UNIVERSIDADE DO VALE DO RIO DOS SINOS — UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA
NÍVEL MESTRADO

ROGER DENIS VIEIRA

COGNIDE: UMA ABORDAGEM PARA INTEGRAÇÃO DE DADOS PSICOFISIOLÓGICOS EM AMBIENTES INTEGRADOS DE DESENVOLVIMENTO

SÃO LEOPOLDO
2021

Roger Denis Vieira

COGNIDE: UMA ABORDAGEM PARA INTEGRAÇÃO DE DADOS PSICOFISIOLÓGICOS EM AMBIENTES INTEGRADOS DE DESENVOLVIMENTO

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos — UNISINOS

Orientador:
Prof. Dr. Kleinner Silva Farias de Oliveira

São Leopoldo
2021

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)

Vieira, Roger Denis

CognIDE: Uma abordagem para integração de dados psicofisiológicos em ambientes integrados de desenvolvimento / Roger Denis Vieira — 2021.

151 f.: il.; 30 cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, 2021.

"Orientador: Prof. Dr. Kleinner Silva Farias de Oliveira".

1. Engenharia de Software. 2. Neurociências. 3. Interação Cérebro-computador. 4. Análise de Dados. I. Título.

CDU 364

Bibliotecária: Silvana Dornelles Studzinski — CRB 10/2524

(Esta folha serve somente para guardar o lugar da verdadeira folha de aprovação, que é obtida após a defesa do trabalho. Este item é obrigatório, exceto no caso de TCCs.)

BOLSAS E AUXÍLIOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Código de Financiamento 001.

Aos que – frequentemente – duvidam de si mesmos.

Aquele que luta com monstros deve acautelar-se para não se tornar também um monstro.
Quando se olha muito tempo para um abismo, o abismo olha para você.
— FRIEDRICH NIETZSCHE

AGRADECIMENTOS

Gostaria de agradecer a minha mãe que, apesar de não possuir um nível de estudo avançado, sempre me proporcionou conforto para poder estudar e alcançar os meus objetivos. Também a minha noiva, pela estabilidade emocional que me ofereceu, sempre ouvindo as minhas reclamações e me trazendo paz em momentos de caos, além disso, agradeço aos meus amigos que, mesmo sendo de contextos diferentes, sempre proporcionaram debates acirrados e fortes argumentações em torno de diversas temáticas. Por fim gostaria de agradecer ao meu orientador, responsável por conduzir-me à concessão da bolsa CAPES-PROSUP, sem a qual não seria possível desenvolver meus estudos, além disso, agradeço por estar disponível quando necessário e oferecer diretrizes precisas que possibilitaram o desenvolvimento deste estudo. Desde já parabenizo o grupo TUG Unisinos[1], que confeccionou e disponibilizou o modelo LaTeX utilizado neste estudo, permitindo assim uma escrita muito mais produtiva. Por fim, agradeço a todos àqueles que compreenderam a minha ausência e flutuações de humor durante o desenvolvimento desta pesquisa.

[1] https://sites.google.com/site/tugunisinos/

"*Still, the words, "this is a crazy idea," echoed in our minds.*".
(SIEGMUND et al., 2020)

RESUMO

A busca pelo entendimento de como o cérebro do desenvolvedor comporta-se durante o desenvolvimento de software tem se mostrado como um objeto de estudo de crescente interesse nos últimos anos. Apesar do desenvolvimento de novos estudos coletando dados psicofisiológicos de desenvolvedores e os utilizando em experimentos, pouco se tem explorado acerca da aplicabilidade de tais dados na engenharia de software. Portanto, o desenvolvimento deste trabalho tem por objetivo propor uma ferramenta para a integração de dados psicofisiológicos em ambientes integrados de desenvolvimento, de modo a apresentá-los aos desenvolvedores e verificar o seu impacto no processo de desenvolvimento de software. Sendo assim, a ferramenta CognIDE foi desenvolvida baseada nas oportunidades identificada por intermédio da condução de uma mapeamento sistemático da literatura, onde foram identificados 2084 estudos, dos quais 27 foram selecionados como estudos primários. Para a avaliação da ferramenta proposta, foi desenvolvido e executado um experimento controlado entre 61 indivíduos da área de tecnologia, visando avaliar o impacto de se apresentar, na IDE, a métrica de Carga Cognitiva sobre as suas Percepções de Anomalias e Níveis de Intenção de Refatoração. Como resultados observou-se que apresentar métrica de Carga Cognitiva, quando esta apresenta o seu valor como Alta, aliada ao número de Anomalias de Códigos, pode auxiliar o desenvolvedor na identificação de anomalias, além de servir de insumo para a decisão de se refatoração trechos de código-fonte. O desenvolvimento deste trabalho, trouxe como principais contribuições: (1) expansão do estado-da-arte relativo à integração de dados psicofisiológicos em IDEs e sua aplicabilidade na engenharia de software; (2) a implementação da ferramenta CognIDE e sua abordagem para a integração de dados; (3) conhecimento empírico sobre o impacto de se exibir dados psicofisiológicos dos desenvolvedores nas IDEs.

Palavras-chave: Engenharia de Software. Neurociências. Interação Cérebro-computador. Análise de Dados.

ABSTRACT

The search for an understanding of how the developer's brain behaves during software development is an object of study of increasing interest in recent years. Despite the development of new studies collecting psychophysiological data from developers and using them in experiments, little has been explored about the applicability of such data in software engineering. Therefore, the development of this work aims to propose a tool for the integration of psychophysiological data in integrated development environments, to present them to developers, and verify their impact on the software development process. Therefore, the tool CognIDE was developed based on the opportunities identified by conducting a systematic mapping of the literature, where 2084 studies were identified, of which 27 were selected as primary studies. For the evaluation of the proposed tool, a controlled experiment was carried out and executed among 61 individuals in the technology area, aiming to assess the impact of presenting, in the IDE, the Cognitive Load metric on their Anomalies Perceptions and Levels of Refactoring Intention. As a result, it was observed that presenting the Cognitive Load metric, when it presents its value as High, combined with the number of Code Anomalies, can assist the developer in the identification of anomalies, in addition to serving as input for the decision of refactoring excerpts from source code. The development of this work brought as main contributions: (1) expansion of the state-of-the-art regarding the integration of psychophysiological data in IDEs and its applicability in software engineering; (2) the implementation of the CognIDE tool and its approach to data integration; (3) empirical knowledge about the impact of displaying developers' psychophysiological data in IDEs

Keywords: Software Engineering. Neuroscience. Brain-computer Interaction. Data Analysis.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

etc.	Et cetera
Hz	Hertz
min.	Mínimo
max.	Máximo
N. do A.	Nota do Autor

LISTA DE SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ADS	Análise e Desenvolvimento de Sistemas
ANOVA	Analysis of VAriance
API	Application Programming Interface
BLE	Bluetooh Low Energy
bps	bits per second
BVP	Blood Volume Pressu
BT	Bluetooth
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CC	Critério de Comparação
CE	Critério de Exclusão
CI	Critério de Inclusão
dB	decibel
DSL	Domain Specific Language
EDA	ElectroDermal Activity
EEG	Electroencephalography
EOG	Electrooculography
ERD	Event-related Desynchronization
ERS	Event-related Synchronization
ETL	Extract, Transform, Load
FAPERGS	Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul
fMRI	Functional Magnetic Resonance Imaging
fNIRS	Functional Near-infrared Spectroscopy
GQM	Goal Question Method
GSR	Galvanic Skin Response
HRV	Heart Rate Variability
HSD	Honestly Significant Difference
HTTP	Hypertext Transfer Protocol
IAF	Individual Peak Alpha Frequency
IDE	Integrated Development Environment
IoT	Internet of Things
JS	JavaScript
JSON	JavaScript Object Notation

KPI	Key Performance Indicator
LSP	Language Server Protocol
PP	Problema de Pesquisa
QE	Quality Engineer
QP	Questão de Pesquisa
REST	Representional State Transfer
RPC	Remote Procedure Call
SDK	Software Development Kit
SMS	Systematic Mapping Study
TCC	Teoria da Carga Cognitiva
TCP	Transmission Control Protocol
TR	Trabalho Relacionado
TS	TypeScript
TSDB	Time Series Database
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VS	Visual Studio
WPS	Waves per Second
XML	eXtensible Markup Language

LISTA DE SÍMBOLOS

α	Alfa
β	Beta
γ	Gama
δ	Delta
θ	Teta
$>$	Maior que
$<$	Menor que
$\leq$	Menor ou igual à
$\geq$	Maior ou igual à
$x..y$	Intervalo fechado de x até y
$\vert$	operador lógico ou
n	tamanho amostral

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de software, enquanto processo lógico e intelectual, demanda que cada desenvolvedor aplique o seu conhecimento em determinada tecnologia para que, sob o embasamento de premissas lógicas oriundas do domínio, possar produzir soluções voltadas para problemas reais na área corporativa, acadêmica, ou ainda, em situações do cotidiano. Tal desenvolvimento, além de demandar que o profissional domine a tecnologia utilizada e tenha conhecimento das regras de domínio, ainda exige que este esteja focado e imerso em seu trabalho, atentando a cada detalhe que possa impactar no comportamento da solução desenvolvida

