avaliações moderadas observadas na primeira e segunda observação anterior mostram que a maioria dos voluntários não estavam dispostos a serem condescendentes e concordarem com qualquer coisa que lhes perguntassem — o que seria uma ameaça denominada de viés de aquiescência (*acquiescence bias* (KNOWLES; NATHAN, 1997)). Por outro lado, a atitude "tudo ou nada" notada na terceira observação mostrou que os participantes muito motivados não estavam preocupados em dar respostas imprecisas nos assuntos tratados, de forma a mostrarem somente a melhor faceta de suas opiniões — o que representaria uma ameaça denominada de viés de "desejabilidade social". (*social desirability*)' (DODOU; WINTER, 2014).

Na área da psicologia da programação, o conceito de "modelos mentais"(veja Seção 3.3) é aplicado para descrever como ocorre o entendimento de um programa pelo programador, durante a leitura do código do programa. De acordo com esse conceito, "entender um programa é equivalente a construir uma representação adequada da situação [que o código representa]" (DETIENNE, 2001).

Os comentários de código são recursos adicionais que dão suporte ao programador para construir o modelo mental do programa — ele adiciona informação à interpretação do código. Considerando que o código do caso de teste implementado é também

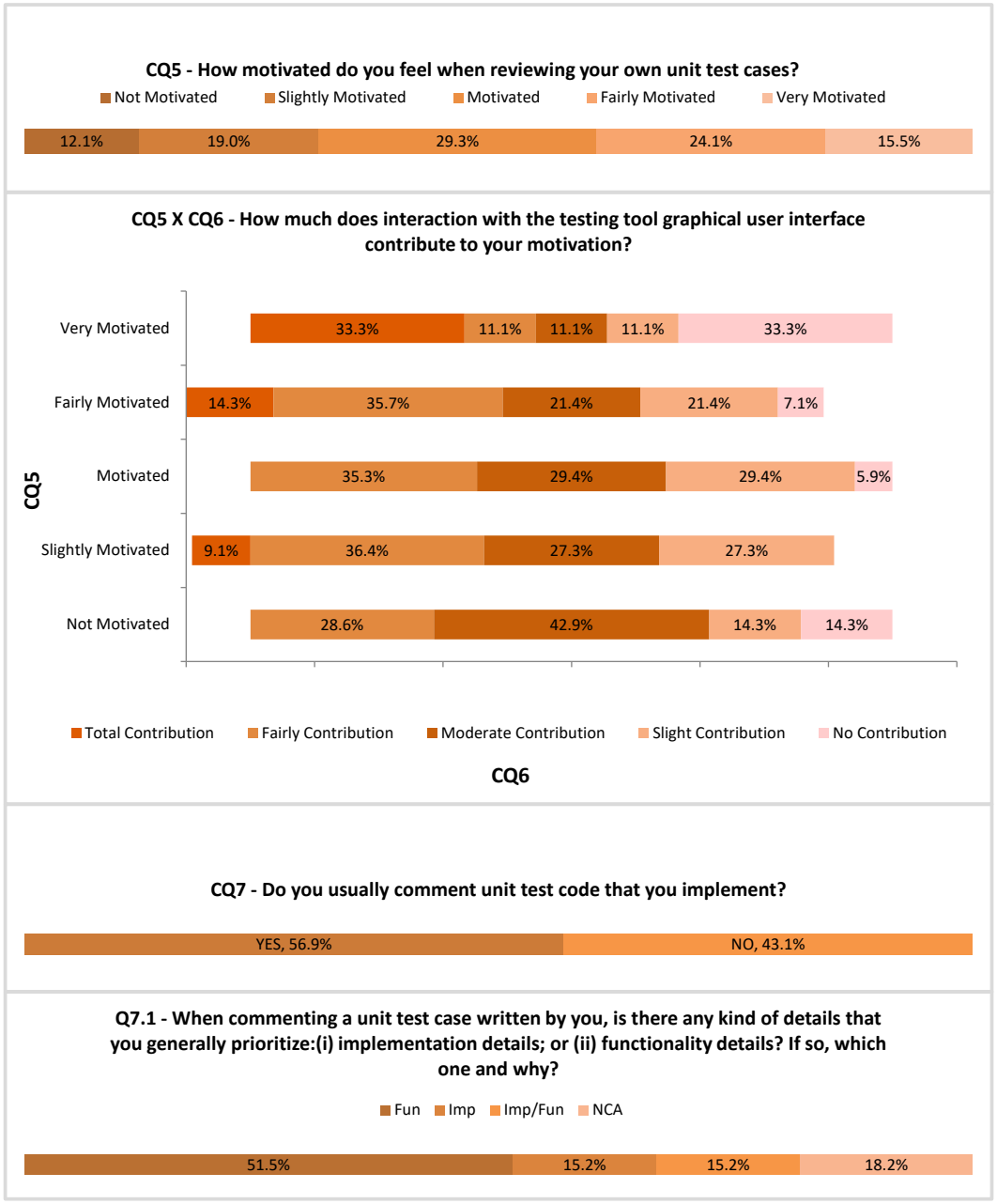

Figura 7.5: Questões objetivas e resultados (primeira parte). Os gráficos foram mantidos na língua original (inglês) por terem sido gerados automaticamente a partir dos dados fornecidos pelos participantes.

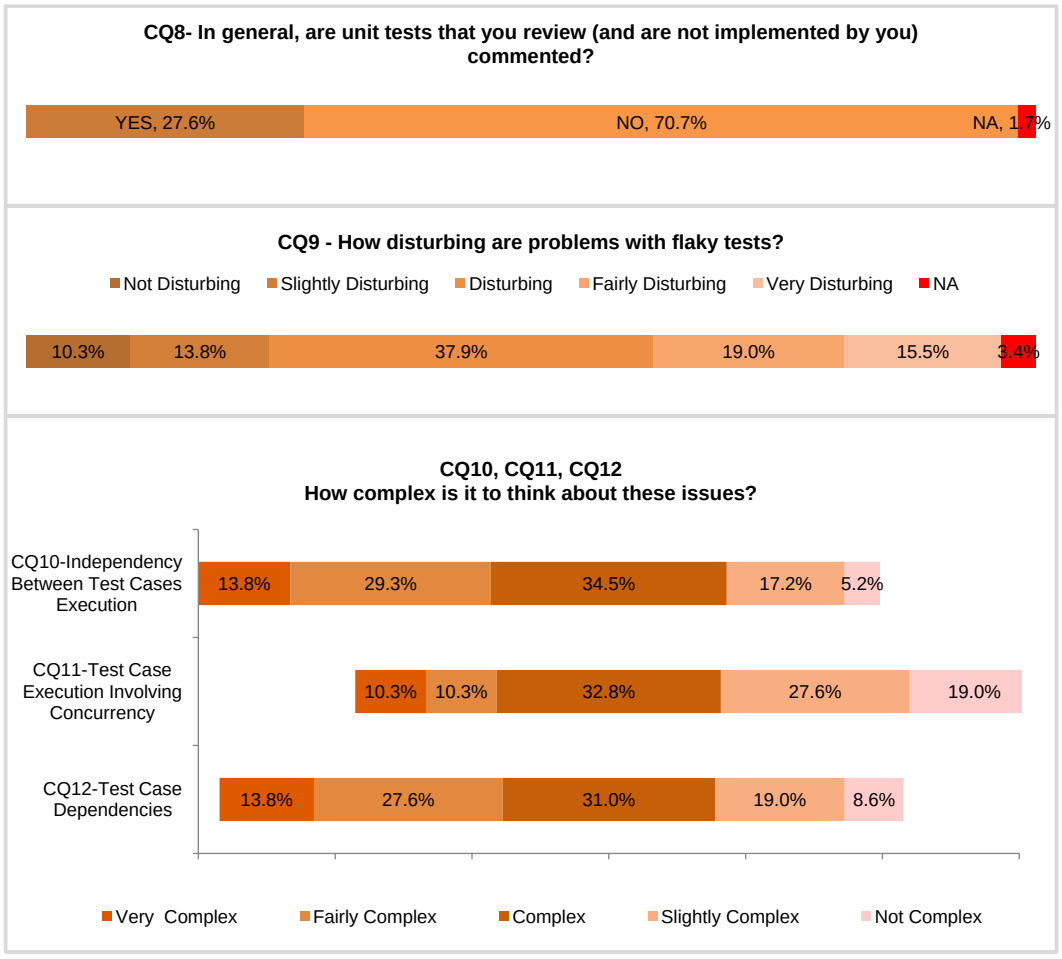

Figura 7.6: Questões objetivas e resultados (segunda parte). Os gráficos foram mantidos na língua original (inglês) por terem sido gerados automaticamente a partir dos dados fornecidos pelos participantes.

um programa, duas questões fechadas foram propostas para entender se, na opinião dos participantes, os casos de teste implementados por eles próprios (CQ7) e por outros desenvolvedores (CQ8) são geralmente comentados. As respostas foram balanceadas entre "sim" e "não" para comentários no próprio caso de teste (CQ7). Por outro lado, as respostas para (CQ8) mostraram uma percepção diferente: 71.9% responderam que os casos de teste que não foram criados por eles mesmos são, em geral, não comentados. As observações anteriores poderiam ser mais precisas se pudéssemos fazer medições diretas dos comentários de códigos presentes nas rotinas de cada voluntário. Entretanto, tal medição demandaria acesso a uma quantidade considerável de código das instituições em que os voluntários trabalham e de seus colegas de trabalho. Isso durante um tempo considerável, além de gerar preocupações relacionadas à privacidade e propriedade intelectual. Esta medição está além do escopo deste estudo, o qual foca nas experiências dos praticantes.

Considerando que o propósito primário das questões objetivas é a caracterização dos participantes — e não de seus artefatos — os resultados obtidos são suficientes para entender como estes voluntários: (i) percebem o uso desse suporte imediato em seus próprios casos de teste; e (ii) percebem o uso do mesmo suporte quando os casos de teste vêm de terceiros.

Ainda dentro do paradigma de modelos mentais, a psicologia da programação distingue dois tipos de representação mental durante o entendimento do programa pelo usuário: o modelo de programa e o modelo situacional. Em resumo, o modelo de programa corresponde ao conhecimento explícito e superficial no código textual e sua estrutura (aspectos de implementação). Isso pode ser entendido em dois níveis: micro e macro. No nível micro, o modelo de programa corresponde às representações das operações elementares e do fluxo de controle entre essas operações. No nível macro, o modelo de programa refere-se às representações dos métodos e estruturas maiores (classes, por exemplo). O modelo situacional corresponde a representações de alto nível (aspectos de funcionalidade) as quais podem ser estáticas e dinâmicas por exemplo, os objetos do domínio do problema; o relacionamento entre os objetos (por exemplo, herança, composição); os objetivos do programa; a comunicação inter-objeto. Considerando isso, uma questão aberta extra (CQ7.1) foi proposta para aqueles que responderam "sim" para (CQ7). A questão perguntava se havia algum detalhe — detalhe de implementação ou de funcionalidade — que o participante geralmente priorizava quando comentava seus códigos de caso de teste unitário. As respostas obtidas foram classificadas em quatro grupos distintos: "priorização da funcionalidade (Fun)" (17/33), "Mesma prioridade para implementação e funcionalidade (ImpFun)" (5/33);"priorização da implementação(Imp)"(5/33); e "Não respondido claramente (NCA)" (6/33). De acordo com esses resultados, a maioria dos participantes que comentam seus próprios casos de teste priorizam a funcionalidade do caso de teste em seus comentários. Considerando que os aspectos da funcionalidade estão relacionados aos elementos de domínio e objetivos do caso de teste, interpretamos esses resultados como uma intenção de escrever comentários que reflitam o modelo situacional dos casos de teste.

Os resultados de CQ9 (Figura 7.6 destacam o quão perturbantes são os "*flaky tests*" para os participantes: 72.4% deles classificaram o problema como "perturbante", "relativamente perturbante" ou "muito perturbante". No trabalho de (LUO et al., 2014) três características estão comumente associadas com os *flaky tests*: a dependência entre casos de teste, testes associados à execução concorrente; e dependência dos casos de teste (por exemplo, teste dependendo de um serviço externo, conteúdo de arquivo, recurso de rede, etc). Desta forma, três questões (CQ10, CQ11 e CQ12) foram propostas para avaliar a percepção desses problemas na revisão dos testes unitários. No gráfico de barra empilhada divergente na Figura 7.6–CQ10, CQ11 e CQ12 pode-se observar que "a

independência entre a execução de casos de teste" e a "dependência de casos de teste" são questões mais complicadas para eles comparadas à "execução do caso de teste envolvendo concorrência".

7.7 *Framework* de Tarefas que Requerem Suporte Cognitivo

Nessa seção, é apresentado o *framework* de tarefas que requerem suporte cognitivo das ferramentas, para a revisão de testes unitários. A descrição do *framework* é apresentada da forma mais específica possível, baseada nas categorias que emergiram do processo de análise.

Cinco termos agrupadores foram usados para estruturar as tarefas no *framework*: "Auto-monitoramento", "Inspeção de casos de teste", "Rascunho da informação de teste unitário", "Descoberta de *flaky tests*" e "Interação com a interface da ferramenta". Cada grupo envolve uma ou mais tarefas e elas não ocorrem necessariamente isoladas. Por exemplo, o rascunho de informações de teste unitário pode ocorrer em paralelo com as tarefas de inspeção de casos de teste.

- **Auto-monitoramento**–dar suporte aos usuários no seu auto-gerenciamento durante as tarefas de revisão de teste unitário

    1. *Suporte ao monitoramento de tarefas executadas/por fazer.* — Os praticantes planejam e monitoram o progresso feito por eles mesmos durante a revisão do teste unitário. Esta prática de auto-monitoramento é heterogênea e atualmente feita por meio de memorização, anotações avulsas e revisões de diferentes tipos de mídias como documentos, diagramas, *post-its* e arquivos texto. Os problemas de auto-monitoramento são relacionados a: (i) monitorar as responsabilidades e tarefas que ele/ela efetuou; (ii) revisar anotações para lembrar o objetivo da unidade de teste e determinar qual cenário ou fluxo de execução resta ser coberto; (iii) monitorar quais mudanças foram especificadas para os testes; (iv) revisar a documentação para lembrar do propósito do teste unitário. Alguns segmentos codificados notáveis sobre esta questão foram: *"Tenho que sempre ter em mente que os testes de unidade existem para testar MEU trabalho, e não um outro código de terceiros ou até mesmo recursos da linguagem[de programação]."; "Eu costumo observar em um arquivo de texto quais os cenários que eu preciso fazer "; "Em geral escrevo algumas notas sobre as entradas e saídas da unidade testada para ajudar a entender os objetivos da unidade"; " Eu sempre usar notas para manter na memória e organizar o propósito de criação de teste de tabelas de verdade "; "Verifico a*

documentação e comparo os cenários esperados e resultados [para lembrar o propósito do teste de unidade]"

- **Inspeção de casos de testes**–dar suporte ao entendimento, organização e comparação dos testes.

    2. *Suporte à inspeção top-down e bottom-up do caso de teste/contexto*–Os profissionais inspecionam os casos de teste alternando duas direções diferentes de análise: *top-down* e *bottom-up*. Na inspeção *top-down*, o entendimento do contexto do caso de teste precede o entendimento do próprio caso de teste e é aplicado para facilitar a interpretação do comportamento do teste e seu objetivo. Alguns exemplos desse tipo de análise são: a revisão do código que é exercitado pelo caso de teste; a leitura da documentação do software e diagramas relacionados; a recordação de propriedades da suíte de teste ou dos casos de teste isto é, convenção de nomenclatura, critério que o teste satisfaz; a consulta de anotações, comentários de código ou mesmo de outros tipos de testes, por exemplo, casos de teste de integração e de sistema. Nas palavras de alguns dos participantes:*"Tento rever o código/funcionalidade que está sendo testado (para contextualização) e depois descrever como o teste é conduzido (para facilitar a leitura)"; " o nome da suíte de teste do contexto de teste (caso Rspec) é muito importante. Também, o nome do arquivo de teste ajuda a entender o propósito do teste."*, ou ainda *"Às vezes eu preciso de muito tempo para entender o cenário de teste"*. Durante a inspeção *bottom-up*, os praticantes analisam um determinado caso de teste para obter um entendimento de seu propósito e contexto. Segmentos codificados que exemplificam este tipo de análise são: *" eu tenho que refatorar o teste porque ele tem que ser suficientemente claro para descrever o software."; "Eu uso muito o JUnit dentro do Eclipse e eu tenho experiência com jasmine/mocha para testes de javascript e o último suporta contextualizar os testes de uma maneira melhor (com describe/it por exemplo). Seria bom se JUnit pudesse fazer o mesmo."; "Em geral, eu tomo algumas notas sobre os problemas que foram encontrados durante a execução dos testes [para lembrar o propósito deles]"*.
    3. *Suporte à comparação entre artefatos de teste unitário*– Os testadores precisam recorrer a diferentes fontes de informação para avaliar se os resultados do teste estão corretos ou não (verificação de oráculo). Estas fontes variam desde informações diretamente fornecidas pelos usuários finais até cenários de caso de uso prescritos, planos de teste, etc. Além disso, os profissionais geralmente comparam dois ou mais testes unitários (ou artefatos associados) durante a refatoração do código do caso de teste. Conforme indicado em alguns segmentos codificados:*"Eu checo a documentação e comparo os cenários esperados*

e resultados"; "Algumas vezes eu consulto outros casos de teste para ver se eu esqueci algum cenário que é recorrente em todos os projetos; [Eu uso] verificação de assertivas e avaliação de similaridade entre casos de teste para evitar redundâncias."

4. *Facilitar o rastreamento de informações pós-execução dos testes*–A visualização das unidades testadas e sua cobertura são duas importantes informações que os profissionais procuram após a execução do caso de teste. De acordo com um dos participantes: "A ligação com o código testado deveria ser mais interativa e dinâmica de maneira a inspecionar o código em caso de erro/falha."; "o fluxo de execução do caso de teste poderia ser descrito graficamente, com um grafo ou diagrama de atividade [para facilitar a interpretação]"
5. *Suporte à busca e organização de casos de teste de acordo com propriedades de interesse*–Durante a revisão dos testes, os profissionais precisam localizar e rearranjar os casos de teste de acordo com propriedades de interesse. Nas palavras dos participantes: *"[...] as IDEs não são muito específicas em indicar onde as classes de teste estão." ou "[A] organização dos testes é muito confusa. Seria melhor organizá-las de uma forma que os desenvolvedores/testadores pudessem definir o que classificar... agrupar por classes, métodos... dar 'dicas' de testes similares"*.

- **Rascunho de informações de teste unitário**–suportar o rascunho de informações relevantes para a análise do teste unitário.

6. *Suporte à manipulação de rascunhos relacionados ao teste unitário*–Os praticantes fazem anotações e esboços na forma de diagramas e desenhos. Estas notas e rascunhos são registrados em papel ou software. Estes rascunhos funcionam como um auxílio de memória para dar suporte em tarefas como: o raciocínio sobre os requisitos de teste e a execução dos testes; compreensão do propósito da unidade sob teste. Alguns segmentos codificados que ilustram preocupações com essas questões são: raciocinar sobre requisitos e execução de casos de teste; entender o propósito da unidade sob teste. Alguns segmentos codificados que ilustram preocupação com essas questões são: *"Uma representação gráfica das entradas e saídas do módulo, bem como o próprio módulo, pode ser útil na revisão dos casos de teste"*; *"Ao executar testes de unidade, no aspecto funcional, uso papel e caneta para listar partições de equivalência e gráfico de decisão. No aspecto estrutural, uso papel e caneta para montar um CFG para ajudar"*; *"Geralmente [eu] escrevo algumas algumas notas sobre as entradas e saídas da unidade testada para ajudar a entender os objetivos da unidade*; *"Desenhar caixas conectadas ajuda a organizar o raciocínio e a identificar as entradas e saídas dos testes"*. Estas

tarefas partilham alguma similaridade com aquelas cobertas nos itens anteriores "Suporte ao monitoramento de tarefas executadas/por fazer". Entretanto, estes diagramas e anotações são orientados à compreensão do comportamento dos testes ao invés do auto-monitoramento dos profissionais.

- **Descoberta de Flaky test**–suporte à revelação de *flaky tests*

  7. *Suporte à aplicação de estratégias para revelar flaky tests no teste unitário*– Os profissionais mencionam o reuso de código legado, a imprecisão da descrição de casos de teste e a superestimação dos profissionais de teste durante a criação dos teste como fatores que contribuem para a ocorrência de *flaky tests*. Além disso, os profissionais adotam algumas estratégias que ajudam na descoberta de *flaky tests*, tais como: (i) a verificação e organização dos casos de teste de acordo com propriedades tais como tempo para executar cada teste e a complexidade dos casos de teste; (ii) a comunicação com os usuários e os membros da equipe; (iii) a observação de alertas disparados por unidades testadas ou dependências de casos de teste; (iv) notar se a unidade testada é executada sob um cenário complexo; (v) a criação de outros casos de teste — por exemplo, um caso de teste para a mesma unidade em um outro cenário; (vi) a observação da recorrência de falhas em uma execução de caso de teste; (vii) a observação do número de dependências dos casos de teste e seus detalhes; (viii) a observação de discrepâncias entre os casos de teste e sua documentação (por exemplo, devido a mudanças frequentes, novos cenários, descrição pouco clara).

- **Interação com a interface da ferramenta** –Suporte à interação do usuário com a interface da ferramenta.

  8. *Simplificar os resultados de teste exibidos pela IDE e melhoria da documentação das ferramentas*–Os praticantes de teste unitário enfrentam problemas com a interpretação dos resultados de teste unitário apresentados pelas *IDEs*. Estas dificuldades variam desde o grande volume de dados produzidos na execução dos teste, até a forma de apresentar os dados. Além disso, foram identificados problemas associados à documentação das ferramentas. Conforme expressado em alguns dos segmentos codificados: *"Quando ocorre um erro, as IDEs poderiam prover uma informação consolidada ao invés de informações tão detalhadas. Se o testador quiser mais detalhes, isso pode ser feito em um segundo passo."*; *"Eu gosto do plugin JUnit no Eclipse. Entretanto, os plugins para Maven/Ant enchem o saco. Você tem que ler todas aqueles arquivos de log malucos para entender alguma coisa quando algo dá errado"*; *"Algumas vezes as saídas podem tornarem-se difíceis de ler por conta dos*

traces da pilha de exceção impressos como uma mensagem de log"; *"Algumas ferramentas não oferecem a documentação detalhada de uma maneira apropriada"*.

7.7.1 Discussão e Considerações de Confiabilidade

A perspectiva adotada nesta pesquisa não foi — e não poderia ser — suficiente para capturar todos os aspectos do fenômeno "teste unitário". Ao invés disso, conforme estabelecido anteriormente, a análise foi restringida para capturar apenas questões de suporte cognitivo da prática de revisão de teste, o interesse principal desta pesquisa.

Tentamos ser o mais preciso possível na descrição das tarefas que requerem suporte cognitivo, baseando a descrição nas categorias emergidas e seus respectivos segmentos codificados. Alguns itens do *framework* como o "Suporte à inspeção *top-down* e *bottom-up* do caso de teste/contexto", o "Suporte à comparação entre artefatos de teste unitário" e "Suporte à busca e organização de casos de teste de acordo com propriedades de interesse" mostram a importância de recuperar e acessar múltiplos tipos de informação durante a revisão dos testes unitários. Outros itens como o "Suporte ao monitoramento de tarefas executadas/por fazer" ou o "Suporte à manipulação de rascunhos relacionados ao teste unitário" exemplificam como a pratica de revisão de teste unitário relaciona-se com a forma particular com que cada profissional pensa e gerencia a sua atividade. Estas tarefas estão relacionadas aos processos de *enriquecimento* dentro da perspectiva da teoria de fornecimento de informações (*Information Foraging* (PIROLLI; CARD, 1999)) e correspondem às atividades que os profissionais executam para modelar o ambiente para se adequarem às estrategias disponíveis (por exemplo, criar "*patches*" para ajudar a processar a informação rapidamente (BURNETT et al., 2009)). Ao dar suporte à execução dessas tarefas, essas ferramentas podem emponderar os usuários a focarem sua atenção em decisões críticas do processo de revisão de teste — e encurtar o caminho entre raciocinar sobre um problema e trabalhar em sua solução.

A seguir, são apresentadas as considerações de confiabilidade com relação à credibilidade, transferibilidade, confiança e validabilidade (GUBA, 1981) desta pesquisa.

O *survey* deste estudo resultou de um processo de refinamento meticuloso (veja Seção 7.3.1) para facilitar o entendimentos das questões pelos participantes — uma preocupação com a credibilidade na obtenção dos dados. O processo incluiu: a execução piloto do *survey*; o refinamento progressivo das questões com a simplificação do vocabulário; e a explicação da terminologia, antes de seu uso no questionário, para evitar má interpretação (por exemplo, a definição do que são "*flaky tests*"). Além disso, a participação dos voluntários e a política de uso de seus dados passou pelo escrutínio do comitê de ética em pesquisa de nossa instituição (parecer consubstanciado disponível

no Apêndice A). O termo de compromisso aprovado no comitê de ética contém várias cláusulas que visam incentivar a honestidade dos voluntários do estudo: o compromisso de tornar a participação do voluntário confidencial e anônima. A liberdade do voluntário aceitar, rejeitar e ignorar o envolvimento no estudo; a garantia de que guardaríamos suas respostas de forma segura e sob nossa responsabilidade; a garantia de que seus resultados seriam tornados anônimos antes de qualquer possível publicação; o uso exclusivo de seus dados apenas para os propósitos estabelecidos. Por fim, o QCA é um método de análise sistemático que tem sido aplicado com sucesso em uma ampla variedade de áreas científicas, especialmente em áreas nas quais os clamores dos participantes são o foco da atenção — por exemplo, enfermagem e psicologia. O QCA é um método flexível e conciso que permite variantes diferentes para o desenvolvimento das categorias, dependendo do propósito do estudo. A escolha da formação indutiva de categorias permitiu desenvolver as categorias diretamente a partir do material. Tais características reforçaram a adequação deste método para os objetivos desta pesquisa e contribuíram para a melhoria da credibilidade da análise.

Com relação à transferibilidade, algumas observações devem ajudar a transmitir ao leitor as limitações deste estudo. Conforme observado por Ritchie e Lewis (2003) “A generalização inferencial na pesquisa social também deve ser uma questão de julgamento e o papel do pesquisador é fornecer a "descrição grosseira"do contexto pesquisado e os fenômenos encontrados (visualizações, processos, experiências, etc.) que permitirão a outros avaliar sua transferência para outra configuração. Tal como acontece com a generalização teórica, a inferência deve permanecer como uma hipótese até ser provada ou refutada por evidências adicionais”. Os voluntários são oriundos de um conjunto bem diversificado de organizações — um total de 29 instituições baseado nos participantes que optaram por fornecer esta informação. Não restringimos a participação dos voluntários no estudo baseado em qualquer critério além de que eles tivessem experiência com o teste unitário. Para maximizar as chances de obter respostas mais enriquecedoras, foi preciso alcançar o maior número de voluntários possível. De um total de 61 respostas, 58 foram consideradas válidas — dois questionários foram respondidos em branco e um admitiu a falta de experiência com o teste unitário no preenchimento das respostas. Além disso, a qualidade das respostas válidas foi suficiente para permitir que as categorias atingissem saturação durante o processo de codificação. Logo, consideramos que o número e a qualidade das respostas obtidas adequaram-se ao objetivo e ao escopo delimitado para esta pesquisa. Além disso, a descrição do *framework* é apresentada em uma forma narrativa precisa, baseada nas categorias e subcategorias que emergiram.

Informações aprofundadas sobre os processos aplicados neste estudo (Seção 7.5) foram apresentadas para melhorar a confiança e validabilidade dos resultados. Isso pode ser observado, por exemplo, na Seção 7.5.1 na qual são descritos os passos adicionais

tomados para mitigar o problema de sobreposição de conceitos durante o processo de codificação. Também, considerando que os participantes são oriundos de diferentes organizações, não é possível ter qualquer controle sobre o seu contexto. Logo, questões fechadas foram propostas — relacionadas aos problemas discutidos nas questões abertas — para capturar e medir características que pudessem tornar os participantes deste estudo mais comparáveis como grupo (ver Seção 7.3). Desta forma, estudos futuros que desejem replicar ou triangular com este estudo podem reusar essas questões fechadas e seus respectivos resultados para selecionar participantes com uma perspectiva mais próxima da opinião dos voluntários deste trabalho. Os instrumentos de coleta de dados adotados também contribuíram para a melhoria da confiança dos resultados. O questionário online é um mecanismo neutro de interrogação. Portanto, ele é menos sujeito a fatores aleatórios indesejados que possam influenciar as respostas dos participantes como por exemplo: a linguagem corporal, a entonação da voz ou o contato visual do entrevistador. Outra vantagem do questionário online é que as respostas já são recebidas diretamente em formato textual, e a transcrição da resposta torna-se desnecessária. Além disso, novos estudos podem fazer uso imediato do questionário definido e promover uma comparação mais consistente dos resultados (PRADO; VINCENZI, 2017).

Apesar das preocupações com a confiabilidade, o trabalho tem limitações que deixam espaço para melhorias. Por exemplo, um estudo de triangulação usando outras formas de interrogação (por exemplo, entrevista semi-estruturada) conduzida por outros pesquisadores e considerando outros conjuntos de participantes, poderia contribuir para a melhoria da credibilidade e validabilidade. Voluntários adicionais e a oportunidade de coletar dados diretamente de seus locais de trabalho poderiam ajudar a tornar os resultados ainda mais representativos e favorecer a transferibilidade. Em um paradigma qualitativo, é importante reconhecer que os dados textuais precisam de interpretação, que pode variar mais do que a interpretação de números mensuráveis, e os pesquisadores assumem a inevitável responsabilidade de interpretar as respostas durante o processo de codificação. Essa é a razão pela qual discutimos questões de generalização em termos de "transferibilidade"em vez de discutir "validade externa". Assim, tais considerações de confiabilidade não podem ser comparadas na mesma base de comparação que considerações de validade no paradigma quantitativo.

Finalmente, destaca-se que os resultados deste estudo representam contribuições inéditas na área, mesmo com as limitações apresentadas. Este trabalho coloca o ser-humano no centro da análise, para pesquisar a evolução das ferramentas de teste unitário. O *framework* apresentado é um modelo teórico que categoriza tarefas, a partir da perspectiva dos praticantes, as quais devem ser apoiadas por ferramentas para alavancar as habilidades próprias dos praticantes. Além disso, o estudo permite replicação direta

e extensão. Nesse sentido, prepara o caminho para novas pesquisas preocupadas com o problema de suporte cognitivo em outras áreas de teste de software.

7.8 Agenda de Pesquisa

Nesta seção uma agenda de pesquisa é apresentada com base nos resultados deste estudo. A agenda serve como um instrumento acionável para guiar o desenvolvimento dos itens do *framework* em novas pesquisas e aplicações envolvendo o suporte cognitivo.

A agenda de pesquisa é estruturada conforme a seguir: cada subseção da agenda corresponde a diferentes temas de pesquisa isto é, estudos a serem desenvolvidos ou aplicações dos resultados do *framework*. Em cada cabeçalho é indicado o tema de pesquisa, seguido pelas tarefas do *framework* (TFs) associadas.

7.8.1 Pesquisar as Especificidades do Uso de Papel e Caneta *versus* Ferramentas de Software nas Anotações e Esboços do Teste Unitário — {TF: 1 e 6 }

A tomada de notas pode ajudar os seres humanos a lidar com a carga cognitiva e a ampliar suas habilidades cognitivas inatas (DROR; HARNAD, 2008; MAKANY; KEMP; DROR, 2009). De acordo com Kiewra (1987), a "tomada de notas incentiva a organização e a organização durante a tomada de notas aumenta a realização". No entanto, ainda há muito debate sobre os benefícios e as desvantagens da tomada de notas usando papel e caneta *versus* usando ferramentas de software, em relação aos seus efeitos sobre a resolução de problemas e as atividades de aprendizagem (FRIEDMAN, 2014). A área educacional tem sido o ambiente natural para tais investigações. No entanto, a resolução de problemas e a aprendizagem também fazem parte da atividade de teste de software. Conforme mostrado nos resultados, os praticantes de teste de unidade usam ambos os tipos de dispositivos em suas anotações e esboços para gerenciar suas tarefas e resolver problemas de revisão de teste de unidade. Nesse sentido, novas pesquisas devem ser conduzidas para avaliar as vantagens e desvantagens intrínsecas de se usar os dois tipos de ferramentas para os praticantes de testes unitários. Essas pesquisas podem incluir estudos experimentais **(por exemplo, Oviatt et al. (2012), Oviatt et al. (2007), Oviatt, Arthur e Cohen (2006))** para comparar a forma como ambos os tipos de ferramentas, bem como as tecnologias híbridas (por exemplo, dispositivos baseados em caneta digital), facilitam ou dificultam os problemas comuns de revisão de teste unitário por exemplo: o raciocínio sobre os requisitos de testes unitários; a delimitação de responsabilidades; o acompanhamento das tarefas realizadas e os fluxos a serem testados; a modelagem da

unidade e do comportamento do caso de teste; a compreensão dos objetivos dos casos de teste, etc.

7.8.2 Desenvolvimento de Novos Mecanismos de Suporte para Apoio à Tomada de Notas e Esboço no Teste Unitário — {TFs: 1 e 6}

Os projetistas de ferramentas podem propor novos mecanismos de suporte para alavancar os recursos do ferramental de testes atual, baseando-se nos resultados oriundos do tema de pesquisa anterior (Tema 7.8.1). Além disso, conforme observado nos segmentos codificados, os esboços e anotações gerados durante a revisão de testes unitários muitas vezes se referem às pessoas e artefatos de desenvolvimento: colegas de equipe, casos de teste, casos de uso, etc. Neste sentido, as ferramentas poderiam, por exemplo, melhorar o suporte aos testadores permitindo a detecção e ligação das anotações e esboços aos artefatos e *stakeholders* que tenham sido registrados na informação anotada — por exemplo, vinculando as anotações aos usuários ou arquivos correspondentes no *Github* ou a um lembrete no calendário. No entanto, estas e novas ideias oriundas de estudos futuros devem ser co-validadas com os usuários. Para isto, os pesquisadores precisam escolher profissionais dispostos a dar *feedback* contínuo sobre estas e outras questões do projeto da interação — como mostrado em Muto, Okano e Kusumoto (2011a). Isso inclui, mas não está limitado a: decidir se os novos mecanismos de suporte devem ser integrados ou separados das *IDEs* existentes; determinar as formas preferidas de se recuperar e organizar as anotações e esboços; escolher metáforas apropriadas para usar na interação com a interface; definir métodos de entrada e saída para registro e leitura das anotações e esboços. Além disso, é importante que os usuários sejam selecionados de diferentes origens — isto é, usuários com diferentes níveis de experiência no teste unitário; com diferentes perfis sociodemográficos; com experiência em diferentes linguagens/tecnologias de desenvolvimento — uma vez que fatores pessoais podem influenciar decisões no projeto da interação (CHALMERS, 2003; BURNETT et al., 2009). Caso a implementação desses mecanismos de suporte envolvam tecnologias de entrada/saída que os desenvolvedores ainda não estão acostumados em seu ambiente de trabalho (por exemplo, uma mesa digitalizadora), o uso de sondas tecnológicas pode ser útil. Uma sonda tecnológica é um instrumento usado para explorar aspectos desconhecidos sobre novas tecnologias. Ela promove o criticismo dos usuários e dá a eles a liberdade de adaptar a tecnologia proposta de novas maneiras criativas. Além disso, é apropriada para ambientes complexos e privativos, nos quais aprender sobre a atitude dos usuários em relação à nova tecnologia pode ser uma situação desafiadora (HUTCHINSON et al., 2003).

7.8.3 Pesquisa de Mecanismos de Suporte que Mitiguem o Risco de Sobrecarga Cognitiva Durante a Análise *Top-down e Bottom-up* no Teste Unitário — {TFs: 2 e 3}

De acordo com a Teoria da Carga Cognitiva, impedimentos, distrações e informações desnecessárias enfrentadas durante o processo de aprendizagem aumentam a carga externa na memória de trabalho e contribuem para a sobrecarga cognitiva (SWELLER, 1988; SWELLER, 1999; MERRIëNBOER; SWELLER, 2005). Considerando que processos de aprendizagem estão envolvidos na compreensão das unidades e de seus casos de teste, a navegação errática pelas informações relativas ao teste unitário pode dificultar as atividades de inspeção *bottom-up* e *top-down*. Um exemplo disso é, por exemplo, ter que revistar diversos artefatos de desenvolvimento para encontrar as informações desejadas, ou debruçar-se em múltiplas descrições redundantes para aprender sobre uma unidade testada e suas dependências. Assim, propõe-se mecanismos de apoio ao testador destinados a facilitar a associação e acesso a informações contextuais dos testes unitários, e recomenda-se pesquisas que avaliem o quão bem esses mecanismos atenuam esses problemas. Por exemplo, a recuperação imediata da documentação específica e dos diagramas associados a um determinado caso de teste unitário, sem perder o contexto atual de visualização do código do caso de teste, reduz a sobrecarga cognitiva? Quais são os efeitos positivos e negativos na compreensão das unidades testadas e seus casos de teste pelos testadores? As obras de Paas et al. (2003), Chen et al. (2013) e Oviatt, Arthur e Cohen (2006) fornecem informações úteis para o projeto de uma configuração experimental para avaliar questões de carga cognitiva.

7.8.4 Projeto de Mecanismos de Recuperação de Informações Contextuais Para e a partir de Testes Unitários — {TFs: 2 e 3 }

O desenvolvimento de qualquer mecanismo de suporte cognitivo, conforme proposto no tema de pesquisa anterior 7.8.3, requer alguns estudos prévios. Por exemplo, como os profissionais geralmente combinam as informações dos artefatos de software durante a análise *top-down*? Quais ferramentas e técnicas estão envolvidas neste processo? Estes e outros aspectos técnicos podem variar entre comunidades de desenvolvimento. Entretanto, estes elementos são necessário para uma proposta de projeto de interação efetiva, e portanto novos estudos de usuário são necessários. Como sugestão, um estudo de usuários em três estágios — composto de um estudo grupo focal, seguido por um estudo de campo e finalmente por entrevistas contextuais, podem ajudar a suprir essa demanda. No primeiro estágio, os pesquisadores devem tentar entender os artefatos, tecnologias e métodos comuns adotados pelos usuários de uma dada comunidade — por exemplo, as

técnicas e estratégias de teste unitário usadas por desenvolvedores de aplicativos android. Além disso, o primeiro estágio serviria para fortalecer os laços entre os participantes e prepará-los para os estágios subsequentes. No estudo de campo, os pesquisadores devem observar como os usuários realizam as análises *bottom-up* e *top-down* em seu ambiente de rotina, sob as condições reveladas na fase anterior. Baseando-se nessas observações, os pesquisadores devem compilar os desafios comuns enfrentados pelos usuários na realização das análises *top-down* e *bottom-up*. As entrevistas contextuais deverão ajudar a resolver as questões que emergirem durante o estudo de campo e a explorar diferentes ideias de soluções de suporte a serem incorporadas nas ferramentas. Para o processo de projeto das soluções, os pesquisadores podem aplicar prototipagem rápida para amadurecer os requisitos de suporte com os participantes.

7.8.5 Pesquisa e Desenvolvimento de Ferramentas Co-Adaptativas para Auxiliar a Busca, Comparação, Organização e Rastreamento das Unidades de Teste e seus Resultados — {TFs: 3, 4, 5, 7 e 8}

Os profissionais regularmente precisam localizar, comparar e reorganizar os casos de teste e unidades testadas de diferentes modos durante a revisão dos testes unitários. Além disso, eles alternam com frequência entre a visualização dos códigos dos casos de teste e os resultados correspondentes. Encontrar uma maneira padrão de apoiar essas tarefas é difícil devido aos diferentes contextos dos profissionais — ferramental de desenvolvimento, metodologias e processos adotados (por exemplo, Test *Driven Development versus* Test After Development (ERDOGMUS; MORISIO; TORCHIANO, 2005)). Além disso, cada indivíduo pode enfrentar esses problemas de forma diferente. Assim, qualquer mecanismo de suporte para essas tarefas deve ser personalizável e aprender com o fluxo de trabalho específico do usuário. Nesse sentido, o usuário poderá adaptar-se sem problemas ao novo suporte e *vice-versa*. O uso de aprendizagem de máquina pode contribuir para esta co-adaptação. Por exemplo: (i) com base nos casos de teste e nas propriedades das unidades testadas que o usuário acessa comumente, a ferramenta poderia sugerir como ordenar ou comparar os casos de teste — por exemplo, casos de teste associados às mesmas dependências, exercitando o mesmo caminho no código ou revelando os mesmos tipos de erros; (ii) a ferramenta poderia sugerir testes unitários que provavelmente deveriam ser refatorados se aprendesse que o acesso extensivo e a modificação nas dependências da unidade testada levam à modificação nos testes de unidade associados; (iii) a ferramenta poderia sugerir testes de unidade obsoletos com base em critérios que podem ser correlacionados como a baixa frequência de visualização dos testes pelo usuário e o teste da unidade incluso por um teste mais efetivo — por exemplo, um teste unitário que cobre o

mesmo caminho e revela mais erros. O trabalho de Oviatt (2006) discute vários princípios de design centrados no usuário associados com a melhoria do desempenho humano. O trabalho de Mackay (1990) fornece a base teórica para entender o fenômeno co-adaptativo entre usuários e softwares personalizáveis. Mathur, Miles e Du (2015) monstraram como alguns mecanismos de suporte adaptativo e métodos de aprendizagem de máquina podem ser usados, por exemplo, para facilitar a execução de testes dirigidos pela interface do usuário.

7.8.6 Novos Estudos para Apoiar as Estratégias de Descoberta de *Flaky Tests* — {TF: 7 }

Várias estratégias que os profissionais usam para ajudar a revelar *flaky tests* emergiram neste estudo. Embora alguns dos temas de pesquisa anteriores apoiem essas estratégias — por exemplo, o Tema 7.8.5 " Pesquisa e desenvolvimento de ferramentas co-adaptivas (...) " para a estratégia "(i) a verificação e organização dos casos de teste de acordo com propriedades como o tempo de execução de cada teste e a complexidade do caso de teste;" — pesquisas adicionais são necessárias para elucidar outras formas de suporte. Por exemplo, considerando que *a comunicação com os usuários e os membros da equipe* é uma estratégia que emergiu neste estudo para revelação de *flaky tests*, pode-se investigar como a colaboração pode prover suporte para alavancar a realização dessa tarefa. Tais estudos poderiam informar melhorias para ferramentas colaborativas e canais de comunicação, a fim de facilitar a descoberta de *flaky tests* — não só para o teste de unidade, mas também para os outros níveis de teste. A pesquisa nesse sentido é muito oportuna. Empresas como o Google, por exemplo, mostraram recentemente seu interesse no problema ao designar uma equipe dedicada a informar os desenvolvedores sobre os danos provocados por *flaky tests*(MICCO, 2016).

7.8.7 Reformulação da Apresentação dos Resultados do Teste e Documentação das Ferramentas — {TF: 8 }

A maneira como as *IDEs* atualmente apresentam os resultados dos testes de unidade foi uma grande queixa relacionada à usabilidade que emergiu neste estudo. Os designers de ferramentas devem promover aos testadores uma melhor experiência nesse sentido, o que inclui, entre outros: a reformulação e a apresentação de informações de execução de teste em formas mais consolidadas; a opção de explorar detalhes da execução dos testes; e a habilidade de subdividir esta exploração em passos sucessivos. Alguns métodos que os pesquisadores poderiam aplicar para resolver esses problemas: (i) testes de usabilidade moderados (VASALOU et al., 2004; BARNUM, 2010) e estudos de

eyetracking (PERNICE; NIELSEN,) poderiam ajudar a entender e encontrar padrões de interação do usuário com as ferramentas de teste de unidade por exemplo: para identificar o que chama a atenção na *GUI* e os fatores que causam isto (para determinar o que erroneamente chama a atenção do usuário, de modo a otimizar o reconhecimento da informação desejada do resultado de teste); (ii) a técnica de pensar em voz alta pode ajudar a descobrir como os usuários inspecionam os *logs* de execução do teste da unidade — e se a informação encontrada coincide com suas expectativas; (iii) a técnica de *card sorting* pode ser usada para reorganizar a estrutura de informações do *log* em uma ordem que faça sentido para os usuários e que atenda suas necessidades.

7.9 Considerações Finais

Neste Capítulo foi detalhado um estudo qualitativo realizado para determinar as demandas de suporte cognitivo para a prática de revisão de testes unitários, considerando a perspectiva de profissionais reais. Inicialmente, são feitas as devidas considerações sobre o propósito e a adequação do método de análise qualitativa empregado no estudo *(Qualitative Content Analysis)*. Em seguida, explica-se como os participantes foram selecionados e detalha-se o processo de elaboração do questionário (*survey*) aplicado. Além disso, as questões de pesquisa que orientaram o estudo são apresentadas; o planejamento, a condução do processo de codificação das respostas abertas e o sistema de categorias resultante são detalhados. Em seguida, os resultados das questões fechadas foram usados para delinear o perfil geral dos participantes e descrever a perspectivas dos mesmos sobre assuntos relacionados aos problemas abordados nas questões abertas. Na sequência, o *framework* de tarefas que requerem suporte cognitivo é apresentado. Este *framework* foi obtido com base no sistema de categorias que emergiu do processo de codificação. Por fim, discute-se as considerações de confiabilidade do estudo e uma agenda de pesquisa é apresentada com base no *framework* proposto.

CAPÍTULO 8

Conclusão

Esta tese de doutorado descreve uma sequência de quatro etapas na pesquisa do problema do suporte cognitivo provido pelas ferramentas de teste no teste unitário.

Na primeira etapa (Capítulo 4) a literatura foi revisada para identificar as soluções que aplicam a visualização no apoio à atividade de teste de software. Inicialmente, foi conduzido um mapeamento sistemático (piloto) orientado a identificar os trabalhos que empregam visualização na atividade de compreensão de software em geral. Este primeiro mapeamento serviu para entender os aspectos gerais de visualização comumente considerados nas técnicas e ferramentas existentes e serviu para obter confiança quanto ao processo de planejamento e condução de um mapeamento sistemático. Com o conhecimento e experiência adquiridos nesse primeiro estudo, um novo mapeamento sistemático foi realizado, desta vez mais específico e orientado às ferramentas e técnicas que empregam a visualização no apoio à atividade de teste de software. Neste novo estudo buscou-se: (i) entender como os trabalhos selecionados se distribuem entre três importantes biblioteca digitais na área de teste (IEEE, ACM e ScienceDirect); (ii) caracterizar o perfil destas propostas com relação a aspectos de teste (e.g. sua distribuição entre os diferentes níveis e fases de teste), aspectos de visualização (e.g. os tipos de substratos e de interação frequentemente empregados) e aspectos de treinamento; (iii) por fim, identificou-se com relação à avaliação das propostas: o precário detalhamento das avaliações na maioria dos trabalhos e o baixo número de avaliações no contexto da indústria.

Na segunda etapa (Capítulo 5) alguns desafios foram superados: destacou-se como a comunidade de teste tem negligenciado os aspectos humanos na pesquisa das ferramentas de teste; o problema da carência de suporte cognitivo no teste unitário foi identificado e proposto, sendo este um problema de pesquisa inédito na área; caracterizou-se como esse novo problema tem encontrado eco em demandas reportadas esparsamente na literatura; e evidenciou-se a necessidade de uma abordagem de pesquisa centrada nos usuários e em suas perspectivas. Além disso, um *framework* inicial foi proposto, o qual é composto de três dimensões de demandas cognitivas no teste unitário. Este *framework* inicial serviu para orientar o desenvolvimento dos passos seguintes da pesquisa.

Na terceira etapa (Capítulo 6) buscou-se entender como as ferramentas de teste unitário que usavam visualização (identificadas no Capítulo 4) abordavam questões de suporte cognitivo. Para isso, examinou-se cada trabalho selecionado, sob as dimensões cognitivas propostas no *framework* inicial. Os resultados revelaram diversas lacunas de suporte cognitivo e oportunidades de melhoria. Além disso, descobriu-se que a falta de envolvimento dos potenciais usuários na pesquisa foi um fator comum entre todos os trabalhos com problemas de suporte cognitivo.

Considerando-se os resultados obtidos nas três etapas anteriores, foi realizado um estudo com profissionais de teste que têm experiência no teste unitário (Capítulo 7). Um *survey* qualitativo com 24 questões foi elaborado para entender como as ferramentas poderiam melhorar o suporte cognitivo provido aos usuários, na prática de revisão de testes unitários. Um total de 61 voluntários responderam ao *survey*, sendo 58 dessas participações consideradas válidas. Na etapa de análise, o método de Análise de Conteúdo Qualitativo (QCA) foi aplicado em sua forma indutiva para elicitar as categorias de problemas de suporte cognitivo considerando a perspectiva dos voluntários. Com base nas categorias que emergiram na análise, um *framework* de tarefas que requerem suporte cognitivo foi proposto. Em seguida, foram feitas as considerações de confiabilidade do estudo. Por fim, mostrou-se a aplicabilidade do *framework* por meio de uma agenda de pesquisa baseada no mesmo.

Juntos, os resultados deste trabalho provêm um conjunto valioso de informação para a evolução das ferramentas de teste de software atuais e futuras—sejam elas ferramentas *standalone*, *plugins* de IDEs ou adaptações nos ambientes em que essas ferramentas são utilizadas. O *framework* obtido por meio do estudo qualitativo leva em consideração problemas correntes que afetam praticantes reais da atividade de teste. Pesquisadores, desenvolvedores de ferramentas de teste proprietárias ou terceirizadas podem usar estas contribuições como orientação para atender necessidades que têm sido negligenciadas ou pobremente suportadas pelas ferramentas atuais. Estas melhorias têm o potencial de impactar a atividade de teste unitário e, consequentemente, a qualidade final do software sob teste. Conforme destacado nos temas da agenda de pesquisa, qualquer proposta que vise melhorar o suporte cognitivo deve envolver a co-participação de praticantes do mundo real, não somente nos estágios finais de validação, mas desde sua concepção. Esse envolvimento é necessário para assegurar uma abordagem centrada no usuário e um processo de melhoria do suporte cognitivo que seja satisfatório.

Esta tese provê contribuições originais para o entendimento e endereçamento de um problema largamente negligenciado na pesquisa de teste de software. Além dos temas propostos na agenda de pesquisa, novos estudos futuros podem abordar o suporte cognitivo em outras práticas do teste unitário e em outros níveis de teste e.g. o processo criativo durante o planejamento e implementação de casos de testes unitários; práticas de

teste de desempenho; teste de integração e sistema etc. Finalmente, novos estudos que desejem replicar ou ampliar os resultados obtidos nesse trabalho são bem vindos. Eles deverão ajudar a reforçar a importância de envolver humanos na pesquisa de ferramentas de teste. Consequentemente, deverão contribuir para tornar o teste de software uma atividade mais prazerosa e eficiente para os profissionais de teste—aqueles que atuam na linha de frente pela melhoria da qualidade de software.

Referências Bibliográficas

ACREE, A. T. et al. *Mutation Analysis*. Atlanta, GA, set. 1979.

ADOLPH, S.; HALL, W.; KRUCHTEN, P. Using grounded theory to study the experience of software development. *Empirical Software Engineering*, v. 16, n. 4, p. 487–513, ago. 2011. ISSN 1382-3256, 1573-7616.

AGRAWAL, H. *Design of Mutant Operators for the C Programming Language*. [S.l.], mar. 1989.

AMMANN, P.; OFFUTT, J. *Introduction to Software Testing*. 1 edition. ed. New York: Cambridge University Press, 2008. ISBN 9780521880381.

APA - American Psychological Association. *Glossary of Psychological Terms*. 2017. [Online]. Available: http://www.apa.org/research/action/glossary.aspx. [Accessed: Aug. 20, 2017].

ATKINSON, C.; BARTH, F.; BRENNER, D. Software Testing Using Test Sheets. In: *2010 Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW)*. [S.l.: s.n.], 2010. p. 454–459.

BARBOSA, E. F.; MALDONADO, J. C.; VINCENZI, A. M. R. Toward the determination of sufficient mutant operators for C. *Software Testing, Verification and Reliability Journal*, v. 11, n. 2, p. 113–136, jun. 2001. John Wiley & Sons.

BARNUM, C. M. *Usability testing essentials: ready, set... test!* [S.l.]: Elsevier, 2010.

BECK, K. *Test Driven Development: By Example*. 1 edition. ed. Boston: Addison-Wesley Professional, 2002. ISBN 9780321146533.

BECK, K. *JUnit Pocket Guide*. 1 edition. ed. Beijing ; Sebastopol, Calif: O'Reilly Media, 2004. ISBN 9780596007430.

BEIZER, B. *Software Testing Techniques*. 2. ed. New York: Van Nostrand Eeinhold, 1990.

BERTIN, J. *Semiology of Graphics: Diagrams, Networks, Maps*. 1 edition. ed. Redlands, Calif: Esri Press, 2010. ISBN 978-1-58948-261-6.

BESHERS, C.; FEINER, S. Autovisual: rule-based design of interactive multivariate visualizations. *IEEE Computer Graphics and Applications*, v. 13, n. 4, p. 41–49, July 1993. ISSN 0272-1716.

BINDER, R. V. *Testing Object-Oriented Systems: Models, Patterns, and Tools*. [S.l.]: Addison Wesley Longman, Inc., 1999.

BOSHERNITSAN, M.; DOONG, R.; SAVOIA, A. From daikon to agitator: lessons and challenges in building a commercial tool for developer testing. In: *In ISSTA '06: Proceedings of the 2006 International Symposium on Software Testing and Analysis*. [S.l.]: ACM Press, 2006. p. 169–180.

BRERETON, P. et al. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, v. 80, n. 4, p. 571–583, abr. 2007. ISSN 0164-1212. Disponível em: <http://www.sciencedirect.com/science/article/pii/S016412120600197X>.

BRIAND, L. Embracing the engineering side of software engineering. *IEEE Software*, v. 29, n. 4, p. 96–96, July 2012. ISSN 0740-7459.

BRITTO, T. C. P.; PIZZOLATO, E. B. Proposta De Guidelines De Interfaces Com Foco Em Aspectos Do Autismo. In: *Companion Proceedings of the 13th Brazilian Symposium on Human Factors in Computing Systems*. Porto Alegre, Brazil, Brazil: Sociedade Brasileira de Computação, 2014. (IHC '14), p. 37–40. Disponível em: <http://dl.acm.org/citation.cfm?id=2738165.2738177>.

BROD, M.; TESLER, L. E.; CHRISTENSEN, T. L. Qualitative research and content validity: developing best practices based on science and experience. *Quality of Life Research: An International Journal of Quality of Life Aspects of Treatment, Care and Rehabilitation*, v. 18, n. 9, p. 1263–1278, nov. 2009. ISSN 1573-2649.

BUDD, A. T. Mutation analysis: Ideas, examples, problems and prospects. In: ____. [S.l.]: North-Holland Publishing Company, 1981. (Computer Program Testing), p. 129–148.

BURNETT, M. et al. End-user software engineering and distributed cognition. In: *2009 ICSE Workshop on Software Engineering Foundations for End User Programming*. [S.l.: s.n.], 2009. p. 1–7.

CARD, S. K.; MACKINLAY, J. D.; SHNEIDERMAN, B. *Readings in information visualization: using vision to think*. San Francisco, Calif.: Morgan Kaufmann Publishers, 1999. ISBN 1558605339 9781558605336.

CARD, S. K.; MACKINLAY, J. D.; SHNEIDERMAN, B. (Ed.). *Readings in Information Visualization: Using Vision to Think*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. ISBN 1-55860-533-9.

CARROLL, J. M. *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*. [S.l.]: MIT Press, 1990. Google-Books-ID: lKcmAAAAMAAJ. ISBN 978-0-262-03163-9.

CHALMERS, P. A. The role of cognitive theory in human–computer interface. *Computers in human behavior*, Elsevier, v. 19, n. 5, p. 593–607, 2003.

CHEN, F. et al. Multimodal behavior and interaction as indicators of cognitive load. *ACM Trans. Interact. Intell. Syst.*, ACM, New York, NY, USA, v. 2, n. 4, p. 22:1–22:36, jan. 2013. ISSN 2160-6455. Disponível em: <http://doi.acm.org/10.1145/2395123.2395127>.

COLANZI, T. E. *Uma abordagem integrada de desenvolvimento e teste de sofware baseada na UML*. Dissertação (Mestrado) — ICMC-USP, São Carlos, SP, abr. 1999.

CONROY, K. M. et al. Automatic test generation from gui applications for testing web services. In: *2007 IEEE International Conference on Software Maintenance*. [S.l.: s.n.], 2007. p. 345–354. ISSN 1063-6773.

CORBIN, J.; STRAUSS, A. Basics of qualitative research: Techniques and procedures for developing grounded theory. In: ____. [S.l.]: SAGE Publications, Inc, 2014. cap. 3: Practical Considerations for Getting Started, p. 31–56.

COTTAM, J. A.; HURSEY, J.; LUMSDAINE, A. Representing unit test data for large scale software development. In: *Proceedings of the 4th ACM Symposium on Software Visualization*. New York, NY, USA: ACM, 2008. (SoftVis '08), p. 57–66. ISBN 978-1-60558-112-5. Disponível em: <http://doi.acm.org/10.1145/1409720.1409730>.

CRAIK, K. J. W. *The Nature of Explanation*. [S.l.]: CUP Archive, 1967. Google-Books-ID: wT04AAAAIAAJ. ISBN 978-0-521-09445-0.

CRESWELL, J. W. Qualitative inquiry and research design: Choosing among five approaches. In: ____. 2nd. ed. [S.l.]: Sage Publications, Inc, 2007. cap. Qualitative Inquiry and Research Design, p. 40. ISBN 1412916062,9781412916066.

CRESWELL, J. W. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, 4th Edition*. 4th edition. ed. Thousand Oaks: SAGE Publications, Inc, 2013. ISBN 978-1-4522-2610-1.

CRESWELL, J. W. Research design: Qualitative, quantitative, and mixed methods approaches, 4th edition. In: ____. 4th edition. ed. Thousand Oaks: SAGE Publications, Inc, 2013. (1, v. 4), cap. 7: Research Questions and Hypotheses, p. 129–143. ISBN 978-1-4522-2610-1.

CRESWELL, J. W. Research design: Qualitative, quantitative, and mixed methods approaches, 4th edition. In: ____. 4th edition. ed. Thousand Oaks: SAGE Publications, Inc, 2013. (1, v. 1), cap. 2: Review of the Literature, p. 25–50. ISBN 978-1-4522-2610-1.

CROWLEY, B. P.; DELFICO, J. F. *Content Analysis: A Methodology for Structuring and Analyzing Written Material: PEMD-10.3.1*. [S.l.]: BiblioGov, 2013. ISBN 978-1-287-20595-1.

DAKA, E.; FRASER, G. A Survey on Unit Testing Practices and Problems. In: *2014 IEEE 25th International Symposium on Software Reliability Engineering (ISSRE)*. [S.l.: s.n.], 2014. p. 201–211.

DALTON, A. R. et al. Visualizing the runtime behavior of embedded network systems: A toolkit for TinyOS. *Science of Computer Programming*, v. 74, n. 7, p. 446–469, 2009. ISSN 0167-6423. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167642309000331>.

DANIEL, P.; SIM, K. Y. Dynamic fault visualization tool for fault-based testing and prioritization. In: *Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on*. [S.l.: s.n.], 2012. p. 301–306.

DELAHAYE, M.; BOUSQUET, L. du. Selecting a software engineering tool: lessons learnt from mutation analysis. *Software: Practice and Experience*, p. n/a–n/a, jan. 2015. ISSN 1097-024X. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1002/spe.2312/abstract>.

DELAMARO, M. E. *Mutação de Interface: Um Critério de Adequação Interprocedimental para o Teste de Integração*. Tese (Doutorado) — IFSC/USP, São Carlos, SP, jun. 1997.

DEMILLO, R. A.; LIPTON, R. J.; SAYWARD, F. G. Hints on test data selection: Help for the practicing programmer. *Computer*, IEEE, v. 11, n. 4, p. 34–43, abr. 1978.

DETIENNE, F. *Software Design - Cognitive Aspect*. Softcover reprint of the original 1st ed. 2002 edition. London ; New York: Springer, 2001. ISBN 9781852332532.

DODOU, D.; WINTER, J. de. Social Desirability is the Same in Offline, Online, and Paper Surveys. *Comput. Hum. Behav.*, v. 36, n. C, p. 487–495, jul. 2014. ISSN 0747-5632. Disponível em: <http://dx.doi.org/10.1016/j.chb.2014.04.005>.

DROR, I. E.; HARNAD, S. Offloading cognition onto cognitive technology. In: DROR, I. E.; HARNAD, S. (Ed.). *Benjamins Current Topics*. Amsterdam: John Benjamins Publishing Company, 2008. v. 16, p. 1–23. ISBN 978-90-272-2246-6 978-90-272-8964-3. DOI: 10.1075/bct.16.02dro. Disponível em: <https://benjamins.com/catalog/bct.16.02dro>.

ELLIS, W. D. *A Source Book of Gestalt Psychology*. [S.l.]: Psychology Press, 1999. ISBN 978-0-415-20957-1.

ELOUSSI, L. *Determining flaky tests from test failures*. Tese (Doutorado) — University of Illinois at Urbana-Champaign, 2015. Disponível em: <http://hdl.handle.net/2142/78543>.

ENGSTRöM, E.; RUNESON, P. Test overlay in an emerging software product line – an industrial case study. *Information and Software Technology*, v. 55, n. 3, p. 581–594, mar. 2013. ISSN 0950-5849. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950584912001061>.

ERDOGMUS, H.; MORISIO, M.; TORCHIANO, M. On the effectiveness of the test-first approach to programming. *IEEE Transactions on Software Engineering*, v. 31, n. 3, p. 226–237, March 2005. ISSN 0098-5589.

ERNST, N. A. *Towards cognitive support in knowledge engineering : an adoption-centred customization framework for visual interfaces*. 110 p. Tese (Doutorado) — University of Victoria, 2004.

ERNST, N. A.; STOREY, M.-A.; ALLEN, P. Cognitive support for ontology modeling. *International Journal of Human-Computer Studies*, v. 62, n. 5, p. 553 – 577, 2005. ISSN 1071-5819. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1071581905000261>.

ESTEFO, P. Restructuring unit tests with testsurgeon. In: *2012 34th International Conference on Software Engineering (ICSE)*. [S.l.: s.n.], 2012. p. 1632–1634. ISSN 0270-5257.

FABBRI, S. C. P. F.; VINCENZI, A. M. R.; MALDONADO, J. C. Introdução ao teste de software. In: ____. 1st. ed. [S.l.]: Campus / Elsevier, 2007. cap. Teste Funcional, p. 9,25.

FLICK, U. (Ed.). *The SAGE Handbook of Qualitative Data Analysis*. 1 edition. ed. Los Angeles: SAGE Publications Ltd, 2013. ISBN 978-1-4462-0898-4.

FLICK, U. An introduction to qualitative research. In: ____. [S.l.]: SAGE Publications Ltd, 2014. cap. 6 - Using the Existing Literature, p. 66–69.

FOWLER, M. *Test Pyramid*. 2015. Http://martinfowler.com/bliki/TestPyramid.html. Disponível em: <http://martinfowler.com/bliki/TestPyramid.html>.

FRIEDMAN, M. C. Notes on note-taking: Review of research and insights for students and instructors. *Harvard University*, 2014.

GAROUSI, V.; ZHI, J. A survey of software testing practices in Canada. *Journal of Systems and Software*, v. 86, n. 5, p. 1354–1376, maio 2013. ISSN 0164-1212. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0164121212003561>.

GENTNER, D.; NIELSEN, J. The Anti-Mac Interface. *Commun. ACM*, v. 39, n. 8, p. 70–82, ago. 1996. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/232014.232032>.

GLASER, B. G.; STRAUSS, A. L. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Edição: 00008. New Brunswick u.a.: Aldine, 1999. ISBN 978-0-202-30260-7.

GRECHANIK, M.; XIE, Q.; FU, C. Creating gui testing tools using accessibility technologies. In: *Software Testing, Verification and Validation Workshops, 2009. ICSTW '09. International Conference on*. [S.l.: s.n.], 2009. p. 243–250.

GUBA, E. G. Criteria for assessing the trustworthiness of naturalistic inquiries. *ECTJ*, v. 29, n. 2, p. 75, jun. 1981. ISSN 0148-5806, 1556-6501. Disponível em: <https://link.springer.com/article/10.1007/BF02766777>.

GYORI, A. et al. Reliable Testing: Detecting State-polluting Tests to Prevent Test Dependency. In: *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. New York, NY, USA: ACM, 2015. (ISSTA 2015), p. 223–233. ISBN 978-1-4503-3620-8. Disponível em: <http://doi.acm.org/10.1145/2771783.2771793>.

HARROLD, M. J.; ROTHERMEL, G. Perfoming data flow testing on classes. In: *II ACM SIGSOFT Symposium on Foundations of Software Engineering*. [S.l.: s.n.], 1994. p. 154–163.

HENDERSON, D. L.; CARTER, S. *Qualitative Research Design*. London: Sage Publications Ltd, 2009. ISBN 978-1-4129-2345-3.

HERNANDES, E. et al. Using GQM and TAM to evaluate StArt - a tool that supports Systematic Review. *CLEI Electronic Journal*, scielouy, v. 15, p. 3 – 3, 04 2012. ISSN 0717-5000. Disponível em: <http://www.scielo.edu.uy/scielo.php?script=sci_arttext&pid=S0717-50002012000100003&nrm=iso>.

HOLLAN, J.; HUTCHINS, E.; KIRSH, D. Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction (TOCHI)*, ACM, v. 7, n. 2, p. 174–196, 2000.

HOWDEN, W. E. *Software Engineering and Technology: Functional Program Testing and Analysis*. New York: McGrall-Hill Book Co, 1987.

HSIEH, H.-F.; SHANNON, S. E. Three approaches to qualitative content analysis. *Qualitative Health Research*, v. 15, n. 9, p. 1277–1288, nov. 2005. ISSN 1049-7323.

HUTCHINS, E. *Cognition in the Wild*. Revised ed. edition. Cambridge, Mass: A Bradford Book, 1996. ISBN 978-0-262-58146-2.

HUTCHINSON, H. et al. Technology probes: Inspiring design for and with families. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2003. (CHI '03), p. 17–24. ISBN 1-58113-630-7. Disponível em: <http://doi.acm.org/10.1145/642611.642616>.

IEEE. *IEEE Standard Glossary of Software Engineering Terminology*. [S.l.], 1990.

IEEE. Systems and software engineering – vocabulary. *ISO/IEC/IEEE 24765:2010(E)*, p. 1–418, Dec 2010.

INOZEMTSEVA, L.; HOLMES, R. Coverage is not strongly correlated with test suite effectiveness. In: *Proceedings of the 36th International Conference on Software Engineering*. New York, NY, USA: ACM, 2014. (ICSE 2014), p. 435–445. ISBN 978-1-4503-2756-5. Disponível em: <http://doi.acm.org/10.1145/2568225.2568271>.

JIA, Y.; HARMAN, M. An Analysis and Survey of the Development of Mutation Testing. *IEEE Transactions on Software Engineering*, v. 37, n. 5, p. 649–678, set. 2011. ISSN 0098-5589.

JOHNSON-LAIRD, P. N. Mental models and deduction. *Trends in Cognitive Sciences*, v. 5, n. 10, p. 434–442, out. 2001. ISSN 1364-6613, 1879-307X. Disponível em: <http://www.cell.com/trends/cognitive-sciences/abstract/S1364-6613(00)01751-4>.

JONES, J.; HARROLD, M.; STASKO, J. Visualization of test information to assist fault localization. In: *Proceedings of the 24rd International Conference on Software Engineering, 2002. ICSE 2002*. [S.l.: s.n.], 2002. p. 467–477.

JR, R. M. G.; UNRAU, Y. A. Social Work Research and Evaluation: Foundations of Evidence-Based Practice. In: ____. [S.l.]: Oxford University Press, 2010. cap. 4 - The Qualitative Research Approach, p. 57. ISBN 978-0-19-988989-1. Google-Books-ID: sYC_WI_COd8C.

KAMARUZAMAN, M. F. et al. Developing User Interface Design Application for Children with Autism. *Procedia - Social and Behavioral Sciences*, v. 217, n. Supplement C, p. 887 – 894, 2016. ISSN 1877-0428. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877042816000471>.

KARAM, M. R.; ABDALLAH, A. A. Assisting in fault localization using visual programming constructs. In: *Canadian Conference on Electrical and Computer Engineering, 2005.* [S.l.: s.n.], 2005. p. 856–860. ISSN 0840-7789.

KASURINEN, J.; TAIPALE, O.; SMOLANDER, K. Analysis of Problems in Testing Practices. In: *Software Engineering Conference, 2009. APSEC '09. Asia-Pacific*. [S.l.: s.n.], 2009. p. 309–315.

KASURINEN, J.; TAIPALE, O.; SMOLANDER, K. Software Test Automation in Practice: Empirical Observations. *Adv. Soft. Eng.*, v. 2010, p. 4:1–4:13, jan. 2010. ISSN 1687-8655. Disponível em: <http://dx.doi.org/10.1155/2010/620836>.

KIEWRA, K. A. Notetaking and review: The research and its implications. *Instructional Science*, v. 16, n. 3, p. 233–249, Sep 1987. ISSN 1573-1952. Disponível em: <https://doi.org/10.1007/BF00120252>.

KNOWLES, E. S.; NATHAN, K. T. Acquiescent Responding in Self-Reports: Cognitive Style or Social Concern? *Journal of Research in Personality*, v. 31, n. 2, p. 293–301, jun. 1997. ISSN 0092-6566. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0092656697921802>.

KOSSLYN, S. M. Mental imagery. In: OSHERSON, D. N.; KOSSLYN, S. M.; HOLLERBACH, J. M. (Ed.). *Visual cognition and action: Vol. 2.* [S.l.]: MIT Press, 1990. cap. 2, p. 73–97.

LAM, H. et al. Empirical studies in information visualization: Seven scenarios. *IEEE Transactions on Visualization and Computer Graphics*, v. 18, n. 9, p. 1520–1536, set. 2012. ISSN 1077-2626.

LANZA, M.; DUCASSE, S. Polymetric views - a lightweight visual approach to reverse engineering. *IEEE Transactions on Software Engineering*, v. 29, n. 9, p. 782–795, Sept 2003. ISSN 0098-5589.

LAPPALAINEN, V. et al. ComTest: A Tool to Impart TDD and Unit Testing to Introductory Level Programming. In: *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: ACM, 2010. (ITiCSE '10), p. 63–67. ISBN 978-1-60558-820-9. Disponível em: <http://doi.acm.org/10.1145/1822090.1822110>.

LAWRENCE, J. et al. How well do professional developers test with code coverage visualizations? an empirical study. In: *2005 IEEE Symposium on Visual Languages and Human-Centric Computing*. [S.l.: s.n.], 2005. p. 53–60.

LAWRENCE, J. et al. How well do professional developers test with code coverage visualizations? an empirical study. In: *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*. [S.l.: s.n.], 2005. p. 53–60. ISSN 1943-6092.

LEE, J.; KANG, S.; LEE, D. Survey on software testing practices. *IET Software*, v. 6, n. 3, p. 275–282, jun. 2012. ISSN 1751-8806.

LEITNER, A. et al. Reconciling Manual and Automated Testing: The AutoTest Experience. In: *40th Annual Hawaii International Conference on System Sciences, 2007. HICSS 2007*. [S.l.: s.n.], 2007. p. 261a–261a.

LI, J.; HORGAN, J. A scalable tool for efficient protocol validation and testing. *Computer Communications*, v. 26, n. 7, p. 766–773, 2003. ISSN 0140-3664. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0140366402002116>.

LIENHARD, A. et al. Test blueprints - exposing side effects in execution traces to support writing unit tests. In: *12th European Conference on Software Maintenance and Reengineering, 2008. CSMR 2008*. [S.l.: s.n.], 2008. p. 83–92.

LUO, Q. et al. An Empirical Analysis of Flaky Tests. In: *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. New York, NY, USA: ACM, 2014. (FSE 2014), p. 643–653. ISBN 978-1-4503-3056-5. Disponível em: <http://doi.acm.org/10.1145/2635868.2635920>.

LUZZARDI, P. R. et al. An extended set of ergonomic criteria for information visualization techniques. In: *Proceedings of the Seventh IASTED International Conference on Computer Graphics And Imaging (Cgim-2004), Kauai*. [S.l.: s.n.], 2004. p. 236–241.

MACKAY, W. E. *Users and customizable software : a co-adaptive phenomenon*. Tese (Thesis) — Massachusetts Institute of Technology, 1990. Disponível em: <http://dspace.mit.edu/handle/1721.1/14087>.

MACKINLAY, J. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, ACM, New York, NY, USA, v. 5, n. 2, p. 110–141, abr. 1986. ISSN 0730-0301. Disponível em: <http://doi.acm.org/10.1145/22949.22950>.

MAKANY, T.; KEMP, J.; DROR, I. E. Optimising the use of note-taking as an external cognitive aid for increasing learning. *British Journal of Educational Technology*, Wiley Online Library, v. 40, n. 4, p. 619–635, 2009.

MALDONADO, J. C. *Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software*. Tese (Tese de Doutoramento) — DCA/FEE/UNICAMP, Campinas, SP, 1991.

MAO, C.; LU, Y. CppTest: A prototype tool for testing c/c++ programs. In: *The Second International Conference on Availability, Reliability and Security, 2007. ARES 2007*. [S.l.: s.n.], 2007. p. 1066–1073.

MATHUR, A. P. On the relative strengths of data flow and mutation testing. In: *Ninth Annual Pacific Northwest Software Quality Conference*. Portland, OR: [s.n.], 1991. p. 165–181.

MATHUR, R.; MILES, S.; DU, M. Adaptive automation: Leveraging machine learning to support uninterrupted automated testing of software applications. *arXiv preprint arXiv:1508.00671*, 2015.

MAYRING, P. Qualitative content analysis: theoretical foundation, basic procedures and software solution. In: ____. [S.l.]: psychopen.eu, 2014. cap. 6, p. 143.

MEADE, E. *Green Bar Addiction*. 2009. Disponível em: <http://wiki.c2.com/?GreenBarAddiction>.

MERRIAM, S. B. *Qualitative Research in Practice: Examples for Discussion and Analysis*. 1 edition. ed. San Francisco: Jossey-Bass, 2002. ISBN 978-0-7879-5895-4.

MERRIëNBOER, J. J. G. van; SWELLER, J. Cognitive Load Theory and Complex Learning: Recent Developments and Future Directions. *Educational Psychology Review*, v. 17, n. 2, p. 147–177, jun. 2005. ISSN 1573-336X. Disponível em: <https://doi.org/10.1007/s10648-005-3951-0>.

MICCO, J. *Flaky Tests at Google and How We Mitigate Them*. 2016. Disponível em: <https://testing.googleblog.com/2016/05/flaky-tests-at-google-and-how-we.html>.

MILES, M. B.; HUBERMAN, A. M.; SALDANA, J. Qualitative Data Analysis: A Methods Sourcebook. In: ____. 3 edition. ed. Thousand Oaks, Californinia: SAGE Publications, Inc, 2013. cap. 2, p. 41–43. ISBN 978-1-4522-5787-7.

MUTO, Y.; OKANO, K.; KUSUMOTO, S. Improvement of a visualization technique for the passage rate of unit testing and static checking and its evaluation. In: *Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA)*. [S.l.: s.n.], 2011. p. 279–284.

MUTO, Y.; OKANO, K.; KUSUMOTO, S. A visualization technique for the passage rates of unit testing and static checking with caller-callee relationships. In: *Parallel and Distributed Processing with Applications Workshops (ISPAW), 2011 Ninth IEEE International Symposium on*. [S.l.: s.n.], 2011. p. 336–341.

MYERS, G. J. A controlled experiment in program testing and code walkthroughs/isnpections. *Communications of the ACM*, v. 21, n. 9, p. 760–768, 1978.

MYERS, G. J. et al. *The Art of Software Testing*. [S.l.]: John Wiley & Sons, Inc., Hoboken, New Jersey, 2004.

MYNATT, E. D. Transforming graphical interfaces into auditory interfaces for blind users. *Human–Computer Interaction*, Taylor & Francis, v. 12, n. 1-2, p. 7–45, 1997.

NEISSER, U. *Cognitive Psychology*. First edition. New York, NY: Appleton-Century 1967, 1967. ISBN 9780390665096.

NG, S. et al. A preliminary survey on software testing practices in Australia. In: *Software Engineering Conference, 2004. Proceedings. Australian*. [S.l.: s.n.], 2004. p. 116–125.

NORMAN, D. Emotion & Design: Attractive Things Work Better. *interactions*, v. 9, n. 4, p. 36–42, jul. 2002. ISSN 1072-5520. Disponível em: <http://doi.acm.org/10.1145/543434.543435>.

NORMAN, D. *The Design of Everyday Things: Revised and Expanded Edition*. Revised. New York, New York: Basic Books, 2013. ISBN 9780465050659.

NORMAN, D. *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*. [S.l.]: Diversion Books, 2014. Google-Books-ID: yPKkBQAAQBAJ. ISBN 978-1-62681-537-7.

NORMAN, D. A. *Things That Make Us Smart: Defending Human Attributes In The Age Of The Machine*. Reprint edition. Reading, Mass: Basic Books, 1994. ISBN 9780201626957.

OFFUTT, A. J. et al. An experimental determination of sufficient mutant operators. *ACM Transactions on Software Engineering and Methodology*, v. 5, n. 2, p. 99–118, abr. 1996.

OFFUTT, A. J.; ROTHERMEL, G.; ZAPF, C. An Experimental Evaluation of Selective Mutation. In: *Proceedings of the 15th International Conference on Software Engineering*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1993. (ICSE '93), p. 100–107. ISBN 978-0-89791-588-5. Disponível em: <http://dl.acm.org/citation.cfm?id=257572.257597>.

OFFUTT, A. J.; ROTHERMEL, G.; ZAPF, C. An experimental evaluation of selective mutation. In: *15th International Conference on Software Engineering*. Baltimore, MD: IEEE Computer Society Press, 1993. p. 100–107.

ORSO, A.; ROTHERMEL, G. Software Testing: A Research Travelogue. In: *Proceedings of the on Future of Software Engineering*. New York, NY, USA: ACM, 2014. (FOSE 2014), p. 117–132. ISBN 978-1-4503-2865-4. Disponível em: <http://doi.acm.org/10.1145/2593882.2593885>.

OSTERWEIL, L. J. et al. Determining the impact of software engineering research on practice. *Computer*, v. 41, n. 3, p. 39–49, March 2008. ISSN 0018-9162.

OVIATT, S. Human-centered design meets cognitive load theory: Designing interfaces that help people think. In: *Proceedings of the 14th ACM International Conference on Multimedia*. New York, NY, USA: ACM, 2006. (MM '06), p. 871–880. ISBN 1-59593-447-2. Disponível em: <http://doi.acm.org/10.1145/1180639.1180831>.

OVIATT, S. et al. Expressive pen-based interfaces for math education. In: *Proceedings of the 8th Iternational Conference on Computer Supported Collaborative Learning*. International Society of the Learning Sciences, 2007. (CSCL'07), p. 573–582. ISBN 978-0-6151-5436-7. Disponível em: <http://dl.acm.org/citation.cfm?id=1599600.1599708>.

OVIATT, S.; ARTHUR, A.; COHEN, J. Quiet interfaces that help students think. In: *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA: ACM, 2006. (UIST '06), p. 191–200. ISBN 1-59593-313-1. Disponível em: <http://doi.acm.org/10.1145/1166253.1166284>.

OVIATT, S. et al. The impact of interface affordances on human ideation, problem solving, and inferential reasoning. *ACM Trans. Comput.-Hum. Interact.*, ACM, New York, NY, USA, v. 19, n. 3, p. 22:1–22:30, out. 2012. ISSN 1073-0516. Disponível em: <http://doi.acm.org/10.1145/2362364.2362370>.

PAAS, F. et al. Cognitive load measurement as a means to advance cognitive load theory. *Educational psychologist*, Taylor & Francis, v. 38, n. 1, p. 63–71, 2003.

PERNICE, K.; NIELSEN, J. *How to Conduct Eyetracking Studies | Nielsen Norman Group Report*. Disponível em: <https://www.nngroup.com/reports/how-to-conduct-eyetracking-studies/>.

PERRY, D. E.; KAISER, G. E. Adequate testing and object-oriented programming. *J. Object Oriented Program.*, SIGS Publications, Denville, NJ, USA, v. 2, n. 5, p. 13–19, 1990. ISSN 0896-8438.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: *12th International Conference on Evaluation and Assessment in Software Engineering*. [S.l.: s.n.], 2008. v. 17, p. 1.

PIROLLI, P.; CARD, S. Information foraging. *Psychological Review*, v. 106, n. 4, p. 643–675, 1999. ISSN 1939-1471(Electronic),0033-295X(Print).

PRADO, M. *Resources_TCSUT*. 2017. Disponível em: <https://app.box.com/s/sk52eu5akh1f0o8dj0qkh4pvp5r2p6lk>.

PRADO, M. P. et al. Wap: Cognitive aspects in unit testing: The hunting game and the hunter's perspective. In: *Software Reliability Engineering (ISSRE), 2015 IEEE 26th International Symposium on*. [S.l.: s.n.], 2015. p. 387–392.

PRADO, M. P.; VINCENZI, A. M. R. Advances in the characterization of cognitive support for unit testing: The bug-hunting game and the visualization arsenal. In: *2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. [S.l.: s.n.], 2016. p. 213–220.

PRADO, M. P.; VINCENZI, A. M. R. *Link to the survey applied with the volunteers*. 2017. [Online] 2017, <http://survey.sogosurvey.com/k/RQsTTWSQsRTsPsPsP>. (Accessed: 11-March-2017).

PRADO, M. P.; VINCENZI, A. M. R. Towards cognitive support for unit testing: a qualitative study with practitioners. *Journal of Systems and Software*, 2018. ISSN 0164-1212. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0164121218300529>.

PRADO, M. P.; VINCENZI, A. M. R.; NASCIMENTO, H. A. D. D. *Artefatos do Mapeamento Sistemático de Ferramentas e Técnicas de visualização para apoio ao teste de software* . 2014. [Online] 2014, <http://testvis-site-mapping.193b.starter-ca-central-1.openshiftapps.com/crbst_2.html>. Systematic Mapping Artifacts – (Accessed: 10-October-2014).

PRADO, M. P. et al. Characterization of techniques and tools of visualization applied to software comprehension: A systematic mapping. In: *ICSEA 2013, The Eighth International Conference on Software Engineering Advances*. [s.n.], 2013. p. 297–303. ISBN 978-1-61208-304-9. Disponível em: <http://www.thinkmind.org/index.php?view=article&articleid=icsea_2013_10_20_10403>.

PREECE, J.; SHARP, H.; ROGERS, Y. *Interaction Design: Beyond Human-Computer Interaction*. 4 edition. ed. Chichester: Wiley, 2015. ISBN 978-1-119-02075-2.

PRESSMAN, R. S.; MAXIM, B. *Software Engineering: A Practitioner's Approach*. 8th revised edition edition. ed. New York, NY: McGraw-Hill Science/Engineering/Math, 2014. ISBN 9780078022128.

RAMLER, R.; CZECH, G.; SCHLOSSER, D. Unit Testing beyond a Bar in Green and Red. In: *Extreme Programming and Agile Processes in Software Engineering*. Springer, Berlin, Heidelberg, 2003. (Lecture Notes in Computer Science), p. 319–321. ISBN 978-3-540-40215-2 978-3-540-44870-9. Disponível em: <https://link.springer.com/chapter/10.1007/3-540-44870-5_39>.

RAPPS, S.; WEYUKER, E. J. Data flow analysis techniques for program test data selection. In: *6th International Conference on Software Engineering*. Tokio, Japan: [s.n.], 1982. p. 272–278.

RAPPS, S.; WEYUKER, E. J. Selecting software test data using data flow information. *IEEE Transactions on Software Engineering*, v. 11, n. 4, p. 367–375, abr. 1985.

RITCHIE, J.; LEWIS, J. Qualitative research practice: A guide for social science students and researchers. In: ____. 1 edition. ed. Los Angeles, Calif.: SAGE Publications Ltd, 2003. cap. 10 - Generalizing from Qualitative Research, p. 264. ISBN 978-0-7619-7110-8.

ROBERTS, J. C. Display models for visualization. In: *Information Visualization, 1999. Proceedings. 1999 IEEE International Conference on*. [S.l.]: IEEE, 1999. p. 200–206.

ROBERTSON, P. K.; FERRARI, L. D. Systematic approaches to visualization: is a reference model needed. *Scientific Visualization*, p. 287–305, 1994.

ROMERO, J. D. et al. ITVT: An image testing and visualization tool for image processing tasks. In: *5th Iberian Conference on Information Systems and Technologies*. [S.l.: s.n.], 2010. p. 1–6. ISSN 2166-0727.

RUNESON, P. A survey of unit testing practices. *IEEE Software*, v. 23, n. 4, p. 22–29, jul. 2006. ISSN 0740-7459.

SCAIFE, M.; ROGERS, Y. External Cognition: How Do Graphical Representations Work? *Int. J. Hum.-Comput. Stud.*, v. 45, n. 2, p. 185–213, ago. 1996. ISSN 1071-5819. Disponível em: <http://dx.doi.org/10.1006/ijhc.1996.0048>.

SCHAEKEN, W.; JOHNSON-LAIRD, P. N.; D'YDEWALLE, G. Mental models and temporal reasoning. *Cognition*, v. 60, n. 3, p. 205–234, set. 1996. ISSN 0010-0277.

SCHREIER, M. The SAGE Handbook of Qualitative Data Analysis. In: ____. 1 edition. ed. Los Angeles: SAGE Publications Ltd, 2013. v. 1, cap. Qualitative Content Analysis, p. 175. ISBN 978-1-4462-0898-4.

SIMON, H. A. Rational choice and the structure of the environment. *Psychological Review*, v. 63, n. 2, p. 129–138, 1956. ISSN 1939-1471(Electronic);0033-295X(Print).

SIMON, H. A. et al. *Research Briefings 1986*. [S.l.]: National Academies Press, 1986. ISBN 9780309036894.

SIMãO, A. S.; MALDONADO, J. C. Mutation based test sequence generation for Petri nets. In: *Proceedings of III Workshop of Formal Methods*. João Pessoa, PB: [s.n.], 2000. p. 68–79.

SOUZA, S. R. S. *Avaliação do Custo e Eficácia do Critério Análise de Mutantes na Atividade de Teste de Software*. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC/USP), São Carlos, SP, junho 1996.

SPENCE, R. *Information Visualization: Design for Interaction*. [S.l.]: Pearson/Prentice Hall, 2007. ISBN 9780132065504.

STOL, K.-J.; RALPH, P.; FITZGERALD, B. Grounded Theory in Software Engineering Research: A Critical Review and Guidelines. In: *Proceedings of the 38th International Conference on Software Engineering*. New York, NY, USA: ACM, 2016. (ICSE '16), p. 120–131. ISBN 978-1-4503-3900-1. Disponível em: <http://doi.acm.org/10.1145/2884781.2884833>.

STONE, D. et al. *User Interface Design and Evaluation*. [S.l.]: Morgan Kaufmann, 2005.

SWELLER, J. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, Lawrence Erlbaum Associates, Inc., v. 12, n. 2, p. 257–285, 1988. ISSN 1551-6709. Disponível em: <http://dx.doi.org/10.1207/s15516709cog1202_4>.

SWELLER, J. *Instructional design in technical areas*. Camberwell, Vic.: ACER Press, 1999. ISBN 0-86431-312-8 978-0-86431-312-6.

The Chisel Group. *Publications*. Victoria, B.C. Canada: [s.n.], 2015. Disponível em: <http://thechiselgroup.org/publications/>.

TILLMANN, N.; HALLEUX, J. de; XIE, T. Transferring an automated test generation tool to practice: from pex to fakes and code digger. In: *ASE*. [S.l.: s.n.], 2014.

TVERSKY, A. Elimination by aspects: A theory of choice. *Psychological Review*, v. 79, n. 4, p. 281–299, 1972. ISSN 1939-1471(Electronic);0033-295X(Print).

VASALOU, A. et al. Human-moderated remote user testing: Protocols and applications. In: *8th ERCIM Workshop, User Interfaces for All, Wien, Austria*. [S.l.: s.n.], 2004. p. 1–10.

VINCENZI, A. M. R. *Subsídios para o Estabelecimento de Estratégias de Teste Baseadas na Técnica de Mutação*. Dissertação (Mestrado) — ICMC/USP, São Carlos, SP, nov. 1998.

VINCENZI, A. M. R. *Orientação a Objeto: Definição e Análise de Recursos de Teste e Validação*. Tese (Doutorado) — ICMC/USP, São Carlos, SP, maio 2004.

Virginia Tech. *JUnit Tips: What is JUnit?* 2013. Disponível em: <http://web-cat.org/junit-quickstart/junit.html>.

WALENSTEIN, A. *Cognitive Support in Software Engineering Tools: A Distributed Cognition Framework*. 402 p. Tese (Doutorado) — School of Computing Science, Simon Fraser University, maio 2002.

WANG, H.; ZHANG, X.; ZHOU, M. MaVis: Feature-based defects visualization in software testing. In: *2012 Spring Congress on Engineering and Technology (S-CET)*. [S.l.: s.n.], 2012. p. 1–4.

WEBER, R. P. Basic content analysis. In: ____. 1 edition. ed. Newbury Park, Calif: SAGE Publications, Inc, 1990. cap. 2, p. 37. ISBN 978-0-8039-3863-2.

WEYUKER, E. J. On testing non-testable programs. *Computer Journal*, v. 25, n. 4, p. 465–470, nov. 1982.

WEYUKER, E. J. The cost of data flow testing: an empirical study. *IEEE Transactions on Software Engineering*, v. 16, n. 2, p. 121–128, fev. 1990.

WIKLUND, K. et al. Impediments for automated testing – an empirical analysis of a user support discussion board. In: *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation (ICST)*. [S.l.: s.n.], 2014. p. 113–122.

WOHLIN, C. *Experimentation in software engineering*. Berlin; New York: Springer, 2012. ISBN 9783642290442 3642290442.

WONG, W. E. et al. A comparison of selective mutation in C and Fortran. In: *Anais do Workshop do Projeto Validação e Teste de Sistemas de Operação*. Aguas de Lindoia, SP: [s.n.], 1997. p. 71–84.

XIE, T. Transferring software testing tools to practice. In: *2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST)*. [S.l.: s.n.], 2017. p. 8–8.

ZENG, X. et al. Automated test input generation for android: Are we really there yet in an industrial case? In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. New York, NY, USA: ACM, 2016. (FSE 2016), p. 987–992. ISBN 978-1-4503-4218-6. Disponível em: <http://doi.acm.org/10.1145/2950290.2983958>.

ZHU, H.; HALL, P.; MAY, J. Software unit test coverage and adequacy. *ACM Computing Surveys*, v. 29, n. 4, p. 366–427, dez. 1997.

APÊNDICE A

Parecer consubstanciado do Comitê de Ética em Pesquisa

Neste apêndice é apresentado o parecer consubstanciado aprovado pelo Comitê de Ética em Pesquisa da Universidade Federal de Goiás (CEP-UFG), para aplicação do *survey* descrito no capítulo 7.

UNIVERSIDADE FEDERAL DE GOIÁS - UFG

PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: Survey sobre práticas de teste unitário e identificação de flaky tests.
Survey on unit testing practices and flakiness identification.

Pesquisador: MARLLOS PAIVA PRADO

Área Temática:

Versão: 1

CAAE: 60732216.9.0000.5083

Instituição Proponente: Instituto de Informática

Patrocinador Principal: Financiamento Próprio

DADOS DO PARECER

Número do Parecer: 1.803.311

Apresentação do Projeto:

Título da Pesquisa: Survey sobre práticas de teste unitário e identificação de flaky tests. Survey on unit testing practices and flakiness identification. Pesquisador Responsável: MARLLOS PAIVA PRADO. N. CAAE: 60732216.9.0000.5083. Instituição Proponente: Instituto de Informática. O survey a ser aplicado nesse projeto de pesquisa tem como principal finalidade a realização de uma análise qualitativa das respostas textuais fornecidas por profissionais de desenvolvimento de software para entendimento das práticas comuns e atuais de revisão de teste de software unitário, com enfoque na identificação de flaky tests (casos de teste não robustos). As categorias e subcategorias geradas a partir da análise qualitativa das respostas deverão viabilizar a proposição de um framework de requisitos funcionais e não funcionais que devem ser atendidos pelas ferramentas de teste de software unitário para facilitar a atividade de revisão de caso de teste unitário e identificação de flaky tests. Esses requisitos são denominados de: "tarefas que requerem suporte cognitivo das ferramentas de teste para apoio à atividade de revisão de teste unitário".

Objetivo da Pesquisa:

Objetivo Primário:

O objetivo primário da pesquisa é elicitar tarefas de teste unitário que demandem suporte

Endereço: Prédio da Reitoria Térreo Cx. Postal 131
Bairro: Campus Samambaia **CEP:** 74.001-970
UF: GO **Município:** GOIANIA
Telefone: (62)3521-1215 **Fax:** (62)3521-1163 **E-mail:** cep.prpi.ufg@gmail.com

UNIVERSIDADE FEDERAL DE GOIÁS - UFG

Continuação do Parecer: 1.803.311

cognitivo da ferramenta para revisão de casos de teste. Desta maneira, a questão de pesquisa principal que queremos responder com a análise desse survey é:RQ1. "Quais são as tarefas referentes à revisão de casos de teste unitário que demandam suporte cognitivo das ferramentas para facilitar a realização dessa atividade, a partir da perspectiva dos profissionais?"

Objetivo Secundário:

Um objetivo secundário da pesquisa consiste em verificar de que forma as tarefas identificadas no objetivo primário atendem ou diferenciam-se das necessidades levantadas previamente por meio de revisão da literatura. Desta maneira, a questão de pesquisa secundária que queremos responder com a análise desse survey é:RQ2. "Como as tarefas que requerem suporte cognitivo das ferramentas, levantadas no presente estudo, atendem ou

diferenciam-se das dimensões do framework inicial proposto em Prado et al., 2015 com base na revisão da literatura?"

Avaliação dos Riscos e Benefícios:

Relatam que os seguintes riscos na condução desse estudo: os pesquisadores informarão tanto no convite para participação do survey quanto na página inicial do survey o propósito da pesquisa, seu caráter anônimo e confidencial. Além disso, o formato digital do survey e a veiculação pela web buscam justamente dar liberdade para que o participante sinta-se livre para aceitar, rejeitar ou ignorar a participação no estudo, uma vez que o participante não terá contato pessoal com os pesquisadores. Estará explicito ainda para o participante uma declaração de que o survey é puramente acadêmico, que os resultados serão tornados anônimos antes de serem publicados e que a desistência de participação no estudo é facultada ao participante a qualquer momento sem qualquer consequência negativa. Disponibilizam contato com os pesquisadores.

Benefícios: (i) a definição de um framework capaz de delinear problemas da prática que têm sido negligenciados pela pesquisa em teste de software até o momento; (ii) uma vez definido o framework, a possibilidade de começar a propor evoluções nas ferramentas que contribuam com a diminuição do esforço empregado pelos profissionais em tarefas não atendidas pelas ferramentas de teste atuais; (iii) possibilidade de fomentar novos estudos semelhantes na área, levando-se em consideração outros tipos de teste de software (teste de integração ou de sistema por exemplo); (iv) os profissionais podem se sentir motivados a continuar contribuindo futuramente com a pesquisa na área de teste, considerando-se a carência de pesquisas atuais na área que levam em consideração a opinião e percepção dos mesmos

na evolução das tecnologias.

Endereço: Prédio da Reitoria Térreo Cx. Postal 131
Bairro: Campus Samambaia **CEP:** 74.001-970
UF: GO **Município:** GOIANIA
Telefone: (62)3521-1215 **Fax:** (62)3521-1163 **E-mail:** cep.prpi.ufg@gmail.com

UNIVERSIDADE FEDERAL DE GOIÁS - UFG

Continuação do Parecer: 1.803.311

Comentários e Considerações sobre a Pesquisa:

O instrumento de coleta de dados será enviado via internet. As questões abertas do survey serão analisadas, seguindo os princípios descrito em Mayring, 2014. A unidade de análise para codificação será cada resposta de questão obtida com a aplicação do survey. O processo de derivação de categorias e subcategorias será predominante indutivo e tem como finalidade entender um fenômeno ainda pouco estudado - no caso, necessidades de suporte cognitivo para melhoria de ferramentas que apoiem a revisão de casos de teste unitário. A parte dedutiva será aplicada no final da análise das respostas e terá como objetivo apenas verificar se as categorias e subcategorias que emergiram são compatíveis ou não com as necessidades levantadas previamente por meio de revisão bibliográfica (dimensões do framework proposto em Prado et al. 2015). As questões fechadas (objetivas) serão analisadas por meio de estatística descritiva. Os resultados da análise das questões fechadas não têm a finalidade de generalizar resultados para a população geral mas sim: (i) viabilizar a caracterização da amostra populacional que se voluntariar a participar do survey. Tal caracterização pretende contribuir com desdobramentos futuros do survey -- i.e. permitindo determinar o perfil geral dos respondentes da pesquisa e viabilizar a comparação entre novos resultados com os resultados obtidos nessa pesquisa (transferability); (ii) melhorar o entendimento a respeito das respostas fornecidas nas questões abertas, enriquecendo as conclusões sobre os resultados obtidos na análise qualitativa.

Coleta de dados a partir de dezembro de 2016.

Considerações sobre os Termos de apresentação obrigatória:

Folha de rosto devidamente assinada.

Anuência do local de Coleta de dados.

Coleta de dados a partir de dezembro de 2016.

Termo de compromisso dos pesquisadores envolvidos.

Instrumento de coleta de dados.

No TCLE asseguram o sigilo da identificação do participante, o direito à indenização e o direito de se retirar ou não responder em caso de constrangimento.

Conclusões ou Pendências e Lista de Inadequações:

Após análise dos documentos postados somos favoráveis à aprovação do presente protocolo de pesquisa, smj deste comitê.

Considerações Finais a critério do CEP:

Endereço: Prédio da Reitoria Térreo Cx. Postal 131
Bairro: Campus Samambaia **CEP:** 74.001-970
UF: GO **Município:** GOIANIA
Telefone: (62)3521-1215 **Fax:** (62)3521-1163 **E-mail:** cep.prpi.ufg@gmail.com

UNIVERSIDADE FEDERAL DE GOIÁS - UFG

Continuação do Parecer: 1.803.311

Informamos que o Comitê de Ética em Pesquisa/CEP-UFG considera o presente protocolo APROVADO, o mesmo foi considerado em acordo com os princípios éticos vigentes. Reiteramos a importância deste Parecer Consubstanciado, e lembramos que o(a) pesquisador(a) responsável deverá encaminhar ao CEP-UFG o Relatório Final baseado na conclusão do estudo e na incidência de publicações decorrentes deste, de acordo com o disposto na Resolução CNS n. 466/12. O prazo para entrega do Relatório é de até 30 dias após o encerramento da pesquisa, prevista para maio de 2017.

Este parecer foi elaborado baseado nos documentos abaixo relacionados:

Tipo Documento	Arquivo	Postagem	Autor	Situação
Informações Básicas do Projeto	PB_INFORMAÇÕES_BÁSICAS_DO_PROJETO_777973.pdf	06/10/2016 09:56:55		Aceito
Declaração de Instituição e Infraestrutura	Anuencia_assinada.pdf	06/10/2016 09:54:36	MARLLOS PAIVA PRADO	Aceito
Projeto Detalhado / Brochura Investigador	projeto_cronograma.pdf	06/10/2016 09:51:27	MARLLOS PAIVA PRADO	Aceito
TCLE / Termos de Assentimento / Justificativa de Ausência	Modelo_de_TCLE.pdf	06/10/2016 09:49:04	MARLLOS PAIVA PRADO	Aceito
Declaração de Pesquisadores	termo_compromisso_assinado.pdf	06/10/2016 09:46:24	MARLLOS PAIVA PRADO	Aceito
Folha de Rosto	folha_rosto.pdf	03/10/2016 15:00:07	MARLLOS PAIVA PRADO	Aceito
Outros	questoes_marllosprado.pdf	27/09/2016 11:34:23	MARLLOS PAIVA PRADO	Aceito

Situação do Parecer:
Aprovado

Necessita Apreciação da CONEP:
Não

Endereço: Prédio da Reitoria Térreo Cx. Postal 131
Bairro: Campus Samambaia **CEP:** 74.001-970
UF: GO **Município:** GOIANIA
Telefone: (62)3521-1215 **Fax:** (62)3521-1163 **E-mail:** cep.prpi.ufg@gmail.com

UNIVERSIDADE FEDERAL DE GOIÁS - UFG

Continuação do Parecer: 1.803.311

GOIANIA, 03 de Novembro de 2016

Assinado por:
João Batista de Souza
(Coordenador)

Endereço: Prédio da Reitoria Térreo Cx. Postal 131
Bairro: Campus Samambaia **CEP:** 74.001-970
UF: GO **Município:** GOIANIA
Telefone: (62)3521-1215 **Fax:** (62)3521-1163 **E-mail:** cep.prpi.ufg@gmail.com

APÊNDICE B

Questões do Survey

Neste apêndice é apresentada uma cópia das questões do *survey* descrito no capítulo 7. Note-se que o documento contendo as questões foi gerado automaticamente pela plataforma na qual o *survey* foi escrito. Consequentemente, a formatação visual e a numeração das questões sofreram alterações de formatação com relação à versão apresentada aos voluntários no formato online. Para verificar as questões no formato online, acesse o link disponível em Prado e Vincenzi (2017).

Survey On Unit Testing Practices and Flakiness identification

IMPORTANT NOTE: Do not use the "back/forward" button of your browser while answering the survey. To navigate in the survey pages, please, use the navigation buttons located in the bottom of each page.

---

page 1

Dear Participant, If you have experience with unit testing, your opinion is valuable in our research. We want to better understand cognitive issues related to unit testing and it is important that this information comes from actual practitioners. We are Marllos Prado and Auri Marcelo Rizzo Vincenzi, researchers from Universidade Federal de Goiás (UFG) and Universidade Federal de São Carlos (UFSCAR) respectively. Our research aims to better understand cognitive support for unit testing, in particular regarding flaky test cases. This means that our main focus is in unit test cases that fail, not due to an error in the tested code, but due to an error in the test case itself. We'd be grateful if you could help us understand how you deal with this issue in daily practice, and how tools can play a role in dealing with flaky testing. The survey should take about 20 minutes. This is a purely academic research project with no commercial interests. We will openly publish the results so everyone can benefit from them, but will anonymize everything before doing so; your responses will be handled confidentially. Please note that you are not obligated to participate in the survey. If at some point during the survey you want to stop, you are free to do so without any negative consequences. Survey data not submitted will not be registered and used.

Thank you for your collaboration!

BY CLICKING ON THE BUTTON "Next >>", YOU INDICATE THAT YOU READ, UNDERSTOOD AND AGREE WITH THE TERMS AND CONDITIONS OF THIS SURVEY:

page 2

1. 1. What is your age? (Select one option)

- ○ Under 18 years
- ○ 18 to 24 years
- ○ 25 to 34 years
- ○ 35 to 44 years
- ○ 45 to 54 years
- ○ 55 to 64 years
- ○ Age 65 or older

2. 2. How many years of experience do you have with unit testing? (Select one option)

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5
- ○ 6
- ○ 7
- ○ 8
- ○ 9
- ○ 10 or more

3. 3. What is your gender (Select one option)

- ○ Female
- ○ Male

4. 4. What do you usually classify as a unit to be tested? (Select one option)

- ○ A single method implemented in the code
- ○ A single class implemented in the code
- ○ A set of two or more methods defining a unit in a higher level of abstraction
- ○ A set of two or more classes defining a unit in a higher level of abstraction

5. 5. How motivated do you feel reviewing your own unit test cases? (Select one option)

Not Motivated 1	Slightly Motivated 2	Motivated 3	Fairly Motivated 4	Very Motivated 5
○	○	○	○	○

6. 6. How much the interaction experience with the testing tool graphical interface contributes to this motivation? (Select one option)

No Contribution 1	Slight Contribution 2	Moderate Contribution 3	Fairly Contribution 4	Total Contribution 5
○	○	○	○	○

7. 7. When you are reviewing unit tests, is there any task that you generally do outside of the development environment (even using paper and pen) to help you follow a specific work-flow? If possible, cite an example of this task and its purpose.

_____
_____

8. 8. When you are reviewing unit tests, is there any task that you generally do outside of the development environment (even using paper and pen) to help you remember the purpose of a unit test case? If possible, cite an example of this task and its purpose.

_____
_____

page 3

9. 9. For unit test reviewing tasks, is there any textual information which is difficult or time consuming to interpret, that you think would be better represented graphically? How that would be?

_____
_____

10. 10. List some (if any) difficulties you experience interacting with the IDE's or testing framework's graphical interface, while performing unit test reviewing tasks. Examples include, but are not limited to: difficulty finding a certain unit test output; tool's documentation; difficulty locating a particular test case; configuration or menu options not clearly indicated, etc.

_____
_____

11. 11. In general, do you write test cases "from blank" or you reuse existing testing code? What motivates you to choose between both alternatives?

_____
_____

12. 12. The interpretation of the unit test case code is always enough for you to remember the purpose of it? If not, do you count on any other resource support?

_____
_____

13. 13. Do you usually document (comment) the unit test case code that you implement? (Select one option)

- ○ Yes
- ○ No

14. 13.1 When documenting a unit test case code written by you, is there any kind of details that you generally prioritize:(i) implementation details; or (ii) functionality details (e.g. the purpose of the test case)? If so, which one and why? [Answer this question only if answer to Q#13 is Yes]

_____
_____

15. 14. The unit test case code that you review and that was not implemented by you are in general documented (commented)? (Select one option)

- ○ Yes
- ○ No

page 4

Consider that a ''Flaky'' test case is any test case that generate an unpredictable or unreliable result, that is, it can pass or fail and you cannot predict, trust or realize its result easily due mistakes on the way it was implemented.

16. 15. How often do you discover flaky test cases while perfoming unit testing activities?

_____
_____

17. 16. How disturbing are these problems (flaky tests)? (Select one option)

Not Disturbing 1	Slightly Disturbing 2	Disturbing 3	Fairly Disturbing 4	Very Disturbing 5
○	○	○	○	○

18. 17. How complex for you is to think about the test case environment dependency (e.g. a test case result depending of an external service, file content, network resource etc.) when reviewing unit test cases? (Select one option)

Not Complex 1	Slightly Complex 2	Complex 3	Fairly Complex 4	Very Complex 5
○	○	○	○	○

19. 18. How complex for you is to think about the unit test case execution when it involves concurrent execution? (Select one option)

Not Complex 1	Slightly Complex 2	Complex 3	Fairly Complex 4	Very Complex 5
○	○	○	○	○

20. 19. How complex for you is to monitor the independence between the test cases execution when reviewing the unit test cases? (Select one option)

Not Complex 1	Slightly Complex 2	Complex 3	Fairly Complex 4	Very Complex 5
○	○	○	○	○

21. 20. In general, what information/actions make you realize that a unit test case that you wrote after a while had problems (was flaky)?

22. 21. Do you use any tool to help you on the task of reviewing unit test cases or flaky tests identification? If so, which?

23. 22. Do you believe that comparing or manually flagging/ranking certain unit test case properties would help on reducing the problems of flaky test case identification? If so, which properties and why?

page 5

24. 23. Do you have any comments or suggestions regarding the topics investigated in this survey?

24. In case you wish to be contacted in the future, about opportunities of collaboration to the current research or to receive a summary of the research results, please provide the following personal information:

25. Name:

26. E-mail:

27. Organization Name: