CBIE 2023
CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO
Uma escola para o futuro: Tecnologia e conectividade a serviço da educação

Uma Abordagem para o Ensino de Testes de Software utilizando Metodologias Ativas em Cursos Superiores de Computação

Isaac Souza Elgrably[1], Sandro Ronaldo Bezerra Oliveira[1]

[1]Programa de Pós-Graduação em Ciência da Computação (PPGCC) – Instituto de Ciência Exatas e Naturais, Universidade Federal do Pará (UFPA) - Rua Augusto Corrêa, 01 – Guamá – Belém - PA - Brasil

isaacelgrably@hotmail.com, srbo@ufpa.br

Abstract. *Software testing is considered a key activity in the software development cycle as it helps computer professionals design good quality software. Thus, this is an essential activity for the software industry, although with all its nuances, learning at the academic level is still not given high priority. Considering that it is a content that needs a good amount of elements of previous knowledge and much of its knowledge needs practical interventions, the importance of using active methodologies was seen, which has been well accepted in more practical computing subjects. Thus, the main objective of this work is to present a teaching approach used with a curriculum and a teaching plan for a software testing subject, based on active methodologies. As a result, there is a teaching plan that is built from active methodologies, which is composed of 32 theoretical and practical classes, distributed in 4 modules, is based on references from a specialized curriculum in software testing, built from a mapping in different assets related to software testing, the main ones being the SBC Curricular Reference Guide, the ACM/IEEE Curriculum Guide and the SWEBOK Software Engineering Knowledge Guide. In addition, two quasi-experiments were carried out in full, the first evaluated the effectiveness of the test teaching approach compared to a traditional approach and the second, in addition to evaluating the effectiveness of the teaching approach against a traditional approach, also made an analysis of the efficiency of learning in the class. Both were analyzed with a statistical approach using the two-sided Student's t distribution, which showed a certain learning gain for the presented approach, being up to p-value = 000007915 in the best case.*

Resumo. *O teste de software é considerado uma atividade chave no ciclo de desenvolvimento de software, pois ajuda os profissionais de computação a projetar software de boa qualidade. Assim, esta é uma atividade essencial para a indústria de software, embora com todas as suas nuances ainda não seja dada alta prioridade ao aprendizado em nível acadêmico. Tendo em vista que é um conteúdo que precisa de uma boa quantidade de elementos de conhecimentos prévios e grande parte do seu conhecimento necessita de intervenções práticas, viu-se a importância de se utilizar as metodologias ativas, que vem tendo uma boa aceitação em disciplinas mais práticas de computação. Assim, o principal objetivo deste trabalho é apresentar uma abordagem de ensino utilizada com um currículo e um plano de ensino de uma disciplina de testes de software, elaborados com base em metodologias ativas. Como resultados tem-se um plano de ensino que é construído a partir de metodologias ativas, composto por 32 aulas teóricas e práticas, distribuídas em 4 módulos, baseado em referenciais de um currículo especializado em*

testes de software, e construído a partir de um mapeamento em diferentes ativos relacionados a testes de software, sendo os principais deles o Guia de Referencias Curriculares da SBC, o guia curricular da ACM/IEEE e o Guia dos conhecimentos de Engenharia de Software SWEBOK. Além disso, dois quase-experimentos foram executados por completo, o primeiro avaliou a efetividade da abordagem de ensino de testes em comparação com uma abordagem tradicional e o segundo além de avaliar a efetividade da abordagem de ensino perante uma abordagem tradicional também fez uma análise da eficiência da aprendizagem na turma. Ambos foram analisados com uma abordagem estatística utilizando a distribuição t de Student bicaudal, que apresentou um certo ganho de aprendizagem para a abordagem apresentada, sendo ele de até p-value = 000007915 no melhor caso.

1. Introdução

Por muito tempo a tarefa de teste de software foi considerada um "castigo" para os programadores. O teste era considerado uma tarefa ingrata, porque se esperava justamente que os desenvolvedores construíssem software de boa qualidade [Wazlawick, 2019]. Vincenzi *et al*. (2018) comentam que mais recentemente percebe-se uma conscientização de que a atividade de teste requer dedicação e disciplina para ser efetiva e eficiente, de forma a identificar o máximo de defeitos possível, contribuindo para a qualidade do produto final.

Nas últimas décadas observou-se um crescimento acentuado de pesquisadores e de desenvolvedores na área de teste de software, isso se deve ao fato da complexidade do software desenvolvido também ter crescido rapidamente, assim como a competição entre as empresas e a necessidade de desenvolver softwares com menos defeitos e mais qualidade [Vincenzi *et al*., 2018].

Analisando que as atividades ligadas a teste geralmente correspondem por 30 a 40% dos custos totais do projeto, apenas uma atenção limitada ainda lhe é dada [TMMi Foundation, 2018]. Assim, consolida-se uma necessidade para o melhor aproveitamento no aprendizado desse importante tópico da Engenharia de Software.

Para o pleno aprendizado de testes de software, além do conhecimento técnico, as habilidades sociais também devem ser ensinadas aos alunos [Dolezal *et al.*, 2019]. Vam Damn (2019) aponta que os testadores de software, além das habilidades técnicas, também devem ter habilidades sociais e Van Veenendaal (2019) afirma que os testadores devem ter habilidades pessoais para influenciar e se comunicar de forma que se tornem vitais para um projeto de software.

Há uma crescente necessidade de experimentar novos modelos pedagógicos para o ensino de teste de software, pois é necessário que o aluno desenvolva autonomia de ideias, pensamento crítico além de construir um bom conhecimento teórico-prático. A formação desse aluno deve alinhar o estado da arte com as tecnologias emergentes na área [Paschoal, 2019].

Também existe a necessidade de investir na formação acadêmica dos alunos de computação. Espera-se que eles incorporem conceitos e práticas de teste de software através do desenvolvimento de iniciativas oriundas da área [Paschoal, 2020].

As metodologias ativas, quando tomadas como base para o planejamento de situações de aprendizagem, podem contribuir significativamente para o desenvolvimento da autonomia e motivação do aluno, pois favorecem o sentimento de pertencimento e coparticipação no processo de aprendizagem [Diesel *et al.,* 2017]. O Guia Curricular da SBC (Sociedade Brasileira de Computação) 2017 indica que nos cursos de Ciência da Computação, sempre que possível, devem ser empregadas

metodologias ativas para que o aluno dedique mais tempo em atividades em que seja protagonista no processo de ensino e aprendizagem [SBC, 2017].

Garousi *et al.* (2020), em sua revisão da literatura, afirmam que a disciplina de testes de software precisa ser ensinada de forma mais ativa, utilizando aprendizagem de forma ativa, e prática para que os estudantes adquiram competências e possam estimular habilidades necessárias no mercado de trabalho.

Diante do exposto, tendo por base a realidade da oferta da disciplina Testes de Software na Universidade dos pesquisadores, assim como as inquietações da experiência do pesquisador autor da tese como aluno de um curso de graduação na área de TI, configuramos a seguinte questão-foco para a pesquisa: *Como usar metodologias ativas como abordagem para o processo de ensino e aprendizagem de conteúdos estudados na disciplina Testes de Software nos cursos de Tecnologia da Informação?*

Para responder a questão-foco, a pesquisa teve como objetivo apresentar um conjunto de elementos que visem auxiliar o ensino de testes de software. Um *curriculum* construído especificamente para o ensino de testes que levou em consideração diversos guias acadêmicos e uma revisão na literatura para sua construção [Elgrably e Oliveira, 2020a], no entanto sua principal contribuição é um plano de ensino utilizado especializado para testes de software. Sua elaboração foi realizada a partir dos principais guias de referências da literatura especializada e culminou em quatro unidades voltadas para o ensino de testes de software, sendo elas: engenharias de software, construção de software, qualidade e testes de software.

O *curriculum* desenvolvido foi apoiado por um plano de ensino [Elgrably e Oliveira, 2020b], que continha um conjunto de metodologias ativas de ensino com Aulas Expositivas Dialogadas, *Coding Dojo*, Aprendizagem Baseada em Problemas, Discussão de Caso Práticos, Sala de Aula Invertida e Jogos Sérios, que foram introduzidos nas unidades de ensino do *curriculum* utilizado. Visando alcançar um certo nível de validação do plano de ensino, foram realizados dois quase-experimentos que tinham como objetivo avaliar o funcionamento do plano de ensino e os ganhos de aprendizagem em relação a uma turma que recebeu o ensino de forma tradicional, porém neste artigo só serão tratados os resultados do segundo quase-experimento, que também foi realizado uma aferição da eficácia de aprendizagem utilizando pré-teste e pós-teste.

O artigo está organizado da seguinte maneira: a Seção 2 apresenta sucintamente a fundamentação teórica; a Seção 3 relata a metodologia seguida para o desenvolvimento da pesquisa; a Seção 4 discute e apresenta os resultados obtidos pela pesquisa; e, por fim, a Seção 5 descreve as considerações finais e os trabalhos futuros.

2. Fundamentação Teórica

Esta seção provê uma visão geral dos conceitos usados como base para o desenvolvimento da pesquisa.

2.1 Abordagem de Ensino para Testes

Pesquisas constatam que existe uma certa deficiência no ensino de alguns tópicos de computação que pode vir a ter uma relação com a maneira com que os professores ensinam o conteúdo de computação, pois ainda são poucas as iniciativas que utilizam metodologias de ensino diferentes da tradicional [Soska, Mottok e Wolff, 2016]. Devem ser estimulados esforços para a adoção de modelos pedagógicos que visam promover cenários que oportunizam a aprendizagem ativa aos alunos de teste de software [Lauvas e Arcuri, 2018].

Para que alguém consiga obter construção de novos conhecimentos, o processo de ensino-aprendizagem é determinante. Com uma abordagem de ensino mais centrada

no aluno o professor deixa de ser um transmissor de conteúdos e passa a orientar os alunos, mediando e estimulando o aprendizado [Hung, 2015].

Algumas iniciativas vêm sendo propostas para capacitar e motivar os profissionais de teste de software: módulos educacionais, teste por pares, TDD - *Test Driven Development*, ensino de teste com programação, jogos educacionais, entre outras [Valle *et al*., 2015].

Paschoal (2019) relata a necessidade em experimentar novos modelos pedagógicos no ensino de teste de software, em especial aqueles que buscam promover maiores condições para o aluno desenvolver autonomia, pensamento crítico e ao mesmo tempo que o aluno consiga ter uma boa formação teórico-prática, alinhando o estado da arte com as tecnologias emergentes da área.

Diversas abordagens de aprendizagem buscam suprir algumas problemáticas encontradas no ensino de testes, como: alunos com falta de experiência em testar adequadamente um software [Lemos *et al*., 2017]; falta de tarefas prática para mostrar aos alunos como os conceitos de testes devem ser aplicados [Scatalon *et al*., 2018; Lemos *et al*., 2017]; e estabelecer mecanismos de apoio que consigam contemplar atividades que simulam problemas reais [Paschoal e Souza, 2018].

Na pesquisa de Valle *et al*. (2015) foi observado que os cursos de graduação da área de computação, em geral, não proporcionam que os estudantes obtenham uma visão integrada dos conteúdos de teste de software com outras disciplinas dos cursos de graduação.

2.2 Métodos Instrucionais de Ensino e Metodologias Ativas

O Ministério da Educação Brasileiro [MEC, 2016] propõe que a metodologia de ensino para cursos de computação deve ser centrada no aluno como sujeito da aprendizagem e apoiada no professor como facilitador do processo de ensino-aprendizagem e que o professor deve mostrar, ainda, as aplicações dos conteúdos teóricos, ser um mediador, estimular a competição e a comunicação, provocar a realização de trabalho em equipe, motivar os alunos para os estudos, orientar o raciocínio e desenvolver as capacidades de comunicação e de negociação. Logo, o uso de metodologias ativas de ensino alinha-se às proposições mencionadas como métodos instrucionais para o ensino.

Uma metodologia ativa de ensino é uma estratégia que coloca um aluno no papel central do aprendizado, sendo ele o protagonista e o professor um mediador e auxiliador do processo de ensino-aprendizagem.

Nas metodologias de aprendizagem ativa busca-se o envolvimento dos alunos por meio de atividades ou discussões visando sempre a troca de conhecimento e enfatizando as experiências entre todos, geralmente com trabalho em grupo [Diesel *et al.*, 2018]. As metodologias usadas na abordagem proposta na tese foram: Aula Expositiva Dialogada, *Coding Dojo*, Discussão de Casos Práticos, Realização de Projetos Práticos, Ensino Lúdico, *Problem Based Learning* (PBL), Sala de Aula Invertida.

2.3 Trabalhos Relacionados

O trabalho desenvolvido por Bennitti (2018) tem como objetivo apresentar uma metodologia utilizando objetos de aprendizagem para ensinar diferentes disciplinas de computação, porém o estudo de caso do trabalho foi realizado em uma disciplina de testes de software. O trabalho utilizou estudos de caso para avaliar os possíveis ganhos de aprendizagem com a metodologia utilizada.

Um outro trabalho que propõe uma metodologia de ensino para a construção de uma disciplina de testes de software é o de Liu (2020), que mistura diferentes métodos de ensino on-line e off-line e possui 3 fases: na Fase 1 tem-se a aprendizagem; na Fase 2

acontece uma internalização e absorção do conhecimento dos conteúdos adquiridos na fase anterior; por fim, na fase 3 os alunos assistem a vídeos resumos para consolidar a capacidade de conhecimento e realizam provas on-line e lições de casa.

Outro trabalho sobre abordagem de ensino para disciplina de testes de software é de Enoiu (2020), que consiste em um projeto de curso sobre testes baseados em modelos em ensino remoto com contexto de ensino em práticas da indústria de software, no qual os alunos assistem palestras on-line antes das aulas e, posteriormente, fazem discussões em salas de aula virtual em grupo, além de técnicas pedagógicas como trabalho em grupo, apresentações de alunos e discussões.

A análise de cada um desses trabalhos relacionados possibilitou observar o uso de diferentes metodologias e construções de ensino-aprendizagem utilizando diferentes estratégias para o ensino de testes de software ou outro tópico da Engenharia de Software. Também foi notada a recente visibilidade e preocupação com o ensino da disciplina de testes de software, como mencionado no trabalho de Garousi *et al*. (2020). No entanto, a abordagem proposta neste trabalho diferencia-se por utilizar também da dimensão de conhecimento da taxonomia revisada de Bloom (Anderson e Krathwohl, 2001), correlacionando os conteúdos com as metodologias ativas de ensino para alcançar os níveis de aprendizagem e promovendo os trabalhos em equipes para que os alunos tenham desafios mais próximos de situações reais da indústria de software, como indicado pela ACM/IEEE (2017) e SBC (2017).

3. Da Concepção à Validação da Abordagem de Ensino

Durante a pesquisa inicial para o desenvolvimento desse trabalho, foi definido um fluxo metodológico, presente na Figura 1 e detalhado em seguida.

3.1 Concepção da Abordagem de Ensino

A concepção da abordagem de ensino perpassou por vários procedimentos metodológicos, descritos a seguir.

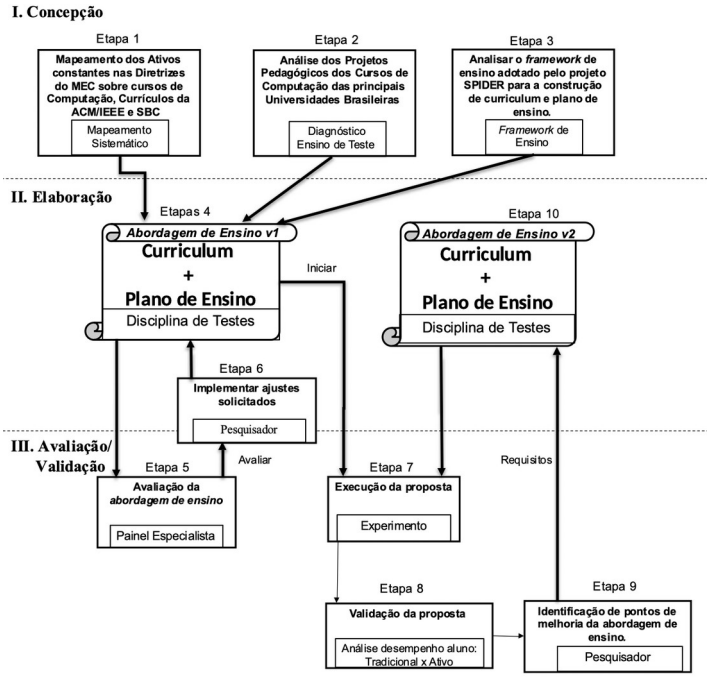

Figura 1. Metodologia da pesquisa.

3.1.1 Mapeamento Sistemático

Para atingir os objetivos desta pesquisa, inicialmente, foi realizada uma análise nas Diretrizes Curriculares Nacionais dos Cursos de Graduação em Computação, aprovadas

em 2016 [MEC, 2016] para identificar os tópicos e as competências importantes para Teste de Software. Em seguida, foi realizado um mapeamento entre os diversos ativos para estabelecer uma base de conhecimento e servir de referência para o desenvolvimento da abordagem de ensino. Os demais ativos são os conteúdos programáticos [ACM/IEEE, 217; ACM/IEEE, 2020; SBC, 2017] dos cursos de graduação em Tecnologia da Informação e suas práticas para Teste de Software.

Posteriormente, foi analisado o conjunto de práticas do SWEBOK (*Software Engineering Body of Knowledge*) [Bourque e Fairley, 2014], que é um importante guia de conhecimento sobre as áreas de conhecimento da Engenharia de Software, e as áreas de processos indicadas pelo TMMi (*Test Maturity Model Integration*) [TMMi Foundation, 2018], que é um modelo que abrange todas as atividades relacionadas à qualidade de um produto de software com foco no teste realizado e, por fim, foram incluídos os valores presentes no Manifesto de Testes Ágeis [Laing e Greaves, 2015].

Assim, conforme mencionado, foram identificadas competências importantes para o aprendizado de Teste de Software. Após a seleção das competências do MEC (Ministério da Educação e Cultura), foi realizado o mapeamento com os diversos guias, definidos anteriormente, para estabelecer uma base de conhecimento para a construção do curriculum de teste de software. Assim, para cada um dos guias foi correlacionada uma das Competências do MEC e seus ativos. Assim, as Competências do MEC ajudaram a extrair o conteúdo de todos os guias analisados para montar a proposta de construção do curriculum de ensino aderente a Teste de Software.

O resultado extraído do alinhamento serviu de referência para orientar as unidades de ensino e as áreas que devem ser abordadas nos cursos de graduação em computação, porém não necessariamente deixam claro quais conteúdos curriculares serão trabalhados, eles apenas apontam o mínimo do que deve ser ensinado. Após essa etapa, um conjunto de insumos foi extraído do alinhamento entre a competência do MEC e cada um dos guias. Um insumo é um elemento essencial para fazer um determinado produto ou serviço. Nesse mapeamento, o conjunto de resultados que foram considerados foram: produtos de trabalho de saída e entrada e, principalmente, o conteúdo gerado a partir de cada um dos guias que estavam aderentes às competências destacadas do MEC. Como resultado desse mapeamento, chegou-se a um total de 75 insumos. Posteriormente, os insumos foram priorizados por meio de um conjunto de critérios selecionados pelos pesquisadores: Simplicidade, Cultura Ágil e *Open Source*, Alinhamento Estratégico e Comparação de Benefícios. Após a análise dos critérios, um total de 40 insumos auxiliaram na construção da curriculum. Mais informações sobre esta encontram-se em [Elgrably e Oliveira, 2019].

3.1.2 Diagnóstico sobre Ensino de Teste

Para averiguar a importância e adequação das instituições nacionais sobre o tópico de testes de software, foi realizado um diagnóstico nas principais instituições brasileiras, que forneceu importantes inferências sobre como o teste de software tem sido abordado no nível curricular, desde assuntos específicos de teste de software, assuntos que contêm tópicos relacionados a provas e quais são os principais tópicos que são ensinados.

A seleção das instituições analisadas vem do RUF – *Ranking* Universitário da Folha de São Paulo de 2019, último ano em que a pesquisa foi realizada devido à pandemia de Covid-19. Além da colocação no *ranking*, outros três critérios também foram levados em consideração, que foram: a conformidade com a nota atribuída pelo MEC; a presença de cursos de computação regulamentados no Brasil; e a acessibilidade às matrizes curriculares dos cursos. Com intenção de ampliar os resultados averiguados em uma pesquisa base [Valle, Barbosa e Maldonado, 2015], foram selecionadas 32 instituições de ensino brasileiras, inicialmente, porém apenas 28 possuíam todos os

critérios estabelecidos. Assim, um total de 28 currículos, ementários e planos pedagógicos de cada uma das instituições foram coletados.

A partir dos resultados do diagnóstico foi verificado que em todas as instituições foram encontradas disciplinas que abrangem conteúdos de testes de software. A principal forma de abordagem das instituições brasileiras sobre conteúdos de testes de software dá-se a partir das disciplinas de Engenharia de Software e Qualidade de Software. Nesse caso, geralmente, apenas uma parte da disciplina é utilizada para apresentação de conteúdo sobre testes de software [Elgrably e Oliveira, 2021a].

Um outro resultado obtido no diagnóstico foi que das 28 instituições verificadas em 68% delas já haviam disciplinas específicas para o ensino de testes de software e 32% ainda não tinham essa especificidade. Além disso, outro resultado coletado no diagnóstico é que não existe um consenso de conteúdos, nem mesmo de nome de disciplina para testes de software. Assim, percebeu-se o uso de diferentes nomes de disciplinas para testes de software encontradas nas instituições consideradas no diagnóstico, a saber: Teste de Software, Verificação e Validação, entre outros.

Também averiguou-se haver uma tendência nas instituições sobre mesclar o conteúdo de Testes de Software com outros tópicos da Engenharia de Software, como Medição e Inspeção de Software. Os resultados obtidos com a realização deste diagnóstico podem ser vistos em [Elgrably e Oliveira, 2021a].

3.1.3 Framework de Ensino do Projeto SPIDER

O projeto SIDER (2009) conta com uma linha de pesquisa voltada para *framework* de ensino (estrutura que compõe curriculum e plano de ensino) em Engenharia de Software na UFPA, nos dois cursos oferecidos pela Faculdade de Computação (CBCC e CBSI) e no Programa de Pós-graduação em Ciência da Computação (PPGCC). O *framework* de ensino, em desenvolvimento pelos membros dessa linha de pesquisa do projeto, já é objeto de inúmeras publicações [Furtado e Oliveira, 2018; Garcia, Carvalho e Oliveira, 2021; Quaresma e Oliveira, 2022; Costa e Oliveira, 2022]. Durante a análise do *framework* adotado pelo projeto, buscou-se compreender a estrutura básica da abordagem e os formatos usados para definir o curriculum e plano de ensino em diferentes áreas da Engenharia de Software.

3.2 Elaboração da Abordagem

Esta fase contempla as seguintes etapas da Figura 1: (i) 4, que refere-se à triangulação dos dados advindos do mapeamento sistemático, do diagnóstico de ensino de teste no Brasil e da análise da abordagem de ensino do projeto SPIDER, para gerar a primeira versão do curriculum e plano de ensino; (ii) 6, a implementação dos ajustes na abordagem de ensino após o Painel de Especialista; e (iii) 10, que trata da versão final da abordagem de ensino com a implementação de mudanças oriundas da avaliação e validação, confirmadas após a realização do quase-experimento. O relato completo da composição da abordagem de ensino (curriculum e plano de ensino) encontra-se em [Elgrably e Oliveira, 2020a; Elgrably e Oliveira, 2020b].

4. Avaliação e Validação do Abordagem de Ensino

No processo de avaliação e validação da abordagem de ensino foi implementado o Painel de Especialistas, com uso de Grupo Focal, e foram realizados dois quase-experimentos para testar, em situação real de aula, o curriculum e o plano de ensino.

4.1 Painel de Especialista

A avaliação da documentação presente na abordagem de ensino de teste de software foi realizada por meio de um Grupo Focal, que é um método de pesquisa que estimula os participantes a refletirem sobre os diversos aspectos de um tema analisado e tem como principal intenção aprimorar o entendimento de como as pessoas sentem-se ou pensam

sobre o tópico destacado. Para esta pesquisa, os critérios de seleção dos especialistas participantes foram: possuir experiência docente e profissional na área de Engenharia de Software; possuir título de doutor; possuir publicações de pesquisas relacionadas ao ensino de alguma área da Computação; ter participado da elaboração ou atualização do projeto pedagógico de sua instituição; e ter adotado alguma abordagem prática de ensino. Assim, três professores de universidades públicas federais brasileiras, que possuem os requisitos descritos, foram selecionados para o painel de especialistas.

Em princípio, houve uma análise e avaliação individual da abordagem de ensino, por meio de questionário eletrônico, contendo critérios relacionados à documentação e usabilidade da abordagem de ensino. Os especialistas consideraram a abordagem adequada para o ensino de teste de software, pois o curriculum tinha uma instância de uso e uma ementa acadêmica no mesmo documento. A maioria dos especialistas considerou que a abordagem apresentada possui um plano de ensino com detalhamento suficiente para fornecer uma base de aprendizado para teste de software. Por fim, os especialistas consideraram a abordagem sem margem de adequação para outros tópicos de computação que não sejam testes de software.

Em um segundo momento de avaliação, houve uma reunião de consenso entre as partes via videoconferência, onde os especialistas apresentaram sugestões para enriquecer o modelo, tais como: não existe uma biblioteca de ativos a ser utilizada na adoção da abordagem; a abordagem é fácil para usar apenas para o assunto apresentado; a abordagem parece extremamente prescritiva e sua adaptação a diferentes ambientes de ensino não parece trivial; e é necessário algum esforço para preparar os projetos e materiais sugeridos. Todas essas sugestões foram consideradas ao criar a próxima versão da abordagem. Os relatos dos resultados apurados do Painel de Especialistas encontram-se em [Elgrably e Oliveira, 2020a; Elgrably e Oliveira, 2020b].

4.2 Quase-Experimentos

O quase-experimento com o uso da abordagem de ensino foi uma das formas encontradas para avaliar e validar nossa proposta e partiu da seguinte questão norteadora: *Qual a eficácia do aprendizado dos alunos quando a abordagem ativa de Testes de Software proposta na tese é adotada sobre a abordagem tradicional no nível Aplicar da Taxonomia Revisada de Bloom?*

Os itens do planejamento do quase-experimento são: (i) **cenário**, as aulas ocorreram de forma remota (por conta da pandemia da COVID-19) a partir do uso do Google Meet, contendo material audiovisual e computadores para o docente e discentes, bem como toda a infraestrutura (ferramentas de software) necessária para a aplicação das práticas; (ii) **personagens**, retratado por uma turma de pós-graduação do PPGCC da UFPA com 43 discentes (divididos em um grupo experimental e outro de controle), dois monitores e o professor da disciplina, responsável pela abordagem de ensino; (iii) **desenvolvimento das aulas**, conforme o plano de ensino, o quase-experimento contabilizou 32 aulas com duração de até 2 horas cada uma (Aula 01, destinada a uma conversa inicial sobre a abordagem de ensino, disponibilização do material de apoio e realização de um pré-teste; Aulas 02 a 07, aulas expositivas dialogadas, aplicação de listas de exercício e da técnica de Coding Dojo do tipo Randori; Aulas 08 a 13, aulas expositivas dialogadas e aplicação da técnica de Coding Dojo do tipo Kake; Aulas 14 a 19, aulas expositivas dialogadas, aplicação de sala de aula invertida e de projetos práticos usando PBL; Aulas 20 a 32, aulas expositivas dialogadas, discussão de casos práticos, aplicação de sala de aula invertida, de Coding Dojo do tipo Kake e de de projetos práticos usando PBL, e realização de pós-teste e de uma coleta de *feedback* dos discentes sobre a abordagem de ensino; e (iv) ***design* da abordagem**, o conteúdo abordado foi o presente no curriculum definido para a abordagem de ensino, os alunos

tiveram acesso aos materiais de apoio para estudar os conteúdos ministrados, e o cronograma do plano de ensino foi modificado com a inserção de um pré-teste e pós-teste. Neste estudo, como mencionado na questão-foco, o objetivo era avaliar a eficácia da aplicação dos conteúdos de Testes de Software a partir dos conteúdos definidos na abordagem de ensino a partir do uso de metodologias ativas.

Os relatos completos dos quase-experimentos (planejamento e resultados dos quase-experimentos) encontram-se em [Elgrably e Oliveira, 2021a; Elgrably e Oliveira, 2022].

4.3 Validação e Discussão do Quase-Experimento

Para avaliar o uso da abordagem de ensino, utilizamos uma combinação das abordagens de cunho qualitativo e quantitativo, conforme descrevemos a seguir. Os resultados obtidos a partir da análise das hipóteses sugerem que esta abordagem tem uma eficácia de aprendizagem superior à eficácia obtida utilizando uma metodologia de aulas tradicionais, visto que as notas médias obtidas pelo grupo experimental durante as avaliações foram significativamente maiores do que as grupo controle.

Esses resultados podem ser atribuídos ao fato de que as metodologias ativas de ensino utilizadas no grupo experimental estarem fortemente focadas nos alunos, na realização de atividades práticas e colaborativas. Perante o exposto, os resultados obtidos assemelham-se ao de diferentes autores que utilizaram abordagens e estratégias mais centradas no aluno [Furtado *et al.*, 2021; Portela *et al.*, 2021], utilizando metodologias ativas de ensino [Fonseca e Mattar, 2017] e utilizando o aprendizado baseado em competências [Benniti, 2018; Yoshioka e Ishitani, 2018].

O fato dos resultados de algumas hipóteses no grupo experimental serem bem maiores, pode ser que em algumas unidades de ensino do curriculum todas as atividades eram em equipe e eram atividades práticas e, como o quase-experimento foi realizado em uma disciplina de Testes de Software de um programa de pós-graduação, possivelmente os alunos tinham experiência profissional no mercado de trabalho ou em atividades relacionadas a grupos de pesquisa do programa. Entretanto, esta mesma característica também se enquadrava aos alunos do grupo de controle.

Nessa aplicação do plano de ensino foi realizada uma análise da eficiência de aprendizagem da abordagem de ensino, onde os alunos deveriam, durante o período de uma aula, responder e enviar um teste para medir o conhecimento geral sobre testes de software. Esse mesmo teste foi passado no primeiro e no último dia de disciplina e os alunos foram informados que deveriam respondê-los sem consulta a nenhum material e que os resultados não eram avaliativos, mas seria importante o pleno desempenho no desafio. A partir dos resultados extraídos dos testes foi possível analisar a hipótese, pois temos diferenças significativas nos resultados obtidos entre o pré-teste e pós-teste, mostrando que a abordagem de ensino de testes foi capaz de acrescentar conhecimento de testes de software no grupo experimental.

Os alunos de ambos os grupos consideraram o conteúdo existente no *curriculum* extremamente relevante e suficiente para o aprendizado sobre testes de software. Os alunos do grupo experimental avaliaram de forma positiva as atividades práticas via metodologias ativas de ensino, mesmo elas sendo realizadas via ensino remoto.

O grupo controle teve uma grande carga horária de aulas teóricas e uma forma de avaliação de provas ao final de cada unidade de ensino. Assim, é possível que os alunos tenham tido dificuldades em aprender certos conteúdos apenas com aulas expositivas clássicas e com poucas possibilidades de colaboração e de trocas de conhecimentos entre os alunos. Isso pode ter ocasionado em uma pontuação consideravelmente inferior àqueles que estavam no grupo experimental.

Durante o quase-experimento realizado com o grupo experimental, foi possível observar algumas fragilidades presentes na abordagem apresentada nesse trabalho. A abordagem de ensino, mesmo com um quantitativo considerável de atividades práticas, ainda se preocupava em utilizar aulas expositivas dialogadas para ministrar o conteúdo estabelecido no *curriculum*, no entanto os alunos consideraram que houve um número elevado de aulas expositivas e sugeriram que deveria ter havido mais atividades de c*oding dojos*, projetos práticos e sala de aula invertida.

Uma outra fragilidade encontrada na abordagem diz respeito ao tempo disponível para a realização das atividades práticas, principalmente utilizando o ensino remoto. Portanto, uma melhor adequação e revisão das atividades de aprendizagem poderia causar uma melhor experiência para os alunos, o que possibilitaria a realização de qualquer uma das atividades dentro do prazo estipulado. A quantidade de conteúdo presente no *curriculum* também foi uma possível fragilidade, pois para desenvolver todo o seu conteúdo foi necessário um elevado quantitativo de aulas, necessitando que os alunos realizassem algumas atividades em horários fora de aula.

As atividades com melhores resultados foram as atividades em grupo, no entanto a atividade de *coding dojo randori* realizado com todos os alunos do grupo experimental foi o que houve maior dificuldade de execução pelos alunos, em razão do grande número de alunos e a pouca intimidade destes com o conteúdo e com a ferramenta, o que acabou prejudicando a pontuação.

5. Considerações Finais

Os resultados de cada etapa da pesquisa possibilitaram responder a questão-foco, bem como atingir o objetivo geral traçado. Dentre os principais resultados alcançados, destacamos: (i) maior participação dos alunos nas aulas; (ii) maior colaboração entre os alunos durante a resolução dos desafios propostos no plano de ensino; (iii) maior engajamento para a resolução dos projetos práticos; (v) melhora no desempenho da maior parte dos alunos no que diz respeito à nota; (vi) emprego de práticas dinâmicas em sala de aula; (vii) *feedback* em tempo hábil do progresso dos alunos na disciplina; e (viii) aceitação da abordagem por parte dos alunos.

A abordagem de ensino como proposta de produto dessa tese, após todas as alterações, mostrou-se adequada e oportuna para mitigar problemas de ensino e aprendizado na disciplina Testes de Software. Entretanto, também possui limitações percebidas, principalmente, durante os quase-experimentos realizados em disciplinas Testes de Software no PPGCC da UFPA. Algumas fragilidades encontradas não foram completamente resolvidas, principalmente, as relativas a tempo e as ferramentas utilizadas no ensino remoto.

Esta pesquisa de doutorado, dada a sua abrangência e relevância possui condições de originar novos trabalhos, tais como: (i) realização de experimentos em turmas presenciais, podendo gerar informações para serem comparadas com as intervenções remotas para verificar possíveis diferenças da forma que a abordagem de ensino de testes de software comporta-se; (ii) utilização de apenas algumas das Unidades de Ensino presentes no curriculum para serem utilizadas como parte de uma outra disciplina, podendo assim haver a expansão de conteúdo gerado; (iii) desenvolvimento de um jogo sério para estimular o aprendizado de testes de software a partir de um jogo de tabuleiro, que trará um desafio épico a ser completado por um grupo de alunos utilizando um *chatbot*, que indica as ações a serem tomadas pelos alunos dependendo das respostas e resoluções que eles tomarem a partir dos desafios encontrados no tabuleiro.

Referências

ACM/IEEE (2017). “Information Technology Curricula: *Curriculum* Guidelines for Baccalaureate Degree Programs in Information Technology”. A Report in the Computing Curricula Series Task Group on Information Technology Curricula. ACM.

ACM/IEEE (2020). “Computing Curricula 2020 – CC2020”. Paradigms for Global Computing Education. ACM.

Anderson, L.W., Krathwohl, D. R. (2001). “A Taxonomy for Learning Teaching and Assesing: A Revision of Bloom's Taxonomy of Educational Objectives”. Longman.

Benitti, F. (2018). “A Methodology to Define Learning Objects Granularity: A Case Study in Software Testing”. Informatics in Education, V. 17, No. 1, pp. 1–20.

Bourque, P., Fairley, R. E. (2014). “SWEBOK Guide V3.0”. Available: www.swebok.org.

Costa, I. E. F., Oliveira, S. R. B. (2022). “A Syllabus to Support Teaching and Learning of Exploratory Test Design and Execution”. In Proceedings of the 14th International Conference on Computer Supported Education (CSEDU 2022), Virtual.

Diesel, A., Santos Baldez, A. L., Neumann Martins, S. (2017). “Os princípios das metodologias ativas de ensino: uma abordagem teórica”. Revista Thema, v. 14, n. 1, p. 268–288.

Dolezal, D., Posekany, A., Vittori, L., Koppensteiner, G., Motschnig, R. (2019). “Fostering 21st Century Skills in Student-Centered Engineering Education at the Secondary School Level: Second Evaluation of The Learning Office”. 2019 IEEE Frontiers in Education Conference (FIE).

Elgrably, I. S., Oliveira, S. R. B. (2019). “Uma Proposta de Ensino ou Aplicação de Testes com Foco em Métodos Ágeis feita através de um Mapeamento de Ativos”. 16th International Conference on Information Systems & Technology Management.

Elgrably, I. S., Oliveira, S. R. B. (2020a). “Construction of a *curriculum* adhering to the Teaching of Software Tests Using elements of Agile Context”. 2020 IEEE Frontiers in Education Conference (FIE).

Elgrably, I. S., Oliveira, S. R. B. (2020b). “Model for Teaching and Training Software Testing in an Agile Context”. 2020 IEEE Frontiers in Education Conference (FIE).

Elgrably, I. S., Oliveira, S. R. B. (2021a). “A Diagnosis on Software Testing Education In The Brazilian Universities”. 2021 IEEE Frontiers in Education Conference (FIE).

Elgrably, I. S., Oliveira, S. R. B. (2021b). “Remote teaching and learning of software testing using active methodologies in the covid-19 pandemic context”. 2021 IEEE Frontiers in Education Conference (FIE).

Elgrably, I. S., Oliveira, S. R. B. (2022). “A Quasi-Experimental Evaluation of Teaching Software Testing in Software Quality Assurance Subject during a Post-Graduate Computer Science Course”. *International Journal of Emerging Technologies in Learning (iJET)*, *17*(05), pp. 57–86. https://doi.org/10.3991/ijet.v17i05.25673.

Enoiu, E. P. (2020). “Teaching Software Testing to Industrial Practitioners Using Distance and Web-Based Learning. Frontiers in Software Engineering Education”. Springer International Publishing.

Fonseca, S. M., Mattar, J. (2020). “Metodologias ativas aplicadas à educação a distância: revisão de literatura”. Revista Educação a Distância e Práticas Educativas Comunicacionais e Interculturais, São Cristóvão-SE, v. 17, n. 2, p. 185-197, 2017.

Furtado, J. C. C., Oliveira, S. R. B. (2018). “A Methodology to Teaching Statistical Process Control in Computer Courses”. 13th International Conference on Evaluation of Novel Approaches to Software Engineering, Portugal.

Furtado, J. C. C., Oliveira, S. R. B., Chaves, R. O., Colares, A. (2021). "An Experimental Evaluation of a Teaching Approach for Statistical Process Control in Computer Courses". International Journal of Information and Communication Technology Education. IGI Global.

Garcia, F. W. S, Carvalho, E. C., Oliveira, S. R. B. (2021). "Use of active methodologies for the development of a teaching plan for the algorithms subject". 51th Annual Frontiers in Education – FIE'21. Lincoln, Nebraska - USA.

Garousi, V., Rainer, A., Lauvas, P., Arcuri, A. (2020). "Software-Testing Education: A Systematic Literature Mapping". In Journal of Systems and Software. (vol. 165, p. 110570). Elsevier.

Hung, H. T. (2015). "Flipping the *Classroom* for English Language Learners to Foster Active Learning". Computer Assisted Language Learning, V. 28.

Laing, S., Greaves, K. (2015). "The Testing Manifesto". Available: http://www.growingagile.co.za/2015/04/the-testing-manifesto.

Lauvas, P., Arcuri, A. (2018). "Recent Trends in Software Testing Education: A Systematic Literature Review". The Norwegian Conference on Didactics in it Education.

Lemos, O. A. L., Silveira, F. F., Ferrari, F. C., Garcia, A. (2017). "The Impact of Software Testing Education on Code Reliability: An Empirical Assessment". Journal of Systems and Software.

Liu, A. (2020). "Design of Blending Teaching Mode for Software Testing Course". 15th International Conference on Computer Science & Education (ICCSE). IEEE.

MEC (2016) "Diretrizes Curriculares Nacionais para os Cursos de Graduação em Computação (DCN16)". Brasil.

Paschoal, L. N. (2019). "Contribuições ao Ensino de Teste De Software com o Modelo Flipped *Classroom* e um Agente Conversacional". Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP.

Paschoal, L.N., Souza, S. R. S. (2018). "A Survey on Software Testing Education in Brazil". 17th Brazilian Symposium on Software Quality. New York. USA. ACM.

Paschoal, L. N., Oliveira, M. M., Melo, S. M., Barbosa, E. F., Souza, S. R. S. (2020). "Evaluating the impact of Software Testing Education through the Flipped *Classroom* Model in deriving Test Requirements". Proceedings of the 34th Brazilian Symposium on Software Engineering.

Portela, C., Vasconcelos, A., Oliveira, S., Souza, M. (2021). "An Empirical Study on the Use of Student-Focused Approaches in the Software Engineering Teaching". Informatics in Education.

Quaresma, J. A. S., Oliveira, S. R. B. (2022) "A Syllabus Proposal for Teaching of Software Development Process in Undergraduate Courses in Computer Science". In Proceedings of the XXXVI Brazilian Symposium on Software Engineering, Virtual.

Scatalon, L. P., Fioravanti. M. L., Prates. J. M., Garcia. R. E., Barbosa. E. F. (2018). "A Survey on Graduates' *Curriculum*-Based Knowledge Gaps in Software Testing". IEEE Frontiers in Education Conference (FIE).

SBC (2017). "Referências de Formação para os Cursos de Graduação em Computação". Brasil.

Soska, A., Mottok, J., Wolff, C. (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". 7TH IEEE Global Engineering Abu Dhabi, United Arab Emirates.

SPIDER (2009). “Projeto SPIDER - Tool Suite for Quality”. UFPA. Disponível em: http://www.spider.ufpa.br/.

TMMI Foundation (2018), “Test Maturity Model Integration – TMMi Release 1.0”.

Valle, P. H. D., Barbosa, E. F., Maldonado, J. C. (2015). “Cs Curricula of the Most Relevant Universities in Brazil and Abroad: Perspective of Software Testing Education”. XVII International Symposium on Computers in Education (SIIE). IEEE.

Van Dam, K. (2019). “The Future of Testing. The Future of Software Quality Assurance”. Springer International Publishing. 197–205.

Van Veenendaal, E. (2019). “Next-Generation Software Testers: Broaden or Specialize! The Future of Software Quality Assurance”. Springer International Publishing. 229–243.

Vincenzi, A. M. R., Delamaro, M. E., Neto, A. C. D., Fabbri, S. C. P. F., Jino, M., Maldonado, J. C. (2018). “Automatização de Teste de Software com Ferramentas de Software Livre”. Rio De Janeiro. Elsevier.

Wazlawick, R. S. (2019). “Engenharia De Software Conceitos E Práticas”. 2. ED. Rio De Janeiro. Elsevier.

Yoshioka, S. R. I., Ishitani, L. (2018). “An Adaptive Test Analysis Based on Students’ Motivation. Informatics in Education”. Vilnius University Press.