Com o avanço da microeletrônica, o surgimento de dispositivos vestíveis capazes de coletar e processar sinais fisiológicos tornou-se possível (ROSTAMI et al., 2015). A popularização destes dispositivos, no que lhe concerne, difundiu-se no meio acadêmico, instigando pesquisadores a utilizarem-nos como ferramenta de coleta em experimentos científicos, em especial naqueles em que busca-se compreender como o cérebro de um desenvolvedor comporta-se durante o seu trabalho (RADEVSKI; HATA; MATSUMOTO, 2015).

Na última década, observou-se um aumento no número de estudos que buscam compreender quais processos cognitivos são apresentados pelo desenvolvedor durante a produção de software (GONCALES et al., 2019), para isto, autores utilizam tecnologias como Eletroencefalografia (EEG) (FRITZ et al., 2014; ROSTAMI et al., 2015; LEE et al., 2016, 2018; KOSTI et al., 2018; SEGALOTTO, 2018), Imagem por Ressonância Magnética Funcional (fMRI) (SIEGMUND et al., 2014, 2017; FLOYD; SANTANDER; WEIMER, 2017; PEITEK et al., 2018), Espectrografia Funcional de Infravermelho Próximo (fNIRS) (FISHBURN et al., 2014; FAKHOURY, 2018; FAKHOURY et al., 2020). Os autores destes estudos advogam que, embasados pelos dados psicofisiológicos coletados, seria possível associar determinados processos cognitivos, ou ainda, padrões de ativação neural, às características presentes no código-fonte, como, por exemplo, as variáveis declaradas sob nomenclatura dúbia, má formatação e apresentação do código-fonte, colorização de componentes sintáticos, entre outros.

Este capítulo visa apresentar os propósitos principais deste estudo, assim como a estrutura adotada para o seu desenvolvimento. Na Seção 1.1 é descrita a relação de problemas abordados e discutidos. Na sequência, a Seção 1.2 discorre acerca das questões de pesquisa que o este estudo visa responder. Os objetivos geral e específicos são apresentados na Seção 1.3 e a abordagem metodológica adotada para alcançá-los é discriminada na Seção 1.4. Por fim, a Seção 1.5 visa explanar as principais contribuições deste estudo e a Seção 1.6, a estrutura adotada para o seu desenvolvimento.

1.1 Problemática

Como apresentado anteriormente, a utilização de dados psicofisiológicos no contexto da engenharia de software, especialmente na execução experimentos buscando compreender o fun-

cionamento do cérebro dos desenvolvedores de software, tem se mostrado como um tópico em crescimento (GONCALES et al., 2019; SIEGMUND et al., 2020). Não obstante, os esforços aplicados nas pesquisas recentemente desenvolvidas, os experimentos realizados utilizam ferramentas para apresentação de código-fonte que não refletem o ambiente de trabalho real adotado pelo desenvolvedor, ainda que ferramentas como o CodersMUSE (PEITEK et al., 2019) e VITALSE (ARNAOUDOVA; FAKHOURY; ROY, 2020) apliquem uma abordagem multimodal para exploração de dados como os oriundos dos experimentos observados. Além disso, observa-se uma falta de consenso por parte dos autores na literatura atual sobre quais e como os processos cognitivos estão relacionados com o desenvolvimento de software (FLOYD; SANTANDER; WEIMER, 2017; GONCALES et al., 2019). Fundamento pelos pontos discutidos, este trabalho, portanto, terá como oportunidades explorar os seguintes problemas de pesquisa (PPs):

- **PP01: falta de uma visão abrangente acerca do estado da arte.** Na atual literatura, observa-se o desenvolvimento de estudos que buscam compreender como fatores fisiológicos e psicológicos se alteram durante as atividades de desenvolvimento de software e compreensão de código. Para isto, estes coletam dados psicofisiológicos de desenvolvedores e utilizam ambientes integrados de desenvolvimento e editores de código em seus experimentos. Apesar do interesse pela produção de estudos relativos à temática, observa-se ainda uma carência na literatura por mecanismos que estabeleçam protocolos bem definidos, estes aplicados à pesquisa e classificação de estudos desta natureza, de forma a apontar possíveis lacunas na literatura e fornecer norteadores para o desenvolvimento de pesquisa futuras.

