Caracterizando Condutores de Carga Cognitiva na Prática de Testes Unitários

Bruno Barroso[1], **Leonardo da Silva**[1], **Rafael de Mello**[1]

[1]Programa de Pós-Graduação em Informática
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

{blemosb,leonardocesar}@ufrj.br, rafaelmello@ic.ufrj.br

Abstract. *Unit testing is an important practice to promote the quality of the developed software product. However, it is common for developers to consider unit testing challenging, feeling little motivated to perform it. In this paper, we report a case study conducted to characterize the first version of a set of cognitive load drivers of unit testing activities. Through interviews conducted with developers from different teams at a Brazilian bank, we identified challenges and problems faced when performing unit tests. From this information, we coded a set of cognitive load drivers that cover different technical and non-technical aspects of the practice. Furthermore, the study results highlight the need for a standardized process to support and promote the practice of unit testing in organizations.*

Resumo. *Testes unitários são uma importante prática para promover a qualidade do produto de software desenvolvido. No entanto, é comum que esta prática seja considerada desafiadora e que desenvolvedores se sintam pouco motivados para realizá-la. Neste artigo, é apresentado um estudo de caso que visa caracterizar uma primeira versão de condutores de carga cognitiva presentes nas atividades de testes unitários de software. Por meio de entrevistas conduzidas com desenvolvedores de diferentes equipes de um banco brasileiro, foi identificado um conjunto de desafios e problemas enfrentados para a realização de testes unitários. A partir dessas informações, foi codificado um conjunto de condutores de carga cognitiva que abrange diferentes aspectos técnicos e não técnicos da prática. Além disso, os resultados do estudo destacam a necessidade de um processo padronizado para apoiar e promover a prática de testes unitários nas organizações.*

1. Introdução

Testar um software significa executá-lo com um conjunto pré-selecionado de entradas e verificar se este se comporta da maneira esperada.[Fraser and Rojas 2019]. O teste unitário, também conhecido como teste de unidade, é uma forma de teste escrita para pequenas unidades de código para o software que está sendo desenvolvido [Kayongo et al. 2016]. Em um ciclo de desenvolvimento do software, o teste unitário é o primeiro nível de teste a ser realizado, e é tipicamente escrito e executado pelo próprio desenvolvedor do código. A possibilidade de desenvolvedores identificarem falhas ao longo de suas atividades reduz o custo e esforço necessários para corrigir falhas que sejam encontradas

em etapas posteriores do desenvolvimento [Pargaonkar 2023]. Além disto, há outros benefícios na prática de testes unitários, como por exemplo contribuir para a documentação do sistema [Shamshiri et al. 2015].

O teste unitário é largamente reconhecido como um importante e valioso meio de melhorar a confiabilidade do software, ao detectar antecipadamente bugs durante o seu ciclo de desenvolvimento. [Xie et al. 2016]. Contudo, alguns desenvolvedores consideram a prática de testes unitários tediosa e se sentem pouco motivados para realizá-la [Saloum and Rissanen 2019]. Quando aplicados, os testes unitários costumam ser executados com apoio de ferramentas de automação. Entretanto, estas ferramentas apresentam limitações de suporte cognitivo para os desenvolvedores [Prado and Vincenzi 2018]. Assim, entende-se que as limitações observadas na realização de testes unitários estão relacionadas à carga cognitiva embutida nessa prática.

A carga cognitiva pode ser definida como a quantidade de informação que a mente humana pode processar de uma só vez durante a aprendizagem [Passos 2020]. Cada tarefa possui sua carga cognitiva específica, que varia conforme a sua complexidade. Um dos grandes desafios é fornecer meios de aprendizagem que permitam a retenção de novas informações na memória de longo prazo [Sweller 2011]. Ao introduzir a teoria da carga cognitiva, Sweller demonstrou que a sobrecarga cognitiva pode reduzir a capacidade de processamento de informações e o desempenho na execução de tarefas.

Uma forma de otimizar o esforço na realização de uma tarefa é minimizar a carga cognitiva envolvida, identificando seus condutores de carga. Considerando a complexidade inerente às atividades de desenvolvimento de software em larga escala, Hegelsson et al. [Helgesson et al. 2019] propuseram a investigação de *condutores de carga cognitiva (cognitive load drivers)*. Condutores de carga cognitiva podem ser definidos como os fatores técnicos e não-técnicos que promovem a carga cognitiva em determinada tarefa. [Belém et al. 2023].

Dentro deste contexto, identificar os condutores da carga cognitiva na prática de testes unitários é um importante recurso para direcionar o desenvolvimento de futuras tecnologias que possam mitigar esta carga, otimizando o esforço dos desenvolvedores e estimulado a prática de testes unitários.

Neste artigo, é apresentado um estudo inicial visando caracterizar os condutores de carga cognitiva envolvidos na prática de testes unitários. Por meio de entrevistas conduzidas com seis desenvolvedores que atuam no departamento de TI de uma organização brasileira, foi identificado um conjunto inicial de condutores, organizados em clusters e temas. No entanto, mais estudos são necessários para refinar esses condutores. Neste sentido, pretende-se replicar o estudo com desenvolvedores de outras organizações.

2. Trabalhos Relacionados

A Teoria da Carga Cognitiva, criada por John Sweller no final da década de 1980, define o uso da teoria evolucionária para considerar a arquitetura cognitiva humana e usa essa arquitetura para conceber procedimentos instrucionais novos[Sweller 2011]. Em trabalhos posteriores, Sweller estuda possíveis maneiras de diminuir esta carga cognitiva, abordando maneiras que facilitam o aprendizado de novas tarefas por estudantes.

A literatura técnica apresenta diversos trabalhos que focam na medição da carga

cognitiva de tarefas de programação. Por exemplo, Stolp [Stolp 2023] sugere a utilização de sensores corporais para a medição da carga cognitiva relacionada à compreensão de código. Em geral, estes trabalhos levam em consideração apenas aspectos isolados da programação, tipicamente voltados para a compreensão e escrita de código, ignorando diferentes dimensões da carga cognitiva. Além disto, a forma de medir a carga cognitiva entre os desenvolvedores de software ainda é dispersa [Gonçales et al. 2019]. Gonçales et al. afirmam que dentre os 33 trabalhos selecionados para sua pesquisa, 55% adotaram o eletroencefalograma (EEG) para monitorar a carga cognitiva, 51% destes estudos aplicaram algoritmos de machine learning para prever esta carga e 48% mediram esta carga no contexto de tarefas de programação. Por outro lado, não identificamos trabalhos que investiguem a carga cognitiva na prática de testes unitários.

A exemplo de [Helgesson et al. 2019], entende-se que é necessário identificar os condutores de carga cognitiva para compreender melhor a multidimensionalidade do impacto desta carga. Por entrevistas com membros de um time de desenvolvimento, Helgesson identificou três principais *clusters* de condutores envolvidos durante a fase de desenvolvimento do software: *ferramenta* (carga cognitiva ligada diretamente ao uso de ferramentas necessárias para a execução da tarefa), *informação* (carga cognitiva associada ao gerenciamento do fluxo das informações dentro da equipe) e *processo de trabalho* (carga relacionada a processos desenhados de forma ineficiente). Com base na experiência de Helgesson et al., foi conduzido um novo estudo visando caracterizar os condutores de carga cognitiva presentes nas atividades de manutenção evolutiva de software em equipes remotas [Belém et al. 2023]. Os autores identificaram o novo cluster *comunicação interna*, além de novos condutores. A Tabela 1 apresenta um exemplo de condutores de carga cognitiva identificados em trabalhos anteriores, para o cluster de comunicação interna.

Tema	**Condutores**
Fluxo da comunicação	Comunicação da equipe deficiente Equipe com horários assíncronos
Apoio e acompanhamento	Indisponibilidade para tirar dúvidas Falta de feedback

Tabela 1. Condutores de carga cognitiva para cluster *Comunicação Interna* [Belém et al. 2023].

É amplamente reconhecido que a execução de testes unitários é uma das mais efetivas maneiras de identificar defeitos [Bandara and Perera 2020]. Além da redução do custo do software pela detecção antecipada de erros, os testes unitários também contribuem para acelerar o desenvolvimento do produto, sendo um recurso importante para viabilizar a entrega de uma solução funcional dentro dos prazos estabelecidos [Runeson 2006].

Através de uma pesquisa de opinião, [Santos et al. 2023] identificou que os desenvolvedores tendem a reconhecer a relevância dos testes unitários. Entretanto, eles também se sentem pouco motivados para realizá-los no seu ambiente de trabalho, o que se reflete na sua cobertura. Este cenário demonstra uma dissonância entre o entendimento dos profissionais e seu comportamento, o que pode prejudicar seu comprometimento e gerar conflitos internos. Neste estudo, os autores também identificaram uma correlação positiva entre a motivação dos desenvolvedores para a realização de testes unitários e suas percepções sobre disponibilidade de treinamento e tempo alocado para a sua realização.

Sem o devido suporte, a prática de testes unitários pode demandar um esforço considerável dos desenvolvedores. Através de uma pesquisa de opinião, Prado et al [Prado et al. 2015] identificaram diferentes demandas de suporte cognitivo relacionadas a ferramentas de automação de teste. Neste sentido, os autores propuseram um arcabouço conceitual descrevendo estas demandas. Por outro lado, este presente estudo baseia-se nos dados de entrevistas para caracterizar os condutores de carga cognitiva na prática de testes unitários, considerando todo o contexto técnico e não-técnico envolvido nesta prática.

3. Estudo de Caso

Com base nas investigações anteriores de condutores de carga cognitiva [Helgesson et al. 2019, Belém et al. 2023] foi planejado um estudo de caso visando *caracterizar os condutores de carga envolvidos na prática de testes unitários de software*. Esta caracterização é realizada através da análise das dificuldades e problemas relatados pelos desenvolvedores ao realizar testes unitários, tendo como base inicial os condutores, temas e clusters propostos em estudos anteriores [Helgesson et al. 2019, Belém et al. 2023]. Assim, foram definidas as seguintes questões de pesquisa:

RQ1. *Quais os problemas e dificuldades enfrentados por uma equipe de desenvolvedores de software na realização de testes unitários?*

RQ2. *Quais são os condutores de carga cognitiva que estão relacionados aos problemas e dificuldades relatados por estes desenvolvedores?*

Neste estudo, consideramos a realização de testes unitários como todas as atividades realizadas pelos desenvolvedores relacionadas ao tema, incluindo a concepção dos testes unitários, sua implantação, validação, (re-)execução, e integração com os demais artefatos de software. Para responder a estas questões, foi conduzido um estudo de caso exploratório elaborado da seguinte forma:

- Revisão da Literatura: definição teórica de carga cognitiva, conceituação de testes unitários de software, identificação de trabalhos anteriores sobre carga cognitiva na engenharia de software e sobre testes unitários.
- Execução da Entrevista: condução de estudo exploratório visando identificar dificuldades/problemas identificados por desenvolvedores na realização de testes unitários de software.
- Codificação da Entrevista: com base na literatura e nas respostas obtidas dos desenvolvedores, codificar os condutores de carga identificados na entrevista, agrupando-os nos temas/clusters propostos pela literatura ou propondo novos.

3.1. Contexto do Estudo

Para a coleta de dados, foram conduzidas entrevistas semiestruturadas, visando obter informações detalhadas sobre os desafios e dificuldades enfrentados pelos desenvolvedores. O público-alvo do estudo inclui profissionais de software que possuem experiência na prática de testes unitários. Como amostra, nós recrutamos seis desenvolvedores que atuam do departamento de Tecnologia da Informação de um banco brasileiro. Todos preencheram um formulário de caracterização do participante, o qual incluía um termo de consentimento e a confirmação de conhecimento prático das atividades de desenvolvimento de softwares e execução de testes unitários.

Dos seis entrevistados (E1-E6), cinco relataram exercer o papel de desenvolvedor, enquanto o outro relatou exercer a função de *tech leader* de testes, ou seja, é o especialista responsável por estabelecer políticas e métricas relacionadas ao assunto testes de software na organização. Este *tech leader* atua como consultor em todos os níveis de teste para equipes que precisam de apoio em tarefas relacionadas a esta atividade. Os outros cinco participantes trabalham em equipes distintas, sendo que todas adotam práticas de métodos ágeis de desenvolvimento de software. Uma destas equipes baseia-se nas práticas do Kanban, enquanto as demais seguem o método Scrum. Dentre as ferramentas utilizadas para execução dos testes unitários, foram citadas para a linguagem Javascript a ferramenta JTest (4 entrevistados) e Jasmine (2 entrevistados) e, para linguagem Java, a ferramenta Junit (2 entrevistados).

Nas equipes da maioria dos entrevistados (E1, E2, E3 e E5), a prática de testes unitários está incluída no critério de aceite das tarefas selecionadas pelas equipes durante a reunião de planejamento. No caso da equipe do entrevistado E4, caso a funcionalidade seja muito complexa, há um desmembramento entre as tarefas de desenvolvimento e de testes unitários. Na equipe do entrevistado E6, a atividade de teste unitário não é exigida como critério de aceite de uma tarefa, ficando a cargo do desenvolvedor executar ou não estes testes.

3.2. Entrevista

A seguir, é listado o script planejado para a entrevista. As perguntas foram elaboradas considerando as questões da pesquisa e trabalhos anteriores investigando a prática de testes unitários [Prado and Vincenzi 2018, Santos et al. 2023]. Antes do inicío da entrevista foi contextualizado para os participantes que o estudo foca especificamente nas atividades inerentes à prática de testes unitários de softwares (excetuando-se outros tipos de testes), englobando todas as ferramentas, atividades e processos empregados por cada entrevistado na execução destas tarefas. As primeiras perguntas da entrevista focam no suporte e nas dificuldades/problemas enfrentados pelo entrevistado para a realização dos testes unitários (perguntas 1, 2 e 3). As quatro perguntas seguintes estão relacionadas à percepção do profissional sobre a como a prática de testes unitários é apoiada pela equipe do entrevistado, considerando seus pares e lideranças (perguntas 4, 5, 6 e 7). Finalmente, a pergunta 8 aborda a integração dos testes unários aos ciclos de entrega da equipe.

1. Quais ferramentas tecnológicas você utiliza para realizar testes unitários? Quais atividades são realizadas com estas ferramentas em seu ambiente de trabalho?
2. Quais são as maiores dificuldades que você já identificou durante a condução dos testes unitários?
3. Descreva resumidamente quais ações poderiam minimizar as dificuldades (ou problemas) identificados na condução dos testes unitários?
4. Qual sua opinião sobre como uma tarefa de teste unitário é apresentada para sua equipe, tanto na sua descrição formal quanto na alocação?
5. Você entende que a prática de execução dos testes unitários é apoiada pela gerência ou líder técnico da sua equipe?
6. Existe colaboração entre os membros das equipes na escrita, revisão e execução de testes unitários em um ambiente ágil? Como você avalia esta colaboração? E por quê?

7. Como você pede ajuda, caso necessário, quando está trabalhando em uma tarefa de testes unitários?
8. Como os testes unitários são integrados aos ciclos de entrega na sua equipe?

4. Execução e Resultados

Todas as entrevistas foram gravadas e transcritas utilizando a ferramenta Microsoft Teams. Durante a execução das entrevistas, todos os participantes demonstraram conhecimento sobre a prática de testes unitários de software e não tiveram problemas em responder às perguntas, com exceção da pergunta sobre como uma tarefa de teste unitário é apresentada para a equipe, que exigiu uma breve explicação para um dos entrevistados. Para a codificação dos dados transcritos, foi utilizada a análise temática, que pode ser entendida como um conjunto de técnicas utilizadas para analisar dados textuais e extrair temas [Vaismoradi et al. 2016]. O material transcrito foi codificado (nível 1) individualmente por dois autores do estudo. Em seguida, os códigos foram agrupados e filtrados, sendo eliminados aqueles repetidos pelo mesmo entrevistado. A partir desta filtragem, os códigos foram categorizados (nível 2). Foi feito ainda um novo refinamento onde códigos similares foram agrupados em um único código, chegando-se na versão final de códigos categorizados. Para responder a RQ2, foram inicialmente identificados códigos que poderiam estar associação aos condutores de carga cognitiva propostos por [Helgesson et al. 2019] e [Belém et al. 2023]. Na falta desta associação, foram propostos novos condutores, temas ou mesmo clusters.

4.1. Problemas e Dificuldades (RQ1)

Com base na transcrição das entrevistas, foram identificados inicialmente 17 códigos diferentes referentes a problemas e dificuldades enfrentados pelos desenvolvedores na prática de testes unitários (nível 1). Estes códigos foram então agrupados em cinco categorias (nível 2). Após o refinamento, alguns códigos foram agrupados por similaridade, resultando em 12 códigos distintos divididos pelo mesmo número de categorias, conforme apresentado na Tabela 2.

Com base nos códigos e categorias identificados, podemos destacar que a falta de padronização do processo entre as equipes é a maior dificuldade percebida pelos entrevistados. Os entrevistados relatam que os critérios e passos necessários para realização de testes unitários ficam a critério de cada equipe. Assim, dependendo da equipe que o desenvolvedor esteja alocado no momento, o processo da execução de tarefa de teste poderá ser diferente, o que demanda um maior esforço do profissional para (re-)adaptação a este processo.

Foram identificados problemas relacionados ao uso de ferramentas para automação de testes unitários. Algumas ferramentas disponíveis não estão preparadas para geração de testes eficientes em sistemas legados, enquanto alternativas mais adequadas são pagas e não são adquiridas pela organização por este motivo. A dificuldade na utilização destas ferramentas também foi evidenciada por [Prado et al. 2015] que citou a falta de suporte cognitivo e dificuldade de uso de ferramentas de automação de testes. Verificou-se também uma dificuldade relacionada à falta de conhecimento sobre a prática de testes utilizando a técnica de TDD (Test-Driven Development). Embora não exista nenhuma orientação formal da organização, foi afirmado por alguns entrevistados que a equipe deve desenvolver testes

Categorias	Códigos
Especificação Da Tarefa e do Sistema (E1,E2,E4)	- Regra de negócio mal especificada(E1) - Simulação do comportamento do usuário(E1) - Dificuldade em compreender código escrito(E2,E4)
Recursos Disponíveis (E2,E4,E6)	- Software disponível ineficiente(E2,E4) - Falta de ambiente pré-configurado para testes (E4,E6) - Falta de equipe de suporte para testes(E4)
Definição de Processo (E1,E2,E3,E4,E5,E6)	- Processo não padronizado (E1,E2,E3,E4,E5,E6) - Testes preteridos em função do prazo(e2,E3,E4,E6) - Dificuldade em entender o processo de teste (E1,E2,E3,E6)
Necessidades de Comunicação (E1)	- Dificuldades para conseguir colaboração(E1)
Conhecimento Técnico (E1,E3,E5)	- Falta de conhecimento em TDD (E1,E3,E5) - Falta de experiência em TDD(E1,E3,E5)

Tabela 2. Problemas e Dificuldades Identificados

unitários utilizando esta técnica, o que representa um desafio para aqueles acostumados à prática tradicional de elaboração de testes unitários após a escrita do código. Assim, esta mudança de prática foi identificada por alguns entrevistados como causadora de maior esforço para a realização de testes unitários. Além desta falta de conhecimento, também foi apontada falta de experiência prática com TDD. Em trabalho anterior, tanto a falta de conhecimento quanto a falta de experiência, já foram considerados obstáculos para a ampla adoção de TDD na indústria[Causevic et al. 2011].

Também foram observados problemas nas informações presentes nos documentos de requisitos, onde são especificadas as regras de negócios necessárias para serem desenvolvidas e testadas. Estes problemas intensificam as necessidades de comunicação, onde o desenvolvedor precisa pedir ajuda ao líder técnico ou até mesmo procurar o usuário diretamente para elucidar dúvidas que auxiliem na escrita dos testes. É comum que o retorno a esta consulta não seja imediato, o que pode levar a atrasos no desenvolvimento. Outras dificuldades relatadas por parte dos desenvolvedores estão relacionadas à dificuldade de compreensão das regras de negócio e do próprio código escrito, impactando a identificação e elaboração de cenários de teste a serem cobertos pelos testes unitários.

4.2. Condutores de carga cognitiva relacionados aos problemas e dificuldades identificados (RQ2)

Para identificar os condutores de carga cognitiva relacionados à atividade de testes unitários de software, buscou-se inicialmente relacionar as dificuldades e problemas codificados aos condutores mapeados por [Helgesson et al. 2019] e [Belém et al. 2023]. Como resultado, constatou-se que alguns não estavam relacionados aos condutores presentes nestas pesquisas. Consequentemente, novos temas e condutores foram criados, embora não tenha sido identificada a necessidadede criar novos clusters.

A Tabela 3 apresenta o novo tema *software*, referente ao cluster *ferramenta*. Condutores e temas novos são indicados com um asterisco. Para este tema, foi proposto o condutor *limitações de software* em virtude das dificuldades encontradas pelos entrevista-

dos para a execução de testes unitários utilizando as ferramentas de automação disponíveis. Considerando o relato dos entrevistados, é possível identificar que estas limitações passam pelo uso de ferramentas que produzam resultados adequados (”*Mas eu ainda não vi nenhuma que me gere um teste a contento”*, E2), pela falta de investimento em ferramentas adequadas (*”...ou é paga e não pode usar, né?”*, E4) e por necessidades de adaptação conforme o sistema operacional (*”eu tô tendo a dificuldade aqui de baixar uma biblioteca específica do Linux, coisa que para o Windows ele já tem”*, E4). Estas circunstâncias exigem dos desenvolvedores uma carga cognitiva adicional para adaptar e improvisar a prática de testes unitários, o que impacta sua fluidez e continuidade.

No cluster processo de trabalho, propomos ainda o novo tema *habilidades pessoais* (Tabela 4), considerando as dificuldades reconhecidas por alguns entrevistados com a prática de testes unitários, em particular com TDD. Para este tema, foram identificados o novo condutor *falta de conhecimento técnico* (”*às vezes é difícil... admitir para mim uma dificuldade, eu tento implementar o tdd, né, que é tentar imaginar o teste antes do desenvolvimento em si... admito que nem sempre é fácil*”, E1) (”*Acho que a principal dificuldade é essa. É, é conhecimento mesmo. E falta de cultura de teste também.*”E5) e o novo condutor *falta de experiência prática* (” *eu não sou, assim, nenhuma pessoa de muita experiência em teste... tive que ir aprendendo*.”, E3) (” *tentar imaginar o teste antes do desenvolvimento em si... eu tento botar isso em prática, admito que nem sempre é fácil*”, E1). É natural que a falta de conhecimento e de experiência exijam do profissional um maior esforço cognitivo, o que se espera que seja atenuado com o aprendizado da prática. No entanto, esta curva de aprendizado pode ser mais persistente no caso de testes unitários, que frequentemente carecem de treinamento e priorização [Santos et al. 2023]. Neste sentido, cabe destacar o fato de desenvolvedores experientes possuírem pouca experiência com uma atividade que é considerada intrínseca às suas atribuições.

Outro condutor encontrado foi *Conflitos na Priorização de Tarefas* onde ficou evidenciado que apesar da relevância da execução dos testes unitários, muitas vezes eles são preteridos caso haja pouco tempo para entrega de determinada tarefa (*”as pessoas falam muito bonito você cobrir o teste desde que não atrapalhe a entrega ou prazo”*E2) (*”Se ela for muito crítica e demorada, muito necessária para um momento, a gente faz o desenvolvimento e o teste unitário colocamos em uma segunda etapa, né?”*E4).

67% dos problemas e dificuldades codificados foram associados a condutores já identificados em trabalhos anteriores [Helgesson et al. 2019, Belém et al. 2023]. Ainda no cluster *processo de trabalho*, o condutor *processos desperdiçados* é representado pela falta de uma definição de um processo de testes unitários padronizado dentro da organização. Embora haja uma diretriz corporativa que estipule uma cobertura mínima de código (indicando que ”*A orientação corporativa é 60% de código coberto*- E2), a condução dessas atividades é deixada à discrição de cada equipe (”*a gente acabou colocando isso na definição de pronto*- E2). Algumas equipes estabelecem critérios adicionais, como considerar uma funcionalidade concluída somente quando estiver acompanhada de testes automatizados (”*...só vai estar pronto quando estiver com os testes automatizados, não considera a funcionalidade pronta se não tiver teste automatizado nela...*- E5), mas a falta de um roteiro ou orientação explícita torna o processo bastante heterogêneo (”*Não existe nenhum roteiro ou nenhuma orientação. Só fala o que você tem que testar, a funcionalidade.*- E4). Embora algumas equipes adotem o Desenvolvimento Orientado a

Testes (TDD), isso não é uma prática generalizada ou oficialmente recomendada (”*Assim, eu sei que existem equipes que atuam com TDD, mas não é regra e não existe no momento isso disseminado como uma prática a ser seguida*- E5). Essa falta de consistência pode resultar em situações em que um desenvolvedor experiente em testes de uma equipe enfrenta dificuldades ao ser realocado, devido à disparidade nos processos entre as equipes, o que aumenta a carga cognitiva e a necessidade de adaptação.

Tema	**Condutor**	**Problemas/Dificuldades Relacionados**
Software*	Limitações de software*	- software disponível ineficiente

Tabela 3. Condutores identificados para o cluster *Ferramenta*

Tema	**Condutor**	**Problemas/Dificuldades Relacionados**
Falta de Suporte	Processos Desperdiçados	- falta de processo padronizado - dificuldade em entender o processo de teste
	Falta de automação	- falta de ambiente pré-configurado para testes
	Esforço desnecessário/redundante	- falta de equipe de suporte para testes
Habilidades Pessoais*	Falta de conhecimento técnico*	- falta de conhecimento em TDD
	Falta de experiência prática*	- falta de experiência em TDD
Ritmo de Trabalho	Conflitos na Priorização de Tarefas*	- testes preteridos em função do prazo

Tabela 4. Condutores identificados para o cluster *Processo de Trabalho*.

Outras dificuldades relatadas pelos entrevistados foram associadas aos condutores do cluster *informação* (Tabela 5). Condutores identificados para este cluster referem-se a dificuldades de entendimento das informações que permitirão a escrita dos testes. Neste contexto, o condutor *informação incompleta* refere-se a regras de negócios mal definidas, que pode levar o desenvolvedor a ter uma interpretação equivocada da necessidade do usuário (*”... dificuldade ali, às vezes, no entendimento do que está sendo pedido na história, na tarefa... às vezes a gente tem uma visão do que é preciso, mas às vezes isso diverge do que o cliente está pedindo.”* E1”). O condutor *dificuldade em localizar a informação* tem a ver com a falta de compreensão de códigos legados, exigindo maior esforço na identificação de como a tarefa deve ser realizada (*”... quando vai mexer com código legado, às vezes a gente tem um pouco de dificuldade... quando o código não foi escrito pensando em testar, ele às vezes é tão complexo”* E2). Para o cluster *Comunicação Interna* (Tabela 6), foi relatada a dificuldade para conseguir colaboração no processo de execução dos testes(*”Às vezes é meio complicado conseguir... quando a gente tem alguma dificuldade, ele já está mais aprofundado no assunto e também ajuda a gente nisso.”*, E1).

Tema	Condutor	Problemas/Dificuldades Relacionados
Integridade da Informação	Informação incompleta	- regra de negócio mal especificada - simulação do comportamento do usuário
Organização da Informação	Dificuldade em localizar a informação	- dificuldade em compreender código escrito

Tabela 5. Condutores identificados para o cluster *Informação*.

Tema	Condutor	Problemas/Dificuldades Relacionados
Apoio e encaminhamento	Indisponibilidade para tirar dúvidas	- dificuldades para conseguir colaboração

Tabela 6. Condutores identificados para o cluster *Comunicação Interna*.

5. Conclusão

Testes unitários são uma importante atividade de desenvolvimento que pode exigir considerável esforço dos profissionais desenvolvedores. No estudo de caso apresentado neste artigo, foi identificado um primeiro conjunto de condutores de carga cognitiva relacionados à prática de testes unitários.

Os condutores apontados estão vinculados às questões de apoio ferramental, processo de trabalho, informação e comunicação interna. No entanto, foi observado uma maior ênfase e diversidade de condutores relacionados ao processo de trabalho, incluindo a falta de suporte, de habilidades pessoais e dificuldades causadas pela ausência de priorização da atividade. Percebe-se que os desenvolvedores sentem falta de uma padronização organizacional das atividades relacionadas aos testes unitários, contrastando com as metas de cobertura exigidas pela organização.

Como trabalhos futuros, planeja-se refinar o conjunto de condutores identificados através da reexecução do estudo com profissionais de organizações com características diferentes à investigada. Através do mapeamento destes condutores, será possível propor e desenvolver tecnologias que apoiem a medição da carga cognitiva na prática de testes unitários de software. Além disto, poderemos direcionar a identificação e desenvolvimento de soluções baseadas em evidência que atenuem a carga cognitiva envolvida nestas atividades.

Disponibilidade dos Artefatos

`https://anonymous.4open.science/r/CargaCognitivaTestesUnitarios-DD84/`

Referências

Bandara, R. and Perera, I. (2020). Unit test code generation tool support for lower level programming languages. In *2020 Moratuwa Engineering Research Conference (MERCon)*, pages 1–6. IEEE.

Belém, P., De Mello, R., Vivacqua, A. S., and Neves De Souza, A. (2023). Investigating the cognitive load drivers of software evolution activities. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*, pages 342–347.

Causevic, A., Sundmark, D., and Punnekkat, S. (2011). Factors limiting industrial adoption of test driven development: A systematic review. In *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*, pages 337–346. IEEE.

Fraser, G. and Rojas, J. M. (2019). Software testing. *Handbook of Software Engineering*, pages 123–192.

Gonçales, L., Farias, K., da Silva, B., and Fessler, J. (2019). Measuring the cognitive load of software developers: A systematic mapping study. In *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, pages 42–52. IEEE.

Helgesson, D., Engström, E., Runeson, P., and Bjarnason, E. (2019). Cognitive load drivers in large scale software development. In *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 91–94. IEEE.

Kayongo, P., Chigona, W., and Mabhena, Z. (2016). Why do software developers practice test-driven development? In *2016 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, pages 357–361.

Pargaonkar, S. (2023). A comprehensive research analysis of software development life cycle (sdlc) agile & waterfall model advantages, disadvantages, and application suitability in software quality engineering. *International Journal of Scientific and Research Publications (IJSRP)*, 13(08).

Passos, E. R. W. (2020). Princípios da teoria da carga cognitiva voltados à educação corporativa.

Prado, M. P., Verbeek, E., Storey, M.-A., and Vincenzi, A. M. (2015). Wap: Cognitive aspects in unit testing: The hunting game and the hunter's perspective. In *2015 ieee 26th international symposium on software reliability engineering (issre)*, pages 387–392. IEEE.

Prado, M. P. and Vincenzi, A. M. R. (2018). Towards cognitive support for unit testing: A qualitative study with practitioners. *Journal of Systems and Software*, 141:66–84.

Runeson, P. (2006). A survey of unit testing practices. *IEEE software*, 23(4):22–29.

Saloum, S. and Rissanen, F. (2019). The impact of gamification in unit testing.

Santos, L., Santos, F., Parreira, R., and de Mello, R. (2023). Investigating the developer's perceptions of unit testing and its practice. In *Anais da VII Escola Regional de Engenharia de Software*, pages 238–247. SBC.

Shamshiri, S., Just, R., Rojas, J. M., Fraser, G., McMinn, P., and Arcuri, A. (2015). Do automatically generated unit tests find real faults? an empirical study of effectiveness and challenges (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 201–211.

Stolp, F. (2023). Assessing cognitive load in software development with wearable sensors. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 227–229. IEEE.

Sweller, J. (2011). Cognitive load theory. In *Psychology of learning and motivation*, volume 55, pages 37–76. Elsevier.

Vaismoradi, M., Jones, J., Turunen, H., and Snelgrove, S. (2016). Theme development in qualitative content analysis and thematic analysis.

Xie, T., Tillmann, N., and Lakshman, P. (2016). Advances in unit testing: theory and practice. In *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16, page 904–905, New York, NY, USA. Association for Computing Machinery.