Revista Eletrônica de Iniciação Científica em Computação, 2026, 24:1
ISSN: 3085-8461 • doi: 10.5753/reic.2026.7750

Artigo de Pesquisa/Research Paper

Uma Avaliação Experimental do Uso de BDD na Automação de Testes de Segurança

An Experimental Evaluation of the Use of BDD in Security Test Automation

Samuel Sousa Teles [Universidade Estadual do Ceará | *samuel.teles@aluno.uece.br*]
Rubens Abraão da Silva Sousa [Universidade Estadual do Ceará |*abraao.sousa@aluno.uece.br*]
Ismayle de Sousa Santos [Universidade Estadual do Ceará |*ismayle.santos@uece.br*]

Universidade Estadual do Ceará, Av. Dr. Silas Munguba, 1700, Campus do Itaperi, Fortaleza, CE, 60741-000, Brasil.

Resumo. Atualmente, a automação de testes de segurança tem se tornado essencial no desenvolvimento de software, especialmente em ambientes ágeis que demandam qualidade contínua e rápida identificação de vulnerabilidades. Entretanto, diferenças na estrutura e na forma de especificação dos testes podem impactar sua compreensão e aplicação por testadores iniciantes. O objetivo deste estudo é investigar e comparar o uso do *Pytest* e do *Robot Framework* com *Behavior-Driven Development* (*BDD*) quanto ao impacto na carga cognitiva e na compreensão de testes de segurança. Para isso, foi realizado um experimento controlado com participantes noviços, avaliando carga cognitiva, demanda temporal percebida e acurácia na interpretação dos testes. Os resultados indicam que o *Robot Framework* com *BDD* apresentou menores níveis de demanda mental, esforço e frustração quando comparado ao *Pytest*, com diferenças estatisticamente significativas nessas dimensões da escala *NASA-TLX*. Em relação à compreensão dos testes, os resultados descritivos apontaram uma tendência favorável à abordagem baseada em *BDD*, especialmente na identificação de vulnerabilidades e na interpretação dos critérios de aprovação, embora essa diferença não tenha sido confirmada estatisticamente. Os resultados sugerem que o uso de *BDD* pode contribuir para reduzir a carga cognitiva percebida durante a análise de testes de segurança por participantes iniciantes.

Abstract. Currently, security test automation has become essential in software development, especially in agile environments that demand continuous quality assurance and rapid vulnerability detection. However, differences in the structure and specification of automated tests may affect their comprehension and application by novice testers. This study investigates and compares the use of *Pytest* and *Robot Framework* with *Behavior-Driven Development* (*BDD*) regarding their impact on cognitive workload and security test comprehension. To this end, a controlled experiment was conducted with novice participants, evaluating cognitive workload, perceived temporal demand, and accuracy in the interpretation of security tests. The results indicate that *Robot Framework* with *BDD* presented lower levels of mental demand, effort, and frustration when compared to *Pytest*, with statistically significant differences observed in these dimensions of the *NASA-TLX* scale. Regarding test comprehension, descriptive results suggested a favorable trend for the *BDD*-based approach, particularly in vulnerability identification and interpretation of test success criteria, although this difference was not statistically significant. Overall, the findings suggest that the use of *BDD* may help reduce the perceived cognitive workload associated with analyzing security tests performed by novice testers.

Palavras-chave: Testes de segurança, Behavior-Driven Development, PyTest, Carga Cognitiva, Experimento controlado.
Keywords: Security testing, Behavior-Driven Development, PyTest, Cognitive Load, Controlled Experiment.

Recebido/Received: 06 March 2026 • **Aceito/Accepted:** 31 July 2026 • **Publicado/Published:** 14 August 2026

1 Introdução

No contexto de desenvolvimento de software, os testes de segurança são essenciais para proteger dados e reduzir custos com falhas. A identificação manual de vulnerabilidades torna-se limitada em ambientes ágeis com integração contínua, exigindo a automação dos testes de segurança para garantir maior cobertura, escalabilidade e detecção precoce de falhas ao longo do ciclo de desenvolvimento [Abdiukov, 2024]. Nesse sentido, Zhang *et al*. [2024] ressaltam que métodos automatizados de testes, como *fuzzy*, *pentest* e análises estática de código e dinâmica, melhoram significativamente a eficiência da detecção de bugs e aprimoramento na qualidade, segurança e suporte durante o desenvolvimento.

Ainda assim, testes de segurança automatizados apresentam desafios relacionados à sua compreensão, interpretação e aplicação, devido à complexidade das ferramentas, à ocorrência de falsos positivos e à necessidade de conhecimentos técnicos especializados, especialmente entre profissionais menos experientes [Aloraini *et al*., 2019; Rajapakse *et al*., 2022]. Diferenças na forma de estruturar e expressar os testes podem aumentar o tempo necessário para análise, gerar erros na identificação de vulnerabilidades e comprometer a eficácia da automação de segurança [Qusef *et al*., 2019].

Conforme Murazvu *et al*. [2024], a falta de expertise, tempo, recursos e organização pode dificultar a adoção de testes automatizados, comprometendo sua eficácia na prática, inclusive em testes de segurança. Além disso, a seleção equivocada de métodos ou instrumentos pode levar à identificação parcial de falhas, ao aumento de resultados imprecisos e à necessidade de refazer o trabalho, colocando em risco a segurança do software e a credibilidade dos testes.

Ferramentas de análise de código estática, como So-

narQube[1] podem detectar vulnerabilidades de segurança em softwares de forma eficaz, mas certos tipos de alertas, como de injeção de comando, controle de processo e tratamento de exceções, podem ter uma maior propensão a falsos positivos, o que requer incorporar a necessidade de revisão constante e manutenção do processo para garantir a qualidade na detecção de vulnerabilidades [Aloraini *et al*., 2019].

Abordagens tradicionais de automação de testes, como o *Pytest*, e abordagens guiadas por comportamento, como o *Robot Framework*[2] utilizando *BDD (Behavior-Driven Development)*, são amplamente utilizadas em aplicações Python, especialmente em testes de aplicações web. Lenka *et al*. [2018] apresentam lacunas na eficácia do *BDD* para automação de testes de segurança, em torno da facilidade de compreensão e o impacto cognitivo sobre desenvolvedores e testadores iniciantes. Conforme evidenciado por Binamungu and Maro [2023], embora o *BDD* seja amplamente adotado, ainda faltam estudos que investiguem sua aplicação em contextos específicos, como segurança.

O *BDD* é uma abordagem de desenvolvimento orientada a comportamento que visa aproximar a comunicação entre desenvolvedores, testadores e *stakeholders* por meio de uma linguagem mais acessível. Nessa abordagem, os testes são especificados com base no comportamento esperado do sistema, geralmente utilizando a estrutura *Given–When–Then (Dado–Quando–Então)*. Por exemplo, um cenário de teste pode ser descrito como: *Given* um usuário autenticado, *When* ele envia uma requisição com entrada maliciosa, *Then* o sistema deve rejeitar a requisição, ilustrando de forma clara o comportamento esperado diante de uma tentativa de *SQL Injection*.

Diante disso, este trabalho apresenta uma investigação experimental entre duas abordagens de automação de testes de segurança: uma abordagem tradicional, utilizando *Pytest*, e outra denominada neste artigo de *SBDD (Security Behavior-Driven Development)*. Esta última consiste na aplicação do *BDD* com o *Robot Framework* para a automação de testes de segurança. O estudo concentra-se em testes de segurança dinâmicos automatizados, voltados à identificação de vulnerabilidades em aplicações web e *Application Programming Interface*, com foco específico na detecção de falhas do tipo *SQL Injection*. O objetivo é investigar e comparar ambas as abordagens, bem como analisar seus efeitos na carga cognitiva de testadores iniciantes em testes de segurança.

Para isso, foi conduzido um experimento controlado. Inicialmente, foi realizado um processo de triagem dos participantes, e posteriormente os selecionados foram distribuídos aleatoriamente em dois grupos: um utilizando *Pytest* e outro utilizando *Robot Framework* com *BDD* para testes de segurança. Após a execução das tarefas, os participantes responderam formulários para avaliação do desempenho na atividade e aplicação da escala *NASA-TLX (NASA Task Load Index)*, desenvolvida por Hart [1988] , utilizada para mensurar a carga cognitiva.

As principais contribuições deste trabalho são: (i) a definição da abordagem *SBDD* para automação de testes de segurança utilizando *BDD* e *Robot Framework*; (ii) a condução de um experimento controlado comparando *SBDD* e *Pytest* na compreensão de testes de segurança por participantes iniciantes; (iii) a avaliação dos efeitos das abordagens sobre a carga cognitiva por meio da escala *NASA-TLX*; e (iv) a disponibilização dos artefatos experimentais para apoiar a replicação do estudo.

Este trabalho se organiza em 2 - Trabalhos Relacionados: são discutidos estudos prévios que destacam contribuições, avanços e lacunas que fundamentam a relevância deste trabalho, 3 - Método: relata o passo-a-passo utilizado para a construção da pesquisa; 4 - Resultados e Discussões: apresenta os dados obtidos e suas implicações; e 5 - Conclusão: resume as contribuições do trabalho, aponta limitações e sugere trabalhos futuros.

2 Trabalhos Relacionados

Esta seção apresenta trabalhos relacionados ao uso do *Behavior-Driven Development* e abordagens de testes de segurança no contexto de desenvolvimento de software. O objetivo é identificar evidências existentes sobre a aplicação dessas práticas, suas contribuições e limitações, destacando lacunas de pesquisa que motivam a investigação experimental proposta neste estudo.

O uso do *BDD* tem sido amplamente adaptado no processo de desenvolvimento de software. Santos *et al*. [2024] investigam o desafio de garantir qualidade e cobertura de testes em POCs no contexto ágil. O estudo está inserido especialmente no domínio de cibersegurança, onde a qualidade é crítica. Para isso, propõem a ferramenta *AutoDevSuite*, baseada em LLMs. A ferramenta gera automaticamente testes a partir de user stories e critérios de aceitação em *BDD*. O objetivo foi aumentar a cobertura de testes sem comprometer a agilidade do desenvolvimento.Como resultado, a cobertura evoluiu de menos de 30% para valores próximos de 100%.Entretanto, o trabalho não considera a avaliação da carga cognitiva dos testadores.

Na perspectiva de integração com práticas de segurança, Wang and Wagner [2018] investigaram o uso do *BDD* combinado com *STPA (System-Theoretic Process Analysis)* como alternativa ao UAT (Teste de Aceitação do Usuário), padrão para verificação de segurança em desenvolvimento ágil. O experimento preliminar indica que a abordagem *STPA-BDD* favorece a comunicação na verificação de segurança em ambientes ágeis. Entretanto, são necessários estudos adicionais para confirmar sua efetividade e ampliar a generalização dos resultados.

De forma complementar, Purkayastha *et al*. [2020] implementaram um método formal de testes de segurança em sistemas de saúde digitais usando *BDD* com *Pytest* e métricas do *CVSS (Common Vulnerability Scoring System)*. O estudo demonstra que o uso de *BDD* com *Pytest* e métricas *CVSS* permite automatizar testes de segurança e monitorar continuamente vulnerabilidades em sistemas de saúde digitais. Porém, o estudo inicial cobre apenas alguns aspectos de segurança (autenticação e disponibilidade).

Em contraste, Rubatino *et al*. [2024] propõem um método que integra *STPA* ao *BDD*, visando apoiar de forma sistemática a elicitação e verificação de requisitos de segurança em sistemas críticos. Contudo, o método foi avaliado

[1] https://www.sonarsource.com/products/sonarqube/
[2] https://robotframework.org/

Tabela 1. Comparação dos trabalhos relacionados

Estudo	*BDD*	Segurança	Avaliação Experimental	Ferramenta
Okubo et al. (2014)	✓	✓	✗	✗
Wang et al. (2018)	✓	✓	✗	✗
Purkayastha et al. (2020)	✓	✓	✗	✓
Lima et al. (2022)	✓	✓	✗	✗
Rubatino et al. (2024)	✓	✓	✗	✗
Shexmo et al. (2024)	✓	✓	✗	✓
Este trabalho	✓	✓	✓	✓

apenas em exemplos, não em projetos reais da indústria, o que pode restringir a generalização dos resultados.

Em um contexto industrial, Lima *et al*. [2022] investigaram como a indústria especifica cenários de teste em projetos com *BDD* e propuseram cenários de teste de segurança para apoiar o desenvolvimento de sistemas secure by design. O estudo demonstra que a especificação de cenários de teste de segurança em projetos *BDD* pode auxiliar no desenvolvimento de software mais seguro, integrando requisitos e testes de forma colaborativa. Ainda assim, o estudo foca em empresas que adotam *BDD* e pode não refletir práticas de organizações que usam outras abordagens de desenvolvimento.

De maneira alternativa, Okubo *et al*. [2014] propõe o método BehaveSafe, que utiliza um grafo de Ameaças e Contramedidas (T&C graph) para definir comportamentos de segurança e critérios de aceitação em projetos *BDD*. O método permite que desenvolvedores verifiquem o nível de segurança das contramedidas implementadas em desenvolvimento ágil, garantindo mitigação adequada das ameaças identificadas. Entretanto, o método foi avaliado apenas em um sistema web e em nível conceitual, não cobrindo ampla variedade de contextos ou validação em larga escala na indústria.

Na Tabela 1, os trabalhos são comparados com base em quatro critérios: uso de *BDD*, aplicação em testes de segurança, presença de avaliação experimental e suporte por ferramentas. A escolha desses critérios foi motivada pelo objetivo deste estudo, que busca analisar não apenas a aplicação do *BDD* no contexto de segurança, mas também a existência de validações experimentais e o suporte prático das abordagens propostas.

A partir dessa análise, observa-se que, embora existam avanços na aplicação do *BDD* em testes de segurança, a maioria dos estudos não realiza avaliações experimentais com participantes, sendo escassos aqueles que consideram fatores humanos, como carga cognitiva em testadores iniciantes.

3 Método

Nesta seção são apresentadas as decisões e o protocolo definido para o experimento controlado entre grupos visando duas abordagens de escrita de testes automatizados utilizando a linguagem de programação Python. Entre as abordagens foram selecionadas i) *Pytest*, biblioteca para escrita de testes unitários dentro de aplicações desenvolvidas em Python; e 2) *Behavior-Driven Development (BDD)* com *Robot Framework*, framework multifuncional com suporte a testes de interface, *Application Programming Interface*, desempenho, entre outros testes automatizados.

O desenho experimental deste estudo foi elaborado com base nas recomendações de Wohlin *et al*. [2024] para experimentação em Engenharia de Software, contemplando a definição dos objetivos e hipóteses de pesquisa, seleção e balanceamento dos participantes, randomização dos grupos, condução do experimento e análise dos resultados.

Considerando a comparação entre essas duas abordagens, o objetivo deste estudo é avaliar seu impacto na compreensão dos testes de segurança e na carga cognitiva percebida por participantes noviços ao analisarem testes de segurança, utilizando a aplicação web OWASP Juice Shop[3] como objeto do estudo. O OWASP Juice Shop é uma aplicação web intencionalmente vulnerável, amplamente utilizada para treinamento e avaliação de práticas de segurança, contendo diversas falhas conhecidas, incluindo vulnerabilidades do OWASP Top 10, como *SQL Injection*, autenticação inadequada e exposição de dados sensíveis. Sua utilização neste estudo permite a aplicação controlada de cenários realistas de testes de segurança. Desta forma, as perguntas de pesquisa e hipóteses de pesquisa foram formuladas e apresentadas abaixo.

RQ1: Qual é o impacto da utilização do *Robot Framework* (*BDD*) vs. *Pytest* na carga temporal percebida por testadores noviços ao analisarem e compreenderem testes automatizados de segurança?

$H_{0,tempo}$: Não há diferença estatisticamente significativa na demanda temporal percebida (escala *NASA-TLX*) por participantes noviços ao analisarem testes em *Pytest* em comparação com o *Robot Framework*.

$H_{1,tempo}$: Existe uma diferença estatisticamente significativa na demanda temporal percebida que participantes noviços levam para analisar os casos de testes de segurança escritos em *Pytest* em comparação com o *Robot Framework*.

RQ2: Como a abordagem *BDD* (*Robot Framework*) influencia a carga cognitiva percebida por testadores noviços em comparação com uma abordagem tradicional (*Pytest*) durante a análise de testes de segurança?

$H_{0,cogn}$: Não há diferença estatisticamente significativa na carga cognitiva (medida pelo *NASA-TLX*) percebida por participantes noviços ao analisarem tarefas de teste de segurança com *Pytest* em comparação com o *Robot Framework*.

$H_{1,cogn}$: A carga cognitiva percebida por participantes noviços será significativamente menor ao analisarem testes escritos em *Robot Framework* em comparação com o *Pytest* para as mesmas tarefas.

RQ3: Qual abordagem (*Robot Framework* vs. *Pytest*) leva a uma maior acurácia na compreensão de testes de segurança por participantes noviços?

[3] https://owasp.org/www-project-juice-shop/

$H_{0,acur}$: Não há diferença estatisticamente significativa na pontuação média do questionário de compreensão entre o grupo que analisa testes em *Pytest* e o grupo que analisa testes em *Robot Framework*.

$H_{1,acur}$: A pontuação média do questionário de compreensão será significativamente maior para participantes noviços que analisam testes escritos em *Robot Framework* em comparação com *Pytest*.

3.1 Planejamento do Experimento

A Figura 1 apresenta uma visão geral do desenho experimental adotado neste estudo. O fluxograma ilustra o processo de recrutamento e triagem dos participantes, a aplicação dos critérios de exclusão, a estratificação por nível de proficiência em Python, a alocação balanceada nos grupos (*Pytest* e *Robot Framework* com *BDD*), bem como a etapa de coleta das variáveis dependentes.

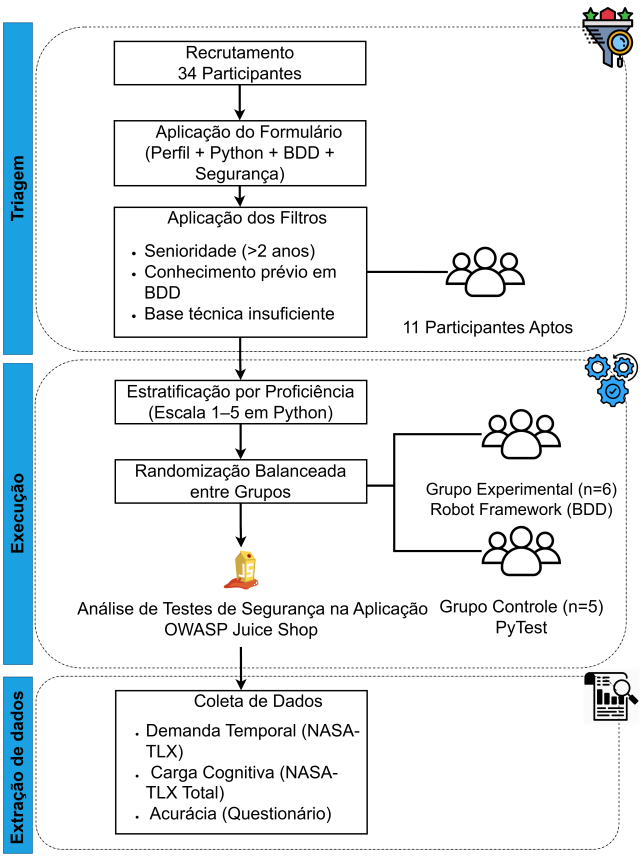

Figura 1. Desenho do experimento controlado

O desenho experimental manipula uma única variável independente: a abordagem de teste de segurança automatizado. Este fator é composto por dois níveis, ou tratamentos. O primeiro tratamento, considerado o grupo de controle, consiste na utilização da abordagem tradicional com a ferramenta *Pytest*. O segundo tratamento envolve a aplicação da abordagem *BDD* com o *Robot Framework* para a execução das mesmas tarefas.

Para avaliar o impacto dos tratamentos, foram definidas três variáveis dependentes, cada uma vinculada a uma questão de pesquisa. A demanda temporal percebida (RQ1) foi mensurada pela dimensão de Demanda Temporal do *NASA-TLX* (escala de intervalo). A carga cognitiva (RQ2) foi aferida pela pontuação no questionário *NASA-TLX* (escala de intervalo). A acurácia (RQ3) foi medida pela corretude da tarefa, avaliada por meio da taxa de acertos, erros e respostas nulas (escala nominal).

A fim de garantir a validade interna do estudo, diversas variáveis foram rigorosamente controladas. O nível de experiência dos participantes foi homogeneizado, selecionando-se apenas indivíduos noviços. Todos os participantes receberam tarefas de teste idênticas e interagiram com a mesma aplicação alvo (OWASP Juice Shop). Adicionalmente, a linguagem de programação subjacente, Python, foi mantida constante em ambos os tratamentos para isolar o efeito exclusivo das abordagens de teste.

3.2 Participantes

Para a participação no estudo, foram realizados dois momentos: (i) divulgação do formulário de triagem via WhatsApp, em grupos de projetos de software, grupos acadêmicos e laboratoriais e (ii) aplicação do instrumento aos participantes. O formulário foi composto por 10 perguntas organizadas nas seguintes categorias: Perfil acadêmico e experiência, Experiência com programação em Python, Experiência com *BDD* e Gherkin e conhecimento em Segurança da Informação. A versão completa do instrumento de triagem[4] está publicamente disponível no repositório Zenodo, visando transparência e reprodutibilidade do estudo.

O estudo adotou como critério de noviço participantes com até dois anos de experiência profissional e sem experiência prévia em *BDD* e testes de segurança, permitindo, entretanto, diferentes níveis de proficiência em programação em Python dentro desse recorte. Dessa forma, foi possível incluir participantes com maior familiaridade técnica, desde que ainda enquadrados no perfil de baixa experiência profissional e sem viés em *BDD*.

Através do formulário, obteve-se um total de 34 participantes. Com o objetivo de restringir a amostra a participantes noviços, foram aplicados três filtros de exclusão: (i) senioridade, excluindo participantes com mais de dois anos de experiência profissional; (ii) viés de conhecimento em *BDD*, uma vez que participantes com conhecimento prévio nessa abordagem poderiam apresentar vantagem em relação àqueles que não a conhecem; e (iii) base técnica mínima, excluindo participantes que indicaram não possuir experiência mínima na escrita de funções em Python ou no uso de ferramentas de testes automatizados. Após a etapa de triagem, foram selecionados 11 participantes para o estudo.

Tabela 2. Caracterização dos participantes

Formação acadêmica	**n**	**Experiência profissional**	**n**
Graduando	10	0 anos	2
Graduado	1	< 1 ano	3
		1–2 anos	6
Uso de Python	**n**	**Proficiência em Python**	**n**
Diário	3	Baixa (1–2)	5
Semanal	3	Alta (3–4)	6
Raramente	5		
Experiência com BDD	**n**	**Conhecimento em Segurança**	**n**
Nenhuma	11	Baixo	8
		Moderado	3

[4]https://doi.org/10.5281/zenodo.20693912

A Tabela 2 evidencia que a amostra é composta predominantemente por participantes com baixa experiência profissional, reforçando a adequação do perfil noviço definido no estudo. Observa-se, entretanto, uma variabilidade controlada na proficiência em Python, com distribuição equilibrada entre níveis baixos e altos, o que justifica sua utilização como variável de controle no balanceamento experimental. Adicionalmente, a ausência de experiência prévia com *BDD* entre os participantes elimina potenciais vieses de familiaridade com a abordagem, enquanto o baixo nível de conhecimento em segurança da informação indica um cenário realista para avaliação de técnicas voltadas a esse domínio.

A avaliação do nível de proficiência em Python foi considerada relevante, uma vez que ambas as abordagens avaliadas são baseadas na linguagem Python, porém diferem em nível de abstração e forma de expressão dos testes. Enquanto o *Pytest* exige maior interação com código procedural, o *Robot Framework* com *BDD* utiliza uma sintaxe mais descritiva. Dessa forma, o nível de proficiência em Python pode influenciar diretamente a compreensão dos testes e a carga cognitiva percebida, sendo tratado como uma variável de controle no balanceamento dos grupos experimentais.

Os participantes foram divididos em dois grupos: i) grupo controle, o uso do *Pytest* para entender os cenários de testes de segurança e ii) grupo experimental, utilizando *Robot Framework* e *BDD* na escrita dos casos. Desta forma, os participantes foram estratificados por nível técnico, sendo inicialmente agrupados conforme sua proficiência em Python e capacidade de escrita de funções, e posteriormente alocados de forma aleatória e balanceada entre os grupos *Pytest* e *Robot Framework*.

Tabela 3. Balanceamento dos grupos por proficiência técnica

Métrica	***Pytest***	**Robot**	**Delta**
Participantes (n)	5	6	–
Nível alto (%)	60	50	Baixa
Nível baixo (%)	40	50	Baixa
Proficiência média (1–5)	2.60	2.83	0.23 (Insig.)

Observa-se que, dentro do conjunto de participantes noviços, cinco foram alocados ao grupo *Pytest*, dos quais 60% apresentaram nível alto de proficiência em Python e 40% nível baixo. No grupo *Robot Framework*, composto por seis participantes, a distribuição foi equilibrada (50% alto e 50% baixo), indicando homogeneidade na composição dos grupos quanto à proficiência técnica.

A diferença na média de proficiência em Python entre os grupos foi de apenas 0.23 pontos em uma escala de 5, sendo considerada insignificante, o que indica equivalência técnica inicial entre os grupos.

3.3 Construção dos Artefatos de Teste

Para a condução do experimento, foram desenvolvidos dois conjuntos equivalentes de casos de teste de segurança aplicados à aplicação OWASP Juice Shop, um utilizando *Pytest* e outro utilizando *Robot Framework* com *BDD* (*SBDD*). O objetivo foi garantir equivalência funcional entre os testes, permitindo comparação justa entre as abordagens.

Os testes contemplaram cenários típicos de vulnerabilidades web, incluindo autenticação, injeção e manipulação de requisições, de acordo com boas práticas sugeridas pela OWASP. Cada caso de teste foi inicialmente especificado de forma neutra, descrevendo o comportamento esperado e o objetivo de segurança. Em seguida, os casos foram implementados e estão disponíveis no GitHub: https://github.com/ASTRID-RESEARCH/SBDD.git

Os casos de teste foram elaborados com base em boas práticas recomendadas pela OWASP, especialmente aquelas descritas no OWASP Top 10. O foco principal foi a categoria A03: Injection, com ênfase na exploração de vulnerabilidades do tipo *SQL Injection* em aplicações web e *Application Programming Interface*.

Os cenários desenvolvidos incluem tentativas de bypass de autenticação por meio de injeção maliciosa em campos de login (CT01–CT03), extração indevida de informações sensíveis, como credenciais de usuários e estrutura do banco de dados (CT04 e CT06), bem como manipulação de consultas para criação de contas não autorizadas (CT05).

Além disso, os testes consideram práticas relacionadas à validação inadequada de entradas e falhas no controle de autenticação, refletindo vulnerabilidades comuns descritas pela OWASP. Esses cenários foram utilizados como base para avaliar a capacidade das abordagens em representar, estruturar e facilitar a compreensão de testes de segurança por testadores iniciantes.

3.4 Análise de Dados

Para a análise dos dados coletados, foi realizada inicialmente uma análise descritiva com o objetivo de caracterizar o desempenho e a percepção dos participantes em relação às abordagens avaliadas. Os dados obtidos por meio do questionário pós-experimento[5] e das tarefas realizadas foram organizados e analisados considerando três aspectos: carga temporal percebida, carga cognitiva e acurácia na compreensão dos casos de teste de segurança.

Para responder à RQ1, foi utilizada a dimensão Demanda Temporal do *NASA-TLX*, sendo analisadas medidas de tendência central e dispersão dos dados para cada abordagem. A opção por utilizar essa dimensão decorre do objetivo de avaliar a percepção dos participantes quanto ao esforço temporal exigido pelas tarefas, e não o tempo real de execução.

Para responder à RQ2, foram analisadas todas as dimensões do *NASA-TLX*, comparando-se os valores obtidos pelos participantes dos grupos *Pytest* e *SBDD*. Foram consideradas medidas descritivas para avaliar diferenças na percepção de carga cognitiva entre as abordagens.

Para responder à RQ3, foi analisada a acurácia dos participantes por meio da taxa de acertos, erros e respostas nulas no questionário de compreensão dos casos de teste de segurança. A acurácia foi mensurada pela capacidade dos participantes identificarem corretamente o objetivo, a vulnerabilidade e o resultado esperado dos casos de teste analisados.

Além da análise descritiva, foi aplicado o teste não paramétrico de *Mann-Whitney U* para comparar os resultados obtidos entre os grupos independentes. A escolha desse teste deve-se ao tamanho reduzido da amostra e à ausência da necessidade de assumir distribuição normal dos dados. Para todas as análises inferenciais foi adotado nível de significância de 5% ($\alpha = 0,05$).

[5] `https://doi.org/10.5281/zenodo.18785762`

4 Resultados e Discussões

Esta seção apresenta os resultados da aplicação do *NASA-TLX*, incluindo a análise das comparações par a par entre as dimensões e das pontuações atribuídas pelos participantes. Os dados são apresentados de forma descritiva e discutidos à luz da carga de trabalho percebida no contexto do estudo.

4.1 RQ1 - Qual é o impacto da utilização do *Robot Framework* (*BDD*) vs. *Pytest* na carga temporal percebida por testadores noviços ao analisarem e compreenderem testes automatizados de segurança?

No que se refere à **RQ1**, que investiga o impacto das abordagens na demanda temporal percebida pelos participantes noviços, os resultados indicam que o *SBDD* foi percebido como menor pressão temporal quando comparado ao *Pytest*. Conforme evidenciado na análise da dimensão Demanda Temporal do *NASA-TLX*, o *Pytest* apresentou valores medianos mais elevados e maior variabilidade, sugerindo maior pressão de tempo percebida durante a execução das tarefas.

Embora o tempo de execução não tenha sido mensurado objetivamente, a análise descritiva da dimensão Demanda Temporal do *NASA-TLX* sugere uma menor pressão temporal percebida pelos participantes que utilizaram o *SBDD* em comparação ao *Pytest*.

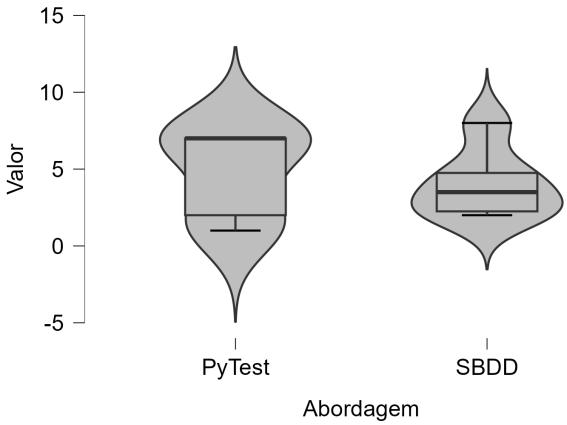

Figura 2. Distribuição de Demanda Temporal *Pytest* e *SBDD*.

Em conformidade com a Figura 2 o *Pytest* apresenta maior variabilidade e valores centrais mais elevados, com mediana em torno de 7, indicando maior pressão de tempo percebida por parte dos participantes. A distribuição sugere que, embora alguns participantes tenham relatado baixa demanda temporal, a maioria percebeu restrições de tempo mais acentuadas durante a execução das tarefas.

Em contraste, o *SBDD* apresenta valores predominantemente baixos a moderados, com mediana próxima de 3,5, indicando menor pressão temporal para a maior parte dos participantes. Apesar da presença de um valor mais elevado, a distribuição mais homogênea sugere que a abordagem proporciona um ritmo de execução mais controlado e previsível. Esses resultados são coerentes com a distribuição mais equilibrada da carga de trabalho observada no *SBDD* nas demais dimensões do *NASA-TLX*.

Para verificar se a diferença observada era estatisticamente significativa, foi aplicado o teste U de Mann-Whitney para comparar a demanda temporal percebida entre os grupos *Pytest* (n = 5) e *SBDD* (n = 7). Os resultados não indicaram diferença estatisticamente significativa entre as abordagens (U = 16,0; p = 0,434). Dessa forma, embora a análise descritiva sugira menor demanda temporal para o *SBDD*, não foi possível rejeitar a hipótese nula $H_{0,tempo}$.

Resposta para a RQ1: Os resultados descritivos sugerem que a abordagem *SBDD* foi percebida pelos participantes noviços como apresentando menor demanda temporal quando comparada ao *Pytest*. Entretanto, a diferença observada não foi estatisticamente significativa segundo o teste U de Mann-Whitney (p = 0,434), não sendo possível rejeitar a hipótese nula $H_{0,tempo}$

4.2 RQ2: Como a abordagem *BDD* com *Robot Framework* influencia a carga cognitiva percebida por testadores noviços em comparação com uma abordagem tradicional *Pytest* durante a análise de testes de segurança?

Em relação à **RQ2**, que avalia a carga cognitiva percebida, os resultados do *NASA-TLX* indicam que a abordagem *SBDD* impôs menor carga cognitiva global quando comparada ao *Pytest*.

Tabela 4. Comparação do peso médio por dimensões entre *SBDD* e *Pytest*

Dimensão	*SBDD*	*Pytest*
Esforço	3,00	2,60
Demanda Mental	3,83	4,20
Demanda Temporal	2,67	2,40
Frustração	1,00	4,20
Desempenho	3,83	1,00
Demanda Física	0,67	0,60

Conforme apresentado na Tabela 4, observa-se que, na abordagem *SBDD*, os pesos atribuídos às dimensões do *NASA-TLX* encontram-se distribuídos de forma mais homogênea, não havendo concentração acentuada em uma única dimensão. Em contraste, na abordagem *Pytest*, duas dimensões se destacam de forma predominante na carga de trabalho percebida.

Destacam-se as dimensões Demanda Mental e Frustração, ambas com peso médio de 4,2, indicando maior sobrecarga cognitiva e emocional. Verifica-se que a dimensão Frustração apresentou peso consideravelmente superior no *Pytest* quando comparada ao *SBDD* (4,2 e 1, respectivamente), sugerindo que os participantes perceberam a abordagem *Pytest* como mais frustrante durante a execução das tarefas.

A Demanda Mental apresentou valores elevados em ambas as abordagens, porém com comportamentos distintos. No *Pytest*, essa dimensão concentra de forma mais acentuada a carga cognitiva (4,2), enquanto no *SBDD* a demanda mental (3,83) é compartilhada com outras dimensões relevantes, como Desempenho (3,83) e Esforço (3), indicando uma dis-

tribuição mais equilibrada da carga de trabalho. Por fim, a Demanda Física apresentou pesos reduzidos em ambas as abordagens (0,67 no *SBDD* e 0,6 no *Pytest*), o que valida a aplicação do instrumento, uma vez que as tarefas analisadas são predominantemente cognitivas..

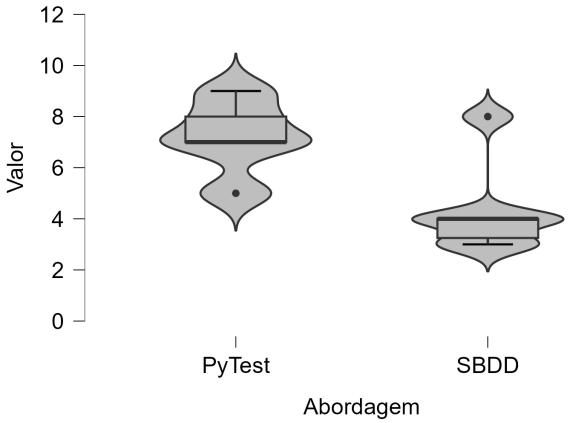

Figura 3. Distribuição de Demanda Mental *Pytest* e *SBDD*.

De acordo com a Figura 3 o *Pytest* apresenta valores consistentemente mais elevados, com mediana em torno de 7, indicando que os participantes perceberam essa abordagem como mentalmente mais exigente. Uma possível explicação reside na natureza estrutural do *Pytest*, que requer leitura e interpretação direta do código, implicando maior esforço de análise sintática e rastreamento lógico das instruções.

De forma contrária, o *SBDD* apresenta valores predominantemente mais baixos, com maior concentração entre 3 e 4, indicando menor demanda mental percebida para a maior parte dos participantes. Uma vez que ao estruturar o comportamento esperado de maneira declarativa e sequencial, o *SBDD* tende a facilitar o mapeamento mental entre cenário e expectativa, reduzindo o esforço necessário para compreensão e elaboração dos testes.

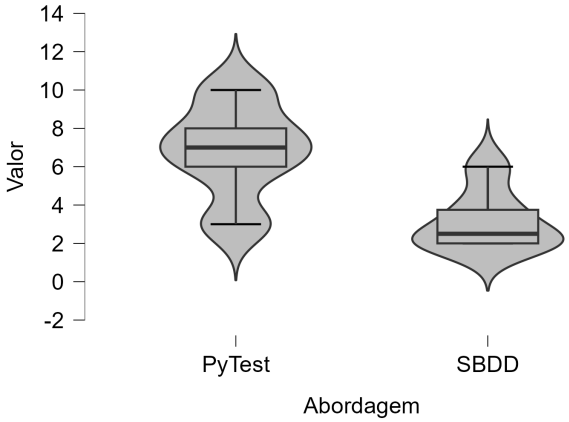

Figura 4. Distribuição de Esforço *Pytest* e *SBDD*.

Conforme a Figura 4, observa-se que o *Pytest* apresenta valores mais elevados e maior dispersão, com mediana em torno de 7 e presença de valores extremos, indicando maior esforço percebido. Esse resultado indica não apenas maior esforço percebido, mas também maior variabilidade na experiência dos participantes, sugerindo que a abordagem pode exigir diferentes níveis de mobilização cognitiva dependendo do perfil técnico individual.

Em contrapartida, o *SBDD* apresenta valores mais baixos e concentrados, com mediana próxima de 2,5, sugerindo uma percepção de esforço mais homogênea e reduzida. A menor dispersão pode sugerir que a estrutura Given–When–Then atua como um mecanismo de padronização cognitiva, oferecendo uma sequência explícita para organização do raciocínio e elaboração dos testes.

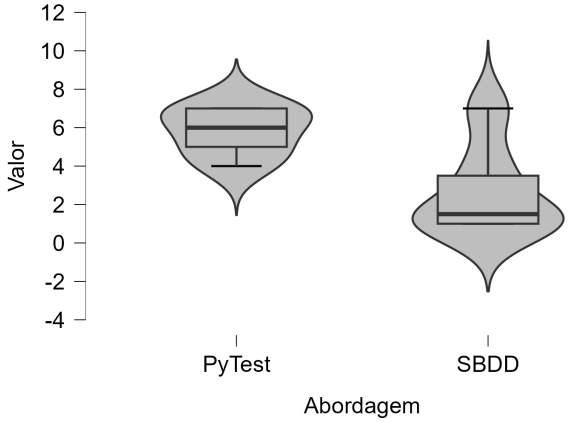

Figura 5. Distribuição de Frustação *Pytest* e *SBDD*.

De acordo com a Figura 5, o *Pytest* apresentou valores predominantemente médios a altos, com mediana próxima de 6, indicando maior percepção de frustração entre os participantes. Esse resultado pode estar associado à necessidade de maior controle sintático e estrutural durante a elaboração dos testes, o que pode gerar incerteza e sensação de maior risco de erro.

Todavia, o *SBDD* apresentou valores majoritariamente baixos, concentrados entre 1 e 2, sugerindo que a estrutura comportamental do *BDD* contribuiu para uma experiência mais previsível e controlada. A organização declarativa dos cenários pode ter reduzido ambiguidades e facilitado o mapeamento entre intenção e verificação do comportamento esperado.

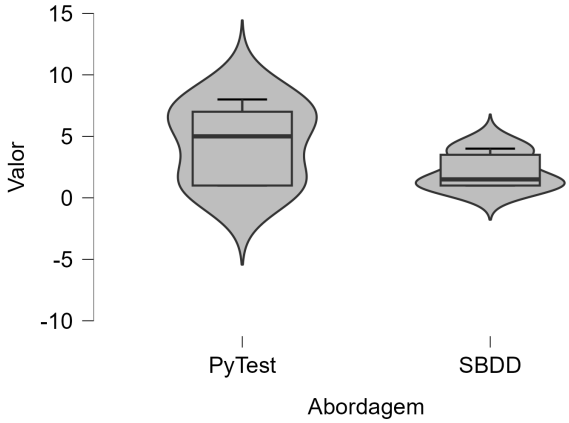

Figura 6. Distribuição de Demanda Física *Pytest* e *SBDD*.

Em conformidade com a Figura 6, observa-se que, em ambas as abordagens, os valores atribuídos à demanda física são predominantemente baixos, o que era esperado, dado que a atividade envolvia essencialmente processamento cognitivo

e interação padrão com computador. No *SBDD*, os valores encontram-se concentrados próximos ao mínimo da escala, indicando percepção quase inexistente de esforço físico.

O *Pytest*, embora também apresente níveis reduzidos, demonstra maior dispersão. Essa variabilidade pode estar associada a diferenças individuais no ritmo de digitação e leitura ou na manipulação direta de código, uma vez que essa abordagem exige maior controle estrutural e sintático.

A ausência de diferenças expressivas na dimensão física reforça que as variações observadas nas dimensões de demanda mental, esforço e frustração estão mais relacionadas às características cognitivas das abordagens do que a fatores externos ou físicos. Tal padrão fortalece a validade interna do experimento, sugerindo que os efeitos identificados derivam principalmente da estrutura das técnicas avaliadas.

Para verificar se as diferenças observadas eram estatisticamente significativas, foi aplicado o teste U de Mann-Whitney às dimensões cognitivas do *NASA-TLX*. Os resultados indicaram diferenças estatisticamente significativas para Demanda Mental (U = 4,5; p = 0,020), Esforço (U = 4,5; p = 0,020) e Frustração (U = 4,5; p = 0,019), com valores mais elevados para o grupo *Pytest*. Para a dimensão Desempenho, não foi observada diferença estatisticamente significativa entre os grupos (U = 10,5; p = 0,143). Esses resultados fornecem evidências de que a abordagem *SBDD* reduz aspectos importantes da carga cognitiva percebida, permitindo rejeitar a hipótese nula $H_{0,cogn}$.

Resposta para a RQ2: Os resultados indicam que a abordagem *SBDD* utilizando *Robot Framework* apresentou menor carga cognitiva percebida em comparação com a abordagem tradicional baseada em *Pytest*. Diferenças estatisticamente significativas foram observadas nas dimensões Demanda Mental (p = 0,020), Esforço (p = 0,020) e Frustração (p = 0,019) da escala *NASA-TLX*, com maiores níveis de carga cognitiva reportados pelos participantes que utilizaram *Pytest*. Esses resultados sugerem que a estrutura descritiva do *BDD* facilita a compreensão dos testes de segurança por testadores noviços, permitindo rejeitar a hipótese nula $H_{0,cogn}$

4.3 RQ3: Qual abordagem *Robot Framework* vs. *Pytest* leva a uma maior acurácia na compreensão de testes de segurança por participantes noviços?

Por fim, no contexto da **RQ3**, que analisa a acurácia na compreensão dos testes de segurança, a Tabela 5 demonstra que o *SBDD* apresentou desempenho superior, especialmente nas categorias Vulnerabilidade e Passou. O *SBDD* obteve maiores taxas de acerto e menor incidência de respostas nulas, enquanto o *Pytest* apresentou maior ambiguidade na interpretação dos critérios de aprovação.

Esses resultados indicam que o *SBDD* proporciona maior clareza semântica e estrutural na especificação dos testes, contribuindo para maior acurácia na análise por participantes noviços, o que leva à rejeição da hipótese nula $H_{0,acur}$.

Além da avaliação subjetiva de carga de trabalho, foi analisado o desempenho dos participantes por meio da taxa de acerto nos casos de teste, classificados em três categorias: Objetivo, Vulnerabilidade e Sucesso. Para cada caso de teste, as respostas foram categorizadas em acerto, erro ou nulo (não soube responder).

Tabela 5. Desempenho dos participantes nas abordagens *Pytest* e *SBDD*

Abordagem	Categoria	Acertos	Erros	Nulos
Pytest	Objetivo	26	1	3
Pytest	Vulnerabilidade	9	9	12
Pytest	Passou	16	3	11
SBDD	Objetivo	28	5	3
SBDD	Vulnerabilidade	16	11	9
SBDD	Passou	31	0	5

De acordo com a Tabela 5 observa-se que ambas as abordagens possibilitam a correta identificação do objetivo dos casos de teste. No *Pytest*, foram 86,67% de acertos, enquanto no *SBDD* foram 77,78%. Na categoria Vulnerabilidade, o *SBDD* apresentou melhor desempenho, com 44,44% de acertos, frente a 30,00% no *Pytest*.

Essa diferença torna-se ainda mais evidente na categoria Passou, onde no *SBDD* 86,11% das respostas foram corretas, sem ocorrência de erros, enquanto no *Pytest* apenas 53,33% foram acertos. Esses resultados sugerem que o *SBDD* oferece critérios de aprovação mais claros. Tais achados estão alinhados com os resultados do *NASA-TLX*, reforçando a relação entre redução da carga cognitiva e melhoria do desempenho.

No que se refere à dimensão Desempenho do *NASA-TLX*, os resultados indicam que, no *SBDD*, os participantes demonstraram maior percepção sobre seu próprio desempenho (peso médio de 3,83), enquanto no *Pytest* essa dimensão apresentou menor influência (peso médio de 1).

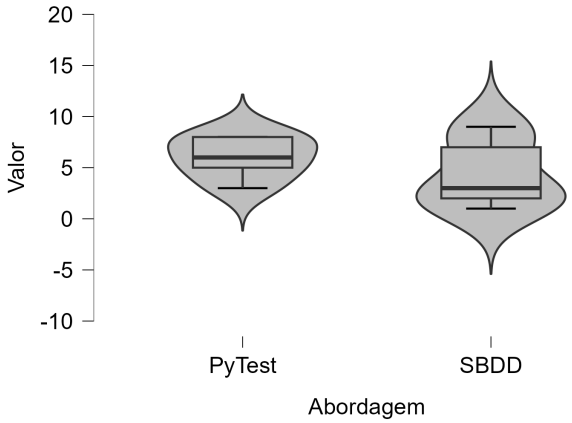

Figura 7. Distribuição de Desempenho *Pytest* e *SBDD*.

Conforme a Figura 7, observa-se que o *SBDD* apresenta valores predominantemente mais baixos na dimensão de desempenho. Considerando que a escala de performance do instrumento *NASA-TLX* é invertida, ou seja, na qual valores mais baixos indicam melhor desempenho percebido e menor carga de trabalho, esse resultado sugere que os participantes se sentiram mais bem-sucedidos e satisfeitos ao utilizar o *SBDD*.

Em contraste, o *Pytest* apresentou mediana em torno de 6, indicando percepção relativamente inferior de desempenho.

Esse achado está alinhado com os resultados previamente observados nas dimensões de demanda mental, esforço e frustração, nas quais o *Pytest* apresentou valores mais elevados.

Foi utilizado o teste não paramétrico de Mann–Whitney U para comparar o desempenho entre os grupos Pytest e SBDD. Os resultados não indicaram diferença estatisticamente significativa entre os grupos (U = 12.5; p = 0.815), não sendo possível afirmar que o grupo Pytest apresenta desempenho superior ao grupo SBDD.

> **Resposta para a RQ3:** Os resultados descritivos indicam uma tendência de melhor desempenho do *SBDD* na compreensão dos testes de segurança por participantes noviços, especialmente na identificação de vulnerabilidades e na interpretação dos critérios de aprovação dos testes. No entanto, o teste estatístico de Mann–Whitney U não indicou diferença significativa entre os grupos (U = 12.5; p = 0.815), de modo que não se rejeita a hipótese nula $H_{0,acur}$, não sendo possível afirmar estatisticamente a superioridade de uma abordagem sobre a outra.

4.4 Ameaças à Validade

Este estudo apresenta algumas ameaças à validade que devem ser consideradas na interpretação dos resultados.

Uma ameaça à validade externa refere-se ao número reduzido de participantes e à concentração da amostra em indivíduos com nível iniciante de experiência. Esse fator pode limitar a generalização dos resultados para contextos que envolvem profissionais mais experientes ou ambientes industriais.

Em relação à validade interna, destaca-se como possível limitação o nível de familiaridade prévia dos participantes com as ferramentas utilizadas, especialmente *Pytest* e *Robot Framework*. Diferenças nessa familiaridade podem ter influenciado a percepção de carga cognitiva e o desempenho na execução das tarefas, introduzindo um possível viés na comparação entre as abordagens avaliadas.

Adicionalmente, o estudo não incluiu métricas objetivas de tempo como variável de análise, uma vez que priorizou a avaliação da percepção temporal por meio da dimensão Demanda Temporal do *NASA-TLX*. Dessa forma, os resultados relacionados ao tempo refletem a percepção dos participantes e não necessariamente o tempo real necessário para execução das atividades.

Embora tenha sido aplicado o teste não paramétrico de *Mann-Whitney U*, o reduzido tamanho da amostra limita o poder estatístico das análises realizadas, podendo dificultar a identificação de diferenças sutis entre as abordagens avaliadas.

Por fim, quanto à validade de construção, embora a escala *NASA-TLX* seja amplamente utilizada para avaliar carga de trabalho percebida, trata-se de uma medida subjetiva baseada na percepção dos participantes, o que pode introduzir vieses individuais nas respostas. Além disso, o experimento foi conduzido utilizando apenas cenários de vulnerabilidade do tipo *SQL Injection* em uma aplicação específica (OWASP Juice Shop), o que pode limitar a generalização dos resultados para outros tipos de vulnerabilidades e contextos de teste de segurança.

5 Conclusão

Compreender o impacto da utilização do *Security Behavior-Driven Development (SBDD)* em comparação com testes tradicionais utilizando *Pytest*, no contexto de automação de testes de segurança, permitiu analisar como diferentes abordagens influenciam a carga de trabalho percebida e a acurácia na compreensão dos testes por participantes iniciantes.

Por meio deste estudo, observou-se que a abordagem *SBDD* apresentou uma distribuição mais equilibrada da carga de trabalho percebida, com menores níveis de demanda mental, esforço e frustração quando comparada ao *Pytest*. As análises estatísticas indicaram diferenças significativas nessas dimensões da escala *NASA-TLX*, sugerindo que a utilização de cenários estruturados por comportamento pode reduzir aspectos relevantes da carga cognitiva em participantes noviços.

Em relação à compreensão dos testes de segurança, os resultados descritivos apontaram uma tendência favorável ao *SBDD*, especialmente na identificação de vulnerabilidades e na interpretação dos critérios de aprovação dos testes. Entretanto, essa diferença não foi confirmada estatisticamente, não sendo possível afirmar a superioridade de uma abordagem sobre a outra quanto à acurácia na compreensão dos testes de segurança.

Esses achados fornecem evidências de que abordagens baseadas em *BDD* podem reduzir aspectos da carga cognitiva percebida durante a análise de testes de segurança por participantes com menor experiência. Além disso, os resultados sugerem potencial para apoiar a compreensão dos testes, embora estudos com amostras maiores sejam necessários para confirmar esse efeito. A partir da investigação empírica conduzida, a relação entre a abordagem utilizada, a carga cognitiva percebida e o desempenho na compreensão dos testes oferece percepções relevantes para a adoção de práticas mais acessíveis na automação de testes de segurança.

Trabalhos futuros podem ampliar a população estudada, incorporando participantes com diferentes níveis de senioridade e experiência em testes de software. Também são recomendadas investigações com amostras maiores, inclusão de métricas objetivas de desempenho, como tempo real de execução das tarefas, e avaliação da abordagem *SBDD* em outros tipos de vulnerabilidades e contextos industriais, permitindo ampliar a generalização dos resultados obtidos neste estudo.

Declarações complementares

Financiamento

O presente trabalho foi realizado com apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (Processo 312173/2025-3

Contribuições dos autores

Samuel contribuiu para a concepção do estudo, investigação, execução da pesquisa e redação do manuscrito. Rubens contribuiu com a investigação, definição da metodologia, redação e revisão do manuscrito. Ismayle contribuiu com a curadoria dos dados, administração do projeto e validação do estudo. Todos os autores leram e aprovaram a versão final do manuscrito.

Conflitos de interesse

Os autores declaram que não têm nenhum conflito de interesses

Referências

Abdiukov, T. (2024). Automated security testing in devsecops pipelines: Integrating ai-based vulnerability discovery and compliance validation. *World Journal of Advanced Research and Reviews*, 22:2083–2093. DOI: 10.30574/wjarr.2024.22.1.1083.

Aloraini, B., Nagappan, M., German, D. M., Hayashi, S., and Higo, Y. (2019). An empirical study of security warnings from static application security testing tools. *Journal of Systems and Software*, 158:110427. DOI: 10.1016/j.jss.2019.110427.

Binamungu, L. P. and Maro, S. (2023). Behaviour driven development: A systematic mapping study. *Journal of Systems and Software*, 203:111749. DOI: 10.1016/j.jss.2023.111749.

Hart, L. E. S. S. G. (1988). *Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research*, pages 139–183. DOI: 10.1016/S0166-4115(08)62386-9.

Lenka, R. K., Kumar, S., and Mamgain, S. (2018). Behavior driven development: Tools and challenges. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 1032–1037. IEEE. DOI: 10.1109/ICACCCN.2018.8748595.

Lima, M. Y., Silva, C., and Souza, R. (2022). Explorando a especificação de cenários de teste de segurança no desenvolvimento orientado por comportamento (behavior driven development). In *Anais do Workshop em Engenharia de Requisitos*. Even3. DOI: 10.29327/1298262.25-23.

Murazvu, G., Parkinson, S., Khan, S., Liu, N., and Allen, G. (2024). A survey on factors preventing the adoption of automated software testing: A principal component analysis approach. *Software*, 3:1–27. DOI: 10.3390/software3010001.

Okubo, T., Kakizaki, Y., Kobashi, T., Washizaki, H., Ogata, S., Kaiya, H., and Yoshioka, N. (2014). *Security and Privacy Behavior Definition for Behavior Driven Development*, pages 306–309. DOI: 10.1007/978-3-319-13835-0_28.

Purkayastha, S., Goyal, S., Phillips, T., Wu, H., Haakenson, B., and Zou, X. (2020). Continuous security through integration testing in an electronic health records system. In *2020 International Conference on Software Security and Assurance (ICSSA)*, pages 26–31. DOI: 10.1109/ICSSA51305.2020.00012.

Qusef, A., Elish, M. O., and Binkley, D. (2019). An exploratory study of the relationship between software test smells and fault-proneness. *IEEE Access*, 7:139526–139536. DOI: 10.1109/ACCESS.2019.2943488.

Rajapakse, R. N., Zahedi, M., Babar, M. A., and Shen, H. (2022). Challenges and solutions when adopting devsecops: A systematic review. *Information and Software Technology*, 141:106700. DOI: https://doi.org/10.1016/j.infsof.2021.106700.

Rubatino, V., Nogueira, A. B., de Souza, F. G. R., and Pagliares, R. M. (2024). *A Method Based on Behavior Driven Development (BDD) and System-Theoretic Process Analysis (STPA) for Verifying Security Requirements in Critical Software Systems*, pages 373–380. DOI: 10.1007/978-3-031-56599-1_48.

Santos, S., Fernandes, R., Santos, M., Soares, M., Rocha, F., and Marczak, S. (2024). Increasing test coverage by automating bdd tests in proofs of concepts (pocs) using llm. In *Anais do XXIII Simpósio Brasileiro de Qualidade de Software*, page 519–525, Porto Alegre, RS, Brasil. SBC. Disponível em: `https://sol.sbc.org.br/index.php/sbqs/article/view/32979`.

Wang, Y. and Wagner, S. (2018). Combining stpa and bdd for safety analysis and verification in agile development. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, pages 286–287. ACM. DOI: 10.1145/3183440.3194973.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2024). *Experimentation in Software Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg. DOI: 10.1007/978-3-662-69306-3.

Zhang, X., Shen, W., Liang, Z., Cui, L., and Wang, Y. (2024). Research and application of automated testing technology for data security vulnerabilities. In *2024 4th International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, pages 01–05. IEEE. DOI: 10.1109/ICMNWC63764.2024.10872029.