- **PP02: carência de uma ferramenta capaz de coletar dados psicofisiológicos e integrá-los nas IDEs com o intuito de medir a atividade neural do desenvolvedor.** Nos estudos produzidos atualmente, nota-se que nos experimentos desenvolvidos, em especial os que visam a compreensão de trechos específicos de código-fonte, utilizam editores e/ou IDEs que não refletem a realidade dos desenvolvedores, implicando na perda de informação referentes ao contexto onde o código-fonte é apresentado. Sendo assim, é possível observar a carência de soluções capazes de integrar os dados psicofisiológicos coletados nos experimentos aos dados oriundos de eventos contextuais gerados pelas IDEs e/ou editores comerciais.

- **PP03: falta de conhecimento empírico sobre os efeitos da presença da métrica psicofisiológica de Carga Cognitiva nas IDEs.** Enquanto os experimentos observados nos estudos aplicam editores e/ou IDEs com funcionalidades reduzidas, ou inexistentes para a compreensão de código-fonte, outros trabalhos visam elucidar os impactos das IDEs e suas funcionalidades na produtividade dos desenvolvedores. Apesar destes esforços, não é possível notar uma intersecção comum entre tais naturezas de estudos, visando analisar

o impacto de se apresentar o indicador psicofisiológico de Carga Cognitiva em IDEs e a sua influência nas atividades de desenvolvimento de software.

1.2 Questões de Pesquisa

Como consequência da problemática discorrida anteriormente, faz-se necessário o desenvolvimento de uma ferramenta que cumpra a função de integrar os dados psicofisiológicos coletados em experimentos, àqueles oriundos dos eventos das IDEs. Tal desenvolvimento, portanto, possibilitaria a relação dos fenômenos observados a partir da análise dos dados psicofisiológicos e das IDEs. Sendo assim, haveria a possibilidade de análise acerca compreensão dos impactos das IDEs na atividade neural do desenvolvedor, sendo esta, uma atividade psicofisiológica. Embasado considerando tais argumentos, chegou-se à seguinte questão de pesquisa utilizada neste estudo:

> **Questão Geral de Pesquisa:** Como a presença do indicador psicofisiológico de Carga Cognitiva, quando apresentado em ambientes integrados de desenvolvimento, pode afetar no processo de desenvolvimento de software?

De forma a auxiliar na busca pela reposta à questão geral de pesquisa, outras três questões de pesquisa foram elaboradas, conforme descritas a seguir:

- **QP01:** Qual o estado-da-arte acerca da integração entre dados psicofisiológicos e IDEs?
- **QP02:** Como realizar a integração entre dados psicofisiológicos e IDEs?
- **QP03:** Como o indicador psicofisiológico de Carga Cognitiva pode influenciar no processo de desenvolvimento de software?

1.3 Objetivos

Uma vez identificada a problemática e o seu escopo (Seção 1.1), além das questões de pesquisa as quais deseja-se responder mediante o desenvolvimento do estudo (Seção 1.2), foi definido o seguinte objetivo de pesquisa geral:

> **Objetivo Geral:** Propor uma ferramenta para a integração entre dados psicofisiológicos coletados de desenvolvedores e IDEs reais, a fim apresentar tais métricas junto ao código fonte e verificar o seu impacto no processo de desenvolvimento de software.

De forma a servir como pivô para a condução do estudo e contemplação do objetivo geral de pesquisa, ainda foram definidos outros três objetivos específicos. A seguir estes são apresentados, seguidos de uma breve descrição:

- **OE-1: conduzir um Mapeamento Sistemático da Literatura acerca da integração de dados psicofisiológicos em IDEs.** Tem por finalidade produzir uma visão do estado-da-arte acerca da integração entre psicofisiológicos e IDEs em estudos experimentais. Para isto, serão coletados estudos já feitos, analisados, classificados e produzidas oportunidades de pesquisa com base nos resultados obtidos.

- **OE-2: propor e implementar uma ferramenta para a integração entre dados psicofisiológicos e IDEs.** Embasado pelas soluções já existentes apresentadas na literatura, implementar um mecanismo que seja capaz de coletar dados psicofisiológicos experimentais, assim como aqueles oriundos de eventos gerados pelas IDEs e armazená-las para a sua utilização em análises posteriores.

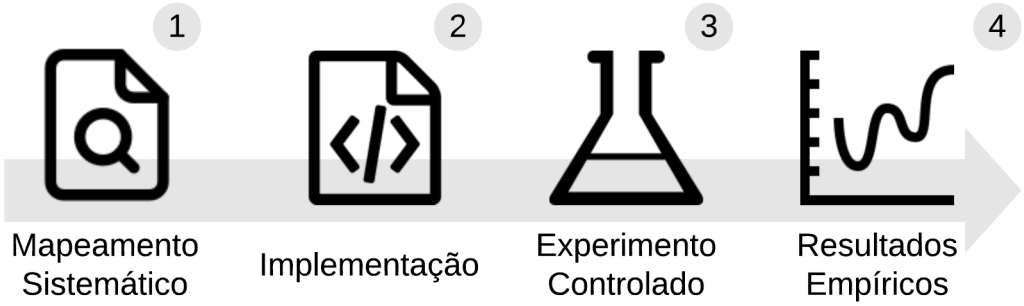
- **OE-3: produzir observações empíricas acerca da presença da métrica psicofisiológica de Carga Cognitiva nas IDEs e o seu impacto no desenvolvimento de software.** Através de um experimento controlado, visa apresentar o indicador psicofisiológico de Carga Cognitiva junto ao código-fonte produzido pelo desenvolvedor e avaliar os efeitos desta presença no processo de desenvolvimento de software.

  .

1.4 Abordagem Metodológica

Esta seção tem por objetivo explanar a metodologia utilizada durante o desenvolvimento do estudo, de forma a contemplar todos os objetivos específicos descritos na Seção 1.3. A Figura 1.4 ilustra as etapas adotadas na condução deste estudo.

Figura 1: Etapas adotadas no desenvolvimento do estudo.



Fonte: Adaptado de Segalotto (2018)

A primeira etapa deste estudo caracteriza-se pela revisão da literatura acerca da utilização de dados psicofisiológicos em experimentos de engenharia de software, além da adoção de IDEs e como estas impactam no trabalho dos desenvolvedores. Portanto, optou-se pela confecção de

um mapeamento sistemático da literatura, buscando assim, contemplar a primeira questão de pesquisa (PETERSEN; VAKKALANKA; KUZNIARZ, 2015).

Em seguida, a segunda fase é constituída pela implementação do CognIDE, a ferramenta que integrará os dados psicofisiológicos coletados no experimento aos eventos produzidos pela IDEs. A produção desta ferramenta descreverá a resposta à segunda questão de pesquisa definida neste estudo. Os detalhes sobre o seu funcionamento, arquitetura e tecnologias adotadas na implementação, estão descritas no Capítulo 4.

A condução de um experimento controlado caracterizará a terceira etapa deste estudo. Neste experimento, apresentado detalhadamente no Capítulo 5, uma amostra de desenvolvedores responderá um questionário contendo questões relacionadas a trechos de código implementados por um desenvolvedor hipotético. Alguns destes trechos de código poderão conter o indicador de Carga Cognitiva. Neste questionário será avaliado o impacto da presença de tal indicador na Percepção de Anomalias de Código e no Nível de Intenção de Refatoração indicado pelos desenvolvedores.

Na quarta e última etapa, os dados provenientes do experimento controlado realizado serão analisados, buscando medir os níveis de significância estatística do efeito da presença da métrica de Carga Cognitiva na Percepção de Anomalias de Código e no Nível de Intenção de Refatoração indicados pelo desenvolvedor. Tal medição visa avaliar o impacto da presença do indicador de Carga Cognitiva nos fatores citados, abrangendo assim a terceira questão de pesquisa.

1.5 Contribuições do Trabalho

Mediante o desenvolvimento deste estudo, uma série de produtos foram gerados, ainda que estes possam ou não ser utilizados fora do contexto em que foram originalmente aplicados, é válido ressaltar as suas principais contribuições.

- **C-1: mapeamento Sistemático da Literatura.** Tal produto poderá ser utilizado como insumo aplicado no embasamento teórico em outros estudos que visam compreender esta temática.

- **C-2: CognIDE:** A ferramenta implementada serve como um processo de ETL entre dados psicofisiológicos e IDEs. Sua aplicabilidade não só se limita ao uso de EEGs, mas pode ser extensível a outros dispositivos sensores utilizados em demais estudos do gênero.

- **C-4: produção de conhecimento empírico.** Tal contribuição servirá de embasamento em estudos futuros que visem compreender a relação entre a presença de indicadores psicofisiológicos nos ambientes integrados de desenvolvimento, e o seu impacto no processo de desenvolvimento de software.

1.6 Organização do Trabalho

O presente estudo está organizado em 6 capítulos, estando estes dispostos da seguinte forma: após uma introdução à temática do trabalho no Capítulo 1, os principais fundamentos teóricