A Survey on Software Testing Education in Brazil

Leo Natan Paschoal
Institute of Mathematics and Computer Science,
University of São Paulo
São Carlos, São Paulo
paschoalln@usp.br

Simone do Rocio Senger de Souza
Institute of Mathematics and Computer Science,
University of São Paulo
São Carlos, São Paulo
srocio@icmc.usp.br

ABSTRACT

Software testing is one of the most important quality assurance activities. However, it is considered a challenge while teaching in undergraduate programs. One of the implied challenges is how to include this topic in computing undergraduate programs and in which level of detail. The industry has recognized the importance of this deeper and more solid formation. Nevertheless, in most cases, teaching software testing is distributed in disciplines and with different levels of detail. Each teaching method of software testing has its own advantages and disadvantages. Understanding this scenario is important for the proposition of improvements and innovations in the way of teaching software testing. This paper presents the results of a survey conducted throughout various Brazilian computing courses in different universities to identify which and how topics on software testing are taught. The objective is to identify the way in which the content is given, the support mechanisms used in the teaching practices, the challenges imposed and the instruments used to evaluate the students' learning. The survey was conducted in November of 2017 with Higher Education Institutions (HEI) lecturers distributed in different regions of Brazil, which have undergraduate programs in computer science. From the results, suggestions are presented and discussed in order to promote the dissemination of knowledge in this area of software engineering.

CCS CONCEPTS

• **Software and its engineering** → *Software verification and validation*; • **Social and professional topics** → Software engineering education;

KEYWORDS

Computer Science Education, Program Testing, Software Testing, Survey

ACM Reference Format:
Leo Natan Paschoal and Simone do Rocio Senger de Souza. 2018. A Survey on Software Testing Education in Brazil. In *17th Brazilian Symposium on Software Quality (SBQS), October 17–19, 2018, Curitiba, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3275245.3275289

SBQS, October 17–19, 2018, Curitiba, Brazil

ACM ISBN 978-1-4503-6565-9/18/10…$15.00
https://doi.org/10.1145/3275245.3275289

1 INTRODUÇÃO

Teste de software é uma atividade presente no processo de desenvolvimento de software, que é conduzida com a finalidade de verificar se um produto de software se comporta como previsto. Desse modo, é uma etapa do processo constituída por um conjunto de atividades em que se identificam possíveis irregularidades. O assunto é abordado pela Engenharia de Software e segundo a definição do *Guide to the Software Engineering Body of Knowledge* (SWEBOK) [23], consiste na verificação dinâmica de que um software fornece comportamentos esperados em um conjunto finito de casos de teste, os quais são adequadamente selecionados do domínio de execução.

Teste de software é o principal meio de verificação e validação usado durante o processo de desenvolvimento de software [7]. Essa atividade deve ocorrer em todas as fases do processo de desenvolvimento de software, porque defeitos podem estar presentes em qualquer uma delas [13]. Apesar da importância da realização do teste de software para comprometimento com a qualidade do software, essa atividade é muitas vezes negligenciada no ensino de cursos de computação. Estudos mencionam que os currículos dos cursos de computação destinam um nível muito baixo de atenção ao teste de software [5]. De acordo com Valle *et al.* (2015a) [26], isso pode ser reflexo dos currículos de referência, uma vez que as universidades elaboram as estruturas curriculares dos seus cursos levando em consideração os currículos de referência propostos pela *Association for Computing Machinery* (ACM) e pela Sociedade Brasileira de Computação (SBC), e esses currículos recomendam que o conteúdo seja abordado como unidades de conteúdos em disciplinas de engenharia de software.

Apesar dos currículos de referência recomendarem o ensino de teste de software como uma unidade de conteúdo da disciplina de engenharia de software, os currículos dos cursos de computação direcionam o ensino de engenharia de software para aspectos de projeto e implementação de software [17]. Assim, teste de software, muitas vezes, não é ensinado. Por consequência da formação não adequada, em virtude dos conceitos, fases, técnicas e ferramentas serem ensinadas brevemente em disciplinas como engenharia de software ou em disciplinas eletivas, os egressos de computação não são treinados adequadamente [28].

Os estudantes de Computação acabam entrando na indústria com pouca ou nenhuma experiência em teste de software [8], algumas vezes se graduam sem saber testar adequadamente um software [17]. Entretanto, é essencial e indispensável que esse profissional tenha um pouco de experiência, uma vez que quanto melhor a experiência do profissional durante o processo de teste de software, mais irregularidades conseguirá identificar [13].

Uma vez que o meio acadêmico não está conseguindo atender satisfatoriamente à demanda existente, com o intuito de oferecer

melhores condições para a condução e realização de educação em teste de software, alguns recursos didáticos digitais mais dinâmicos e intuitivos vêm sendo desenvolvidos, como jogos educacionais [10], redes sociais [9], módulos educacionais [1], entre outros. Também existem algumas pesquisas que relatam sobre a importância da abordagem de ensino, uma vez que algumas abordagens utilizadas no ensino de teste de software podem não ser eficientes, como destacado por Smith *et al.* (2012) [22]. Nesse sentido, é importante experimentar novas abordagens, principalmente quando as mesmas têm viés a aprendizagem ativa [19], dado que é necessário mais exposições às práticas de teste de software [17].

Apesar de existirem abordagem e mecanismos de apoio sendo construídos, não existe um panorama que demonstre quais são os conteúdos relativos à teste de software que são ensinados nos cursos de formação superior, como ciência da computação, sistemas de informação, engenharia de computação e até mesmo engenharia de software. Ou seja, mesmo quando teste de software está presente nos currículos das universidades, não está claro o que realmente é ensinado. Assim, sem o reconhecimento do que vem sendo ensinado, os mecanismos de apoio construídos correm o risco de ficarem subutilizados.

No contexto de educação superior brasileira em computação, alguns estudos tentam reconhecer se teste de software é ensinado, tomando como base a análise de alguns currículos de universidades [26, 5]. Todavia, uma análise baseada em documentos não consegue explorar alguns pontos como métodos de ensino e recursos didáticos utilizados pelo professor, técnica e ferramentas de teste de software que são previstas nos currículos e nos planos de ensino e que realmente são ensinadas, entre outros. Em contrapartida, um *survey* descritivo pode ser uma ferramenta interessante para tornar mais explícito o que a comunidade de computação ensina sobre teste de software.

Diante disso, visando reconhecer e estabelecer um panorama sobre o ensino de teste de software no Brasil, um *survey* foi planejado e executado com professores atuantes em cursos de educação superior em computação. Assim, este artigo tem o intuito de apresentar o processo de planejamento do *survey*, a condução e os resultados obtidos. Além disso, para apoiar o domínio e a disseminação do conhecimento na área, serão apresentadas algumas perspectivas de iniciativas que podem ser realizadas e esforços que merecem atenção.

Este artigo está estruturado da seguinte forma. Na Seção 2 é mencionado o processo de elaboração e condução do *survey*. Os resultados obtidos a partir da condução do *survey* são apresentados na Seção 3. A Seção 4 é destinada à discussão sobre os resultados. Por fim, na Seção 5 são apresentadas as considerações, limitações deste estudo e perspectivas de trabalhos futuros.

2 PLANEJAMENTO E EXECUÇÃO DO *SURVEY*

O *survey* foi planejado seguindo o processo proposto por Kasunic (2005) [15] para *design* efetivo de *surveys* para a área de engenharia de software. Além disso, foram utilizados direcionamentos descritos por Shull *et al.* (2008) [16]. Nesse sentido, o processo de condução do *survey* envolveu sete etapas, representadas na Figura 1.

No decorrer das próximas subseções serão apresentados detalhes do processo de condução.

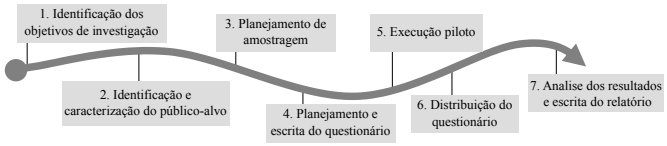

Figura 1: Processo adotado na condução do *survey*

2.1 Identificação dos objetivos de investigação

Em virtude do objetivo principal da condução do *survey* ser a identificação de como teste de software está sendo ensinado em cursos de graduação em computação no Brasil, procurou-se estabelecer quais informações necessárias deveriam ser coletadas. Assim, inicialmente procurou-se reconhecer os conteúdos que estão sendo ensinados, caracterizar o nível de conhecimento sobre teste de software dos professores que ministram o conteúdo, detectar as principais dificuldades em ensinar teste de software, descobrir as abordagens[1] utilizadas para ensinar o conteúdo, constatar os mecanismos de apoio ao ensino que são adotados pelos professores, e compreender como os professores mensuram o conhecimento dos alunos sobre o conteúdo ensinado. Assim, a partir do esclarecimento das necessidades, o objetivo principal foi fracionado em objetivos específicos. Com base nisso, os objetivos do *survey* foram especificados por meio da abordagem GQM (*Goal/Question/Metric*) proposta por Basili (1992) [3], descritos na Tabela 1.

Para apoiar a definição das questões e métricas foram utilizados alguns artefatos de apoio como: o guia SWEBOK, a ontologia OntoTest – uma ontologia de Teste de Software [2], e os resultados do mapeamento sistemático conduzido por Valle *et al.* (2015b) [27].

2.2 Identificação e caracterização do público-alvo

O público-alvo definido para participar do *survey* é composto por professores que lecionaram a disciplina de teste de software (ou similares) ou engenharia de software em cursos superiores de computação (ciência da computação, sistemas de informação, engenharia de computação ou engenharia de software), que são autorizados e reconhecidos pelo Ministério da Educação (MEC), que estejam disponíveis no sistema de tramitação eletrônica dos processos de regulação e-MEC[2], regulamentados pelo Decreto nº. 5.773, de 9 de maio de 2006[3], que não estejam extintos ou em fase de extinção, isto é, que estão em atividade. Após a definição do público-alvo, foi planejado o processo de obtenção de amostras.

2.3 Planejamento de amostragem

O processo de obtenção de amostras da população, foi planejado para seguir uma metodologia que oferecesse subsídios para obtenção de resultados da amostra que fossem representativos de toda a população. Assim, em um primeiro momento pretendeu-se utilizar uma amostragem probabilística, em que cada elemento da população tivesse a mesma probabilidade de ser selecionado para

[1] Entende-se como abordagem a forma como o conteúdo é abordado. Assim, envolve método de ensino, modelo pedagógico, dentre outras.
[2] Disponível em: <http://emec.mec.gov.br/>
[3] Mais informações disponíveis em: <http://www2.mec.gov.br/sapiens/portarias/dec5773.htm>

Tabela 1: Objetivos do *survey*

Objetivo (Goal)	Questão (Question)	Métrica (Metric)
Reconhecer os conteúdos que estão sendo ensinados.	1. Quais os níveis de teste que são ensinados?	1. Conjunto de opções com os principais níveis de teste.
	2. Quais técnicas de teste são ensinadas?	2. Conjunto de opções com as principais técnicas de teste de software.
	3. Quais os critérios de cada técnica de teste de software são ensinados?	3. Conjunto de opções de critérios agrupados conforme as principais técnicas de teste.
	4. Selecione e/ou indique outros temas relacionados que são ensinados.	4. Conjunto de opções com outros temas.
Caracterizar o nível de conhecimento dos professores que ministram o conteúdo.	1. Qual o conhecimento dos professores sobre a existência da atividade?	1. Conjunto de opções que possibilitem os professores caracterizarem os seus níveis de conhecimento.
Constatar os principais desafios em ensinar o conteúdo.	1. Quais são as principais dificuldades em ensinar teste?	1. Resposta aberta
Descobrir as abordagens utilizadas para ensinar o conteúdo.	1. Quais são os tipos de abordagens utilizadas para ensinar teste?	1. Lista de abordagens identificadas por Valle, Barbosa e Maldonado (2015) + Flipped Classroom.
Detectar os mecanismos de apoio ao ensino que são adotados pelos professores.	1. Quais são os mecanismos de apoio/ recursos didáticos digitais que são utilizados para auxiliar no ensino de teste de software?	1. Lista de recursos segundo o que foi identificado por Valle, Barbosa e Maldonado (2015).
Compreender como os professores mensuram o conhecimento dos alunos sobre o conteúdo.	1. Quais os instrumentos utilizados para mensurar o conhecimento dos estudantes sobre o conteúdo ensinado?	1. Lista de instrumentos avaliativos.

constituir a amostragem [6]. Para tanto, como estratégia para identificar os professores atuantes foi considerado localizar os cursos de graduação em atuação por meio do sistema e-MEC e entrar em contato com os coordenadores dos cursos para que estes passassem o contato dos professores que atuam em cada instituição. Como engenharia de software é uma disciplina recomendada pelos currículos de referências, acredita-se que cada instituição de ensino deve oferecer a disciplina e, portanto, cada instituição deve possuir pelo menos um professor que já tenha ministrado o conteúdo de teste de software, mesmo que em uma disciplina não específica.

Foram identificados os cursos de graduação em computação (bacharelado) em atividades no Brasil. Em outubro de 2017 foi realizada uma busca no portal e-MEC a fim de localizar os cursos e os coordenadores vinculados a esses cursos, que resultou em 1438 cursos em atividade. A partir dos dados coletados sobre os cursos, foi possível observar que o campo "coordenador", não estava preenchido em muitos cursos, comprometendo a identificação. Além disso, os cursos que possuíam um coordenador cadastrado não possuíam o contato do coordenador, dificultando o processo de identificação para o possível contato. Essas adversidades acabaram dificultando a obtenção de amostras probabilísticas, como seria desejável. Com isso, foi necessário aplicar processos não probabilísticos de amostragem.

Com a inacessibilidade a toda a população, recorreu-se às listas eletrônicas de comissões especiais[4], mantidas por profissionais e instituições participantes da SBC. Essas listas possuem como membros pesquisadores e professores da área de computação. Nesse sentido, o tipo de amostragem foi definido como não probabilística, selecionada por conveniência.

2.4 Planejamento e escrita do questionário

Nesta etapa do processo foram revistos os objetivos pré-definidos na primeira etapa do processo, bem como a forma como os dados dos participantes que compõem a amostra seriam coletados. Considerando que os participantes são assinantes das listas de *e-mails* das comissões mencionadas, foi definido que o *survey* seria disponibilizado eletronicamente, em formato de questionário eletrônico. Ele foi construído utilizando a ferramenta Google Forms[5]. O questionário *web* foi estruturado por sete seções. As questões estão disponíveis em: <http://bit.ly/2PHfKac>.

A primeira seção contém uma apresentação do *survey* em que é descrito o objetivo do mesmo, o público-alvo, esclarecimentos sobre a importância da participação de professores atuantes na área, as datas importantes que delimitam o prazo em que o *survey* estaria sendo disponibilizado para receber respostas e uma declaração. Ela também possui um link para um glossário[6] dos termos que aparecem no *survey*.

A segunda seção foi construída para coletar informações que caracterizam os participantes, são elas: nome do participante, afiliação, estado da instituição de ensino, e a maior titulação acadêmica do professor. Esses dados também são utilizados para validação dos participantes, visto que, por decorrência do questionário ser aberto, corria-se o risco desses participantes não serem professores atuantes em cursos de computação. Assim, a partir dessas informações, foi possível validar a autenticidade recorrendo-se a plataforma Lattes[7]. Nessa plataforma, os currículos dos professores que responderam o *survey* foram identificados e analisados, visando confrontar os dados que os mesmos informaram nesta seção.

A terceira seção teve o intuito de coletar informações sobre a modalidade de ensino em que o conteúdo de teste de software é ministrado pelos professores, o curso em que o conteúdo é ministrado e se o curso em que cada respondente atua possui uma disciplina específica destinada ao ensino de teste de software. Essa última questão foi definida com o objetivo de ser utilizada em apoio a um desvio condicional construído. Nesse sentido, se o respondente marcar a opção que o curso possui uma disciplina específica, ele é convidado a responder as questões disponíveis na quarta seção, caso contrário, será direcionado para a quinta seção.

A quarta seção foi estabelecida com vistas a coleta de informações sobre as disciplinas ministradas pelos participantes do *survey*.

[4]Mais informações podem ser obtidas em: <http://www.sbc.org.br/22-destaques/34-listas-eletronicas>
[5]Mais informações podem ser obtidas em: <https://www.google.com/forms/about/>
[6]O glossário encontra-se disponível em: <http://bit.ly/2NsPHlO>
[7]Mais informações podem ser obtidas em: <http://lattes.cnpq.br/>

Essa seção só é respondida por professores que atuam em disciplinas específicas sobre teste de software. Foi definida para possibilitar uma análise dos dados por grupos, em que um grupo é constituído por respostas de professores que ministram o conteúdo em disciplinas específicas e o outro grupo representa as respostas sobre o conteúdo ministrado em disciplinas não específicas. Assim, destina-se a coleta de dados sobre cada disciplina, a saber: carga-horária da disciplina, formato de oferta da disciplina (obrigatório ou opcional).

A quinta seção aborda as questões que foram estipuladas para recolher informações como os níveis de teste, as técnicas de teste de software, e os critérios das principais técnicas. Também foi escrita uma questão para os professores indicarem outros assuntos que são ensinados. Essas questões foram definidas com apoio de um especialista na área. Adicionalmente, foram definidas duas questões que não estão diretamente relacionadas a identificação do conteúdo que é ministrado pelos professores. Uma delas foi determinada para os professores relatarem as principais dificuldades que os mesmos possuem ao ministrar o conteúdo de teste de software e outra é destinada a uma autoanálise sobre o conhecimento sobre o conteúdo que é ministrado.

Na sexta seção foram disponibilizadas duas questões. Uma questiona os professores sobre os tipos de abordagens que são utilizadas para ensinar teste de software. Foram definidas como alternativas de respostas para essa questão as abordagens identificadas por Valle *et al.* (2015b) [27] e acrescentou-se como opções algumas abordagens que são utilizadas no ensino de engenharia de software. A outra pergunta foi desenvolvida para determinar quais os mecanismos de apoio que são utilizados pelos professores atuantes no ensino dos conteúdos de teste de software. Como opções de respostas para essa questão, foram utilizados os mecanismos identificados por Valle *et al.* (2015b) [27] e foram adicionados outros recursos que poderiam ser pertinentes.

A sétima e última seção do *survey* foi desenvolvida para atingir o último objetivo pré-definido na abordagem GQM. Uma questão interroga os professores sobre os instrumentos que os mesmos utilizam para mensurar o conhecimento dos estudantes. Algumas opções de respostas foram definidas e também foi disponibilizado um espaço para os professores descreverem sobre outros instrumentos. Após a definição e construção do questionário eletrônico, foi realizada uma execução piloto, visando analisar a validade do instrumento.

2.5 Execução piloto

Kasunic (2005) [15] menciona sobre a importância da execução piloto do *survey*, a fim de detectar os problemas existentes no mesmo, verificar se as perguntas são compreensíveis, se as perguntas certas foram feitas para atingir o objetivo, e quanto tempo os participantes levam para completarem o questionário. Nesse sentido, o estudo piloto contribui com a qualidade do instrumento. Uma forma de verificar a validade de um instrumento é por meio da avaliação do seu conteúdo [11], que por sua vez consiste em uma medida subjetiva para identificar o quanto os itens de um instrumento são apropriados, por meio da opinião de especialistas [18]. Hauck *et al.* (2011) [14] prepararam quatro questões abertas para avaliar o conteúdo de *surveys*, em execuções piloto: (1) o questionário contém tudo que é esperado para contemplar o seu objetivo? (2) o questionário contém quaisquer informações não desejáveis ou desnecessárias ao contexto e objetivo da pesquisa? (3) você conseguiu compreender adequadamente as perguntas? (4) existe algum erro ou inconsistência no questionário? .

Um grupo de professores e alunos de doutorado do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, atuantes na área de teste de software, foram convidados por *e-mail* para participar do teste piloto. Esses especialistas foram escolhidos pelos critérios de disponibilidade e proximidade com o grupo onde esta pesquisa foi desenvolvida. Cinco especialistas (dois doutorandos e três doutores) participaram do estudo piloto, respondendo as questões definidas por Hauck *et al.* (2011) [14] e emitindo *feedback* sobre o *survey*. A avaliação dos especialistas foi positiva, com sugestões para: diminuir a quantidade de perguntas, deixando algumas não obrigatórias; reduzir o tempo estimado para responder o *survey* para 10 minutos (ao invés de 30 minutos) e sugestão para incluir caixas de seleção para algumas perguntas.

2.6 Distribuição do questionário

A partir do parecer dos especialista, ajustes foram feitos no *survey* para que o mesmo pudesse ser distribuído para a amostragem. O questionário foi distribuído por meio de *e-mails* cadastrados nas listas das comissões mencionadas. Essa distribuição ocorreu entre 13 e 30 de novembro de 2017. Depois do fim do período em que o questionário ficou disponível para recebimento de respostas, os dados coletados foram analisados, visando estabelecer como os conteúdos de teste de software são ensinados.

3 RESULTADOS

3.1 Caracterização dos participantes

O *survey* foi respondido por cinquenta e dois professores, atuantes em diferentes instituições de ensino, de diferentes estados brasileiros, que abrangem as cinco regiões do país. A Tabela 2 apresenta o número de participantes de acordo com o estado e região do país. Percebe-se que o maior número de participantes atua na região Sudeste (20 participantes – 38%), seguido pela região Sul (19 participantes – 36,54%), região Nordeste (8 participantes – 15,38%), Centro-Oeste (3 participantes – 5,77%) e por último a região Norte (2 participantes – 3,85%). Apesar da pesquisa ter conseguido respondentes de diversos lugares, não foi possível obter a participação de professores de todos os estados brasileiros. Assim, uma análise por estado não foi possível. Também não foi possível caracterizar como teste de software é ensinado por região do Brasil, porque a variância amostral é alta (74,30), ou seja, o número total de participantes não está concentrado em torno da média.

Para apoiar a caracterização, foi solicitado aos participantes que informassem a maior titulação acadêmica. A partir disso foi possível observar que a maioria dos participantes possuem doutorado (61,54%). Do total de participantes, 36,54% possuem a titulação de mestre e 1,92% de especialista.

Apenas 1 professor dos 52 participantes ministra a disciplina na modalidade de ensino a distância (EAD). Como o número de professores que ministram disciplinas em EAD é muito pequeno (1,92% em relação aos 52 participantes), a análise foi realizada desconsiderando se a disciplina é ministrada na modalidade EAD ou presencial.

Tabela 2: Origem dos participantes

Região	Estado	Número de participantes	Frequência por estado
Centro-Oeste	Goiás	3	5,77%
Nordeste	Bahia	1	1,92%
	Paraíba	3	5,77%
	Pernambuco	1	1,92%
	Piauí	3	5,77%
Norte	Amazonas	2	3,85%
Sudeste	Minas Gerais	6	11,54%
	Rio de Janeiro	1	1,92%
	São Paulo	13	25,00%
Sul	Paraná	11	21,15%
	Rio Grande do Sul	8	15,38%

Foi definida uma questão buscando identificar em quais cursos os participantes atuam. Dentre os 52 participantes, alguns informaram que atuam em mais de um curso. Para apoiar a análise, um diagrama de Venn foi utilizado. Assim, na Figura 2 é possível observar relações de união e intersecção entre os cursos em que os participantes atuam. Com ele é possível observar que 18 professores atuam apenas no curso de ciência da computação, 5 professores atuam em engenharia da computação, 4 professores atuam apenas em engenharia de software e 9 professores atuam apenas em sistemas de informação. Os demais professores atuam em mais de um curso. Além disso, é possível observar, por exemplo, que nenhum dos participantes atua em todos os quatro cursos.

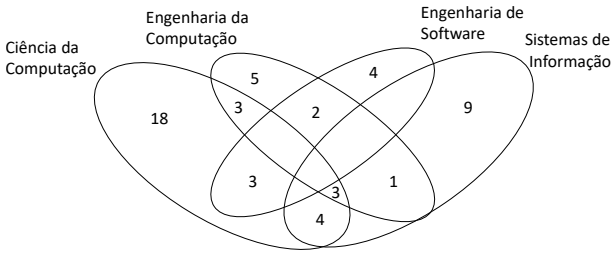

Figura 2: Cursos de atuação

Levando em consideração os trabalhos conduzidos por Silva *et al.* (2011) [21], Benitti e Albano (2012) [5] e Valle *et al.* (2015a) [26], foi criada uma questão para averiguar se os participantes ministram o conteúdo de teste de software em uma disciplina específica para o conteúdo – em que apenas teste de software é ensinado – ou uma disciplina não específica como engenharia de software. Como resultado, vinte e oito participantes (54%) relataram que ministram o conteúdo em disciplina não específica como um conteúdo da disciplina, e vinte e quatro (46%) lecionam o conteúdo em uma disciplina específica de teste de software. Como o número de participantes que ministra em disciplina específica é próximo do número de participantes que ministra o conteúdo em disciplina não específica, a análise foi realizada considerando essa divisão.

Os 24 participantes que ensinam teste de software como uma disciplina específica apresentaram algumas informações sobre a disciplina. Foram solicitadas as seguintes informações: o nome da disciplina, a carga-horária da disciplina em horas e se a disciplina é oferecida aos alunos como uma disciplina obrigatória ou uma disciplina optativa[8].

A Tabela 3 apresenta os nomes que as disciplinas recebem. Conforme é possível observar, não existe uma unanimidade em questão de nomenclatura. É possível destacar que alguns professores ensinam o conteúdo em disciplinas de tópicos de engenharia de software. Nota-se também dois pontos interessante: (1) das 24 disciplinas de teste, uma é específica para a técnica de teste baseado em modelos; (2) teste de software é ensinado por 6 professores como uma atividade de verificação e validação.

Tabela 3: Nomenclatura das disciplinas especificas de teste de software

Nome da disciplina	Participantes	Frequência
Teste de software	4	16,67%
Tópicos em Engenharia de Software	4	16,67%
Verificação e validação	4	16,67%
Teste e inspeção de software	2	8,33%
Verificação e validação de software	2	8,33%
Inspeção e teste de software	1	4,17%
Processos de teste de software	1	4,17%
Teste e confiabilidade de software	1	4,17%
Teste e depuração de código	1	4,17%
Teste e depuração de software	1	4,17%
Testes de Software	1	4,17%
Testes de software baseado em modelos	1	4,17%
Tópicos em Tecnologia da Informação	1	4,17%

Em relação a carga-horária, observa-se que as instituições, quando possuem uma disciplina específica de teste de software, destinam em média 59,4 horas para ensinar o conteúdo de teste de software. A Tabela 4 apresenta a carga-horária das disciplinas em relação ao número de participantes. Na tabela, nota-se que um professor ministra o conteúdo em uma disciplina com 85 horas, e dois professores mencionaram que ministram o mesmo conteúdo em uma disciplina com carga-horária de 30 horas.

Tabela 4: Carga-horária das disciplinas

Carga-horária da disciplina	Participantes	Frequência
60 horas	11	45,83%
75 horas	3	12,50%
30 horas	2	8,33%
72 horas	2	8,33%
32 horas	1	4,17%
45 horas	1	4,17%
51 horas	1	4,17%
64 horas	1	4,17%
80 horas	1	4,17%
85 horas	1	4,17%

Para finalizar as informações sobre as disciplinas, foi solicitado que os participantes informassem se a disciplina de teste é obrigatória ou optativa. 62,5% dos participantes mencionaram que a

[8]Entende-se como disciplina optativa aquela em que o aluno pode optar por fazer ou não.

disciplina é oferecida como obrigatória e 37,5% mencionaram que ela é oferecida como optativa. Com isso, percebe-se que quando os cursos destinam uma disciplina específica para teste de software, a maioria oferece a disciplina como obrigatória.

3.2 Conteúdos ensinados

Quando questionados sobre quais são os níveis de teste de software ensinados em cursos de computação, os resultados da pesquisa apontam que os principais níveis de teste (teste de unidade, teste de integração e teste de sistema) são ensinados tanto pelos professores que ministram disciplinas específicas de teste quanto professores que ministram teste em uma disciplina não específica (*e.g.* engenharia de software). A Tabela 5 apresenta os resultados completos, considerando o nível de teste, número de professores que ministra disciplina específica e número de professores que ministra disciplina não específica.

Tabela 5: Níveis de teste ensinados

Nível de teste	Disciplina específica	%	Disciplina não específica	%
Teste de unidade	24	100	28	100
Teste de integração	23	95,83	23	82,14
Teste de sistema	17	70,83	22	78,57
Teste de aceitação	3	12,5	1	3,57
Teste de estresse	1	4,17	2	7,14
Teste de carga	1	4,17	0	0
Teste de regressão	1	4,17	0	0
Teste de usabilidade	1	4,17	0	0
Teste de desempenho	0	0	1	3,57
Teste de escalabilidade	0	0	1	3,57
Teste de lançamento	0	0	1	3,57

Todos os professores ensinam teste em nível de unidade, mais da metade ensina teste de integração e teste de unidade. Com os resultados, é possível observar que a maioria dos professores concentra os seus esforços em ensinar os alunos a realizar teste em pequenas unidades de software. Acredita-se que um fator que pode causar esses resultados está associado com a complexidade de se trabalhar com testes de software que consigam contemplar um sistema completo. Além disso, alguns professores relataram que outros níveis são ensinados, tais como teste de aceitação, teste de regressão, dentre outros.

Em relação às técnicas de teste de software que são ensinadas, destacam-se o teste funcional e o teste estrutural. Ambas as técnicas são ensinadas por mais da metade dos professores que ministram o conteúdo em uma disciplina específica e disciplina não específica. No caso da técnica de teste funcional, ela é ensinada por todos os professores que ministram o conteúdo em uma disciplina específica, mas nem por todos os que ministram em disciplina não específica. Observa-se que quando há uma disciplina específica para o conteúdo, as técnicas de teste baseado em defeito e teste baseado em modelos são ensinadas por mais da metade dos professores. Em contrapartida, quando a disciplina não é específica, mais da metade deixa de ensiná-las. A Tabela 6 apresenta os resultados detalhados.

Um possível motivo que pode estar relacionado com o número de professores que ensinam teste baseado em defeitos é o alto custo associado à atividade. Em relação ao teste baseado em modelos a construção de modelos formais de sistemas complexos, pode ser extremamente complexa. Acrescenta-se também que os livros de referência utilizados por grande parte das instituições de ensino brasileiras podem também influenciar, pois alguns livros que não são específicos sobre teste de software não exploram detalhes sobre essa técnica. Os resultados da pesquisa apontam direcionamentos importantes, visto que aparentemente os professores dão preferência ao ensino de uma técnica de Teste de Software que utiliza como fonte de dados à documentação do software e não a implementação do software.

Tabela 6: Técnicas de teste ensinadas

Técnicas de teste	Disciplina específica	%	Disciplina não específica	%
Teste funcional	24	100,00	25	89,29
Teste estrutural	19	79,17	20	71,43
Baseado em defeitos	18	75,00	8	28,57
Baseado em modelos	14	58,33	7	25,00

Apesar de o teste funcional ser ensinado por 100% dos professores que ministram o conteúdo em uma disciplina específica, nem todos eles ensinam os critérios dessa técnica. Dos 24 professores, apenas 20 mencionam os critérios dessa técnica. Os dados revelaram que o critério particionamento em classes de equivalência é ensinado por 20 (83,33%) dos 24 professores, sendo o critério do teste funcional que se sobrepõem aos demais. Esse critério é seguido por análise do valor limite que é lecionado por 19 (79,17%) dos 24 participantes. De maneira semelhante, a técnica de teste funcional e seus respectivos critérios aparecem em maior intensidade que as outras técnicas quando o conteúdo é ensinado em disciplinas não específicas de teste.

Teste estrutural é a segunda técnica de teste com maior repercussão nos dados obtidos, ensinado por mais da metade dos professores, tanto por aqueles que ministram o conteúdo em disciplina específica quanto por aqueles que não ministram. Os dois principais critérios dessa técnica (critérios baseados em fluxo de controle e baseados em fluxo de dados) são ensinados por pelo menos metade dos professores que ministram o conteúdo em disciplina específica. Entretanto, o mesmo não ocorre quando a disciplina não é específica, isto é, nesse caso apenas os critérios baseados em fluxo de controle são ensinados.

Em relação aos critérios do teste baseado em defeitos, quando teste é ensinado em disciplina específica, mais da metade dos participantes (75%) revelaram que ensinam o critério teste de mutação. Em contrapartida, quando essa técnica é ensinada em uma disciplina não específica, mais da metade (71,43%) dos participantes revelou que não a ensina.

O teste baseado em modelos é pouco explorado pelos professores que representam a amostragem desta pesquisa. Acredita-se que teste baseado em modelos não seja tão explorado quanto os critérios de teste funcional, por exemplo, porque os casos de teste precisam ser derivados de um modelo. Esse modelo deve ser criado a partir da especificação do programa ou software que se deseja testar. Assim, a execução da atividade pode consumir um tempo que muitas vezes o professor não tem disponível em sala de aula.

A Tabela 7 apresenta os resultados em relação aos critérios de teste de software que são ensinados, por técnica de teste.

Tabela 7: Critérios de teste ensinados por técnica

Técnica de teste	Critério de teste	Disciplina específica	%	Disciplina não específica	%
Funcional	Particionamento em classes de equivalência	20	83,33	19	67,86
	Análise do valor limite	19	79,17	16	57,14
	Grafo causa-efeito	11	45,83	10	35,71
	Teste funcional sistemático	11	45,83	8	28,57
Estrutural	Critérios baseados em fluxo de controle	19	79,17	18	64,29
	Critérios baseados em fluxo de dados	12	50,00	11	39,29
Baseado em defeitos	Teste de Mutação	18	75,00	8	28,57
Baseado em modelos	Máquina de estados finitos	8	33,33	1	3,57
	Geração de sequência de teste	11	45,83	7	25,00

Para finalizar a seção de reconhecimento sobre os assuntos de teste de software que são ensinados, os participantes tiveram que indicar outros temas relacionados que eles ensinam. Observa-se na Tabela 8 os outros assuntos que são ensinados. Por meio da tabela, nota-se que os conteúdos depuração de software, outras atividades de VV&T, e terminologia de teste são abordados em uma maior porcentagem por professores que atuam em disciplinas não específicas de teste. Uma observação que merece ser destacada é em relação a terminologia de teste, que embora seja um assunto abordado por mais da metade dos professores que compõem a amostra, é um tema que é ensinado menos do que os principais níveis de teste, e que as técnicas e alguns critérios de teste.

Tabela 8: Outros assuntos ensinados sobre teste

Outros assuntos ensinados	Disciplina específica	%	Disciplina não específica	%
Ferramentas de teste	19	79,17	22	78,57
Terminologia de teste	16	66,67	19	67,86
Outras atividades de verificação, validação e teste (VV&T)	13	54,17	18	64,29
Documentação de teste	12	50,00	12	42,86
Processo de teste	11	45,83	12	42,86
Geração automática de dados de teste	7	29,17	4	14,29
Depuração de software	6	25,00	9	32,14

3.3 Nível de conhecimento dos professores

Visando caracterizar o nível de conhecimento dos professores que ensinam teste de software, que compõem a amostragem, foi solicitado aos mesmos que identificassem o quanto sabem sobre teste. Foi pré-definido quatro opções: o professor possuí conhecimento geral sobre teste, mas nunca aplicou uma técnica e um critério de teste de software; possui conhecimento em nível de aplicação em sala de aula e já conduziu a atividade em aplicações em sala de aula; possui conhecimento em nível de pesquisa acadêmica, isto é, faz pesquisa na área de teste; possui conhecimento em nível industrial, já tendo conduzido a atividade na indústria de software. Conforme é possível observar na Tabela 9, a maioria dos participantes possui conhecimento em nível de pesquisa acadêmica. Os dados também revelam que existe um número considerável de professores com experiência de conduzir a atividade de teste no âmbito industrial.

Tabela 9: Nível de conhecimento dos professores

Nível de conhecimento	Participantes	Frequência
Conhecimento geral	6	11,54%
Conhecimento em nível de aplicação em sala de aula	13	25,00%
Conhecimento em nível de pesquisa acadêmica	19	36,54%
Conhecimento em nível de teste em aplicações na indústria de software	14	26,92%

3.4 Desafios em ensinar teste de software

Com o propósito de constatar quais são os principais desafios e dificuldades em ensinar o conteúdo de teste de software, foi solicitado aos participantes que escrevessem quais são as dificuldades. A Tabela 10 apresenta as principais dificuldades listadas pelos participantes. Conforme esperado, os professores relataram que o conteúdo, por ser amplo, requer uma disciplina específica. Um número elevado de professores destacou que é difícil fazer com que os alunos compreendam a real importância da atividade, um professor inclusive relatou "tenho a impressão que [os alunos] estão sempre focados em aprender programação e não dão tanta importância para os testes". Dos 52 professores, três informaram que não sentem dificuldades em ensinar o conteúdo.

3.5 Abordagens de ensino

O *survey* conduzido também permitiu revelar os tipos de abordagens que estão sendo utilizadas pelos professores que participaram da amostragem. Os resultados da pesquisa (Tabela 11) apontam que a abordagem tradicional continua protagonizando as práticas de ensino, no entanto, os professores estão utilizando outras abordagens de ensino, normalmente mesclando o tradicional com uma abordagem com viés às metodologias ativas de aprendizagem.

3.6 Mecanismos de apoio

Também foram identificados os mecanismos de apoio ao ensino de teste de software que são utilizados pelos participantes. Os dados da pesquisa apontam que 46 (88%) dos 52 participantes ensinam

Tabela 10: Dificuldades encontradas pelos professores

Dificuldades/desafios	Participantes
Falta tempo para ensinar o conteúdo, por causa de não se ter uma disciplina específica para ensinar o conteúdo	15
Fazer o aluno compreender a importância da atividade	12
Ensinar práticas de teste	9
Faltam materiais didáticos com exemplos práticos e relevantes	6
As ferramentas são difíceis de serem utilizadas	4
Falta de acesso a ferramentas (*open-source*) comerciais	3
Falta de experiência com o assunto	3
Ensinar os critérios do teste estrutural	2
Pouca estabilidade das ferramentas e frameworks	2
Falta integração do conteúdo com outras disciplinas do curso	1
Falta tempo para relacionar com outras atividades do ciclo de vida	1

Tabela 11: Abordagens utilizadas no ensino de teste

Abordagens de ensino	Participantes
Método de ensino tradicional	43
Aprendizagem baseada em projetos	20
Aprendizagem baseada em problemas	17
Revisão por pares	15
Ensino conjunto de teste com programação	13
Flipped classroom	3
Aprendizagem baseada em jogos	1

utilizando slides, 41 (79%) utilizam capítulo de livro, 28 (54%) fazem uso de artigos, 25 (48%) aproveitam-se dos benefícios dos Ambientes Virtuais de Aprendizagem, 20 (38%) utilizam mecanismos no formato de tutoriais. Em relação aos mecanismos menos utilizados aparecem o vídeo, utilizado por apenas 9 (17%) dos 52 respondentes; *quizes*, usado por 5 (10%) professores; módulos educacionais, utilizados por 4 (8%) professores, redes sociais e jogos educacionais ambos utilizados por 3 (6%) participantes; GitHub[9] e *podcasts* mencionados por 1 (2%) professor. A Tabela 12 resume essas informações.

Tabela 12: Mecanismos de apoio utilizados

Mecanismos de apoio	Participantes
Slides	46
Capítulos de livro	41
Artigos	28
Ambientes virtuais de aprendizagem	25
Tutoriais	20
Vídeos	9
Quizzes	5
Módulos educacionais	4
Jogos educacionais	3
Redes sociais	3
Podcasts	1
Vídeo chamada com profissionais da indústria	1

[9]Mais informações disponíveis em: <https://github.com/>

3.7 Avaliação do conhecimento

Por fim, o último objetivo do *survey* teve o intuito compreender como os professores mensuram o conhecimento dos estudantes. Algumas opções de instrumentos avaliativos foram disponibilizadas. Nota-se na Tabela 13 que o instrumento que se destacou foi "trabalhos práticos", uma vez que 45 (87%) dos 52 participantes assinalaram essa opção. Outros instrumentos que são bastante utilizados são as provas dissertativas (37 – 71% – dos 52 participantes), sendo seguido por provas objetivas (19 – 37% – dos 52), provas práticas (18 – 35% – dos 52), trabalhos de pesquisa (13 – 25%), estudos de caso (11 – 21%). O instrumento que é menos utilizado para mensurar o conhecimento dos estudantes é "atividades para prática do conhecimento (exercícios)", sendo indicado por apenas 1 (2%) participante. A Tabela 13 apresenta os resultados encontrados.

Tabela 13: Instrumentos avaliativos utilizados

Instrumentos de avaliação	Participantes
Trabalhos práticos	45
Provas dissertativas	37
Provas objetivas	19
Provas práticas	18
Trabalhos de pesquisa	13
Estudos de caso	11
Lista de exercícios	1

4 DISCUSSÕES

Este artigo focou especificamente em tentar estabelecer um panorama sobre como teste de software é ensinado no Brasil, sem se basear em análise documental. Assim, a principal diferença entre os trabalhos já conduzidos [21, 5, 26] que buscaram estabelecer um panorama, consiste em fazer uma pesquisa utilizando um método de pesquisa quantitativo. Para tanto, a pesquisa buscou contactar os professores que atuam na área de teste de software e engenharia de software. Apesar da dificuldade em conseguir uma amostragem probabilista da população, foi possível obter uma significativa participação de professores, atuantes na comunidade brasileira de computação. Com um número limitado de participantes, a pesquisa foi conduzida tentando constatar evidências sobre os conteúdos de teste que são ensinados e como eles são abordados pelos professores.

Em relação ao conteúdo, foi possível observar que independentemente do assunto ser ministrado em uma disciplina específica ou não, os professores se esforçam para ensinar os principais níveis e técnicas de teste de software. Mesmo existindo um esforço pela comunidade, quando esse assunto é ensinado em uma disciplina de engenharia de software, não são vistos todos os critérios de teste. O mesmo ocorre quando há uma disciplina específica, pois nem sempre o professor abrange todos os critérios de teste mais conhecidos. Pode ser que isso tenha relação com a carga-horária da disciplina, uma vez que foi observado que não há uma uniformidade em relação à carga horária quando comparadas diferentes disciplinas. Um resultado relevante, mas que já era esperado, é que os critérios de técnicas que possuem um custo mais elevado para ser aplicado, por exemplo, teste baseado em defeitos e teste baseado em modelos, são ensinados por mais da metade dos participantes quando o conteúdo é lecionado em uma disciplina específica.

Ainda em relação ao conteúdo, um resultado interessante foi observado. Conceitos e terminologias sobre teste de software não são abordados por todos os professores que participaram. Inicialmente, quando o *survey* foi planejado, acreditava-se que esse seria um assunto abordado pelos participantes, pois é esse tópico que define as diferenças entre as principais terminologias (*e.g.* engano, defeito, erro, falha, dado de teste, caso de teste, conjunto de casos de teste, dentre outros). No estudo de Delamaro *et al.* (2016) [12] os autores reforçam a importância de definir os termos de jargão antes de começar a entender o conteúdo de teste de software.

Outro resultado interessante refere-se ao nível de conhecimento e formação dos participantes. A maioria possui o nível mais alto em termos de titulação acadêmica, o que pode significar que possuem uma formação mais sólida na área. Além disso, a maioria atua na pesquisa em teste de software, ou seja, não apenas ensinam teste, mas buscam contribuir com o avanço do estado da arte e da prática na área. O outro resultado importante é que existe um número interessante de participantes que adquiriram experiência de teste na indústria de software.

Com a condução do *survey* foi possível observar as reais dificuldades do docente ao lecionar o conteúdo, destacando que as ferramentas que apoiam a condução da atividade não são fáceis de utilizar, algumas delas estão desatualizadas. Outro ponto interessante é que os professores estão encontrando dificuldades em motivar os alunos a fazê-los compreender a importância da atividade para a qualidade do software. Acredita-se que essa dificuldade possa estar relacionada com a estratégia de ensino adotada, visto que, os resultados deixam evidente que o método tradicional de ensino é utilizado pela maioria dos professores. Por outro lado, observa-se também que vários professores estão procurando empregar metodologias ativas de aprendizagem, as quais são muito adequadas a essa disciplina. Essas metodologias oferecem oportunidades para o aluno participar de uma maneira mais efetiva na disciplina [24].

Em relação aos mecanismos de apoio que são utilizados no ensino de teste, os resultados foram surpreendentes. Hoje, observa-se no estado da arte diversos tipos de mecanismos sendo desenvolvidos como jogos educacionais [20], ambientes de apoio ao ensino [25], módulos e objetos de aprendizagem [4]. Apesar de existirem esforços na comunidade para desenvolver esses mecanismos, nota-se que os resultados do *survey* revelam que os mesmos vêm sendo subutilizados. Por exemplo, apenas três professores dos 52 utilizam os jogos educacionais. Os resultados instigantes propõem uma reflexão a respeito da produção de material didático que não está sendo utilizado. Os materiais didáticos certos estão sendo desenvolvidos? Nesta pesquisa, os professores informaram que há uma necessidade por materiais didáticos com exemplos reais, que ofereçam suporte ao professor para demonstrar exemplos mais próximos do "mundo real". Os resultados encontrados podem apoiar investigações futuras.

Por fim, foi possível identificar como os professores verificam se os seus alunos estão aprendendo o conteúdo. Os resultados observados estavam de acordo com o que se espera na área, ou seja, mais da metade dos professores utilizam trabalhos práticos na disciplina. Teste de software é um conteúdo bastante prático, similar à codificação de software. Outro resultado que não ocorreu como era esperado foi em relação ao instrumento lista de exercícios. Apenas um professor utiliza esse instrumento para avaliar o quanto o aluno está aprendendo ou aprendeu na disciplina.

5 CONCLUSÕES

Este artigo apresentou o planejamento, execução e os resultados da execução de um *survey* que foi construído para caracterizar como teste de software é ensinado no Brasil. O processo de Kasunic (2005) [15], para construção de *survey* para área de engenharia de software e alguns direcionamento de Shull *et al.* (2008) [16] foram utilizados. Para apoiar a definição dos objetivos foi utilizado a abordagem GQM proposta por Basili (1992) [3]. Com auxílio da abordagem GQM, o objetivo principal foi fragmentado em objetivos específicos. Definiu-se algumas questões com base nos objetivos e também métricas.

Visando mitigar as possíveis ameaças à validade da pesquisa, foi realizado uma execução piloto do *survey* com especialistas da área. Com os dados dessa execução e *feedback* dos participantes, foi feita uma revisão do questionário em relação ao formato e quanto à formulação das perguntas. Além disso, visando mitigar essa ameaça à validade, foi construído um glossário que contém toda a terminologia sobre teste de software, mecanismos de apoio para o ensino de teste e abordagens. Esse artefato também foi analisado pelos especialistas durante a execução piloto. O artefato foi utilizado durante a distribuição do *survey*.

Outra possível limitação que reflete em possíveis ameaças à validade deste estudo está relacionada ao fato dos resultados do *survey* serem baseados em respostas subjetiva dos professores. É possível que os professores tenham respondido de forma a não representar a verdadeira maneira sobre como ensinam teste de software. O número limitado de professores que participaram do *survey* também pode ser considerada uma ameaça à validade. Foi possível obter um conjunto bom de respostas, mas seria importante uma maior participação de professores atuantes em todos os estados do país.

Como resultado, pôde-se observar que os principais níveis, técnicas e critérios de teste são ensinados. Quando teste é ensinado em disciplina específica, é possível abranger um número maior de técnicas e critérios. Dentre as técnicas, teste funcional e os critérios particionamento em classes de equivalência e análise do valor limite se destacam em relação ao número de professores que ensinam critérios de teste. Constatou-se também algumas dificuldades que os professores precisam enfrentar ao ensinar o conteúdo. Em relação à abordagem de ensino, foi revelado que maior parte dos professores ensinam teste usando método tradicional, com slides (mecanismo de apoio).

Os resultados também demonstram que os professores necessitam de mecanismos de apoio ao ensino de teste que contenham exemplos práticos. Desse modo, acredita-se que os atuais mecanismos de apoio não satisfazem o professor por eles não conseguirem abordar uma visão mais prática sobre teste.

A partir da apresentação do *survey* neste artigo, novas perspectivas sobre a pesquisa em educação em teste de software surgem, dentre eles: (1) investir na experimentação de novas abordagens de ensino; (2) estabelecer mecanismos de apoio que consigam contemplar atividades que simulam problemas reais; (3) definir estratégias

para o conteúdo de teste ser abordado em disciplina específica de teste em cursos de computação.

Uma nova perspectiva de pesquisa futura é reconhecer quais as ferramentas que apoiam a automatização da atividade de teste e que são utilizadas pelos professores. Essa investigação torna-se relevante porque, dentre as dificuldades relatadas pelos professores, as ferramentas para automatização recebem destaque. Outra sugestão é fazer a tradução do *survey* para o idioma inglês e distribuir, tentando reconhecer e caracterizar como teste de software é ensinado internacionalmente.

AGRADECIMENTOS

Os autores gostariam de agradecer ao CNPq, CAPES e a Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP – Processo 2017/10941-8 e Processo 2013/07375-0) pelo apoio financeiro.

REFERÊNCIAS

[1] E. F. Barbosa e J. C. Maldonado. 2011. Ima-cid: an integrated modeling approach for developing educational modules. *Journal of the Brazilian Computer Society*, 17, 4, 207–239.

[2] E. F. Barbosa, E. Y. Nakagawa e J. C. Maldonado. 2006. Towards the establishment of an ontology of software testing. Em 18th International Conference on Software Engineering & Knowledge Engineering.

[3] V. R. Basili. 1992. Software Modeling and Measurement: The Goal/Question/Metric Paradigm. Rel. téc. College Park, MD, USA. https://drum.lib.umd.edu/bitstream/handle/1903/7538/?sequence=1.

[4] F. B. V. Benitti. 2018. A methodology to define learning objects granularity: a case study in software testing. *Informatics in Education*, 17, 1, 1–20.

[5] F. B. V. Benitti e E. L. Albano. 2012. Teste de software: o que e como é ensinado? Em *Workshop sobre Educação em Computação (WEI)*. (Jul. de 2012), 01–10.

[6] W. O. Bussab e P. A. Morettin. 2013. *Estatística Básica*. Saraiva, São Paulo.

[7] P. J. Clarke, D. L. Davis, R. Chang-Lau e T. M. King. 2017. Impact of using tools in an undergraduate software testing course supported by wrestt. *ACM Trans. Comput. Educ.*, 17, 4, Article 18, 18:1–18:28.

[8] P. J. Clarke, D. Davis, T. M. King, J. Pava e E. L. Jones. 2014. Integrating testing into software engineering courses supported by a collaborative learning environment. *Trans. Comput. Educ.*, 14, 3, Article 18, 18:1–18:33.

[9] P. J. Clarke, J. Pava, Y. Wu e T. M. King. 2011. Collaborative web-based learning of testing tools in se courses. Em *42nd ACM Technical Symposium on Computer Science Education*, 147–152.

[10] B. S. Clegg, J. M. Rojas e G. Fraser. 2017. Teaching software testing concepts using a mutation testing game. Em *39th International Conference on Software Engineering: Software Engineering and Education Track*, 33–36.

[11] D. A. Cook e T. J. Beckman. 2006. Current concepts in validity and reliability for psychometric instruments: theory and application. *The American Journal of Medicine*, 119, 2, 166.e7–166.e16.

[12] M. E. Delamaro, J. C. Maldonado e M. Jino, editores. 2016. *Conceitos básicos. Introdução ao Teste de Software*. Elsevier, Rio de Janeiro, Brasil. Parte 1, 1–8.

[13] O. M. Ekwoge, A. Fontão e A. C. Dias-Neto. 2017. Tester experience: concept, issues and definition. Em *41st Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 01, 208–213.

[14] J. C. R. Hauck, C. G. von Wangenheim e Algo von Wangenheim. 2011. Método de Aquisição de Conhecimento para Customização de Modelos de Capacidade/Maturidade de Processos de Software. Rel. téc. Florianópolis, Santa Catarina, Brasil. https://goo.gl/4mEiTg.

[15] M. Kasunic. 2005. Designing an effective survey. Rel. téc. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

[16] F. Shull, J. Singer e D. I. K. Sjøberg, editores. 2008. *Personal opinion surveys. Guide to Advanced Empirical Software Engineering*. Springer London, London, 63–92.

[17] O. A. L. Lemos, F. F. Silveira, F. C. Ferrari e A. Garcia. 2017. The impact of software testing education on code reliability: an empirical assessment. *Journal of Systems and Software*.

[18] M. S. Litwin. 1995. *How to Measure Survey Reliability and Validity*. SAGE Publication, Thousand Oaks, California.

[19] L. N. Paschoal, L. R. Silva e S. R. S. Souza. 2017. Abordagem flipped classroom em comparação com o modelo tradicional de ensino: uma investigação empírica no âmbito de teste de software. Em *XXVIII Simpósio Brasileiro de Informática na Educação (SBIE 2017)*. (Nov. de 2017), 01–10.

[20] J. M. Rojas, T. D. White, B. S. Clegg e G. Fraser. 2017. Code defenders: crowdsourcing effective tests and subtle mutants with a mutation testing game. Em 39th International Conference on Software Engineering (ICSE), 677–688.

[21] T. G. Silva, F. M. Müller e G. Bernardi. 2011. Panorama do ensino de engenharia de software em cursos de graduação focado em teste de software: uma proposta de aprendizagem baseada em jogos. *RENOTE - Revista Novas Tecnologias na Educação*, 9, 2, 01–10.

[22] J. Smith, J. Tessler, E. Kramer e C. Lin. 2012. Using peer review to teach software testing. Em *Ninth Annual International Conference on International Computing Education Research*, 93–98.

[23] IEEE Computer Society, Pierre Bourque e Richard E. Fairley. 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. (3rd ed.). IEEE Computer Society Press, Los Alamitos, CA, USA. ISBN: 0769551661, 9780769551661.

[24] A. Soska, J. Mottok e C. Wolff. 2016. An experimental card game for software testing: development, design and evaluation of a physical card game to deepen the knowledge of students in academic software testing education. Em 7 th IEEE Global Engineering Education Conference (EDUCON), 576–584.

[25] D. M. Souza, S. Isotani e E. F. Barbosa. 2015. Teaching novice programmers using progtest. *International Journal of Knowledge and Learning*, 10, 1, 60–77.

[26] P. H. D. Valle, E. F. Barbosa e J. C. Maldonado. 2015. Cs curricula of the most relevant universities in brazil and abroad: perspective of software testing education. Em *2015 International Symposium on Computers in Education (SIIE)*, 62–68.

[27] P. H. Valle, E. F. Barbosa e J. C. Maldonado. 2015. Um mapeamento sistemático sobre ensino de teste de software. Em XXVI Simpósio Brasileiro de Informática na Educação (SBIE 2015), 71–80.

[28] W. E. Wong. 2012. Improving the state of undergraduate software testing education. Em *2012 ASEE Annual Conference & Exposition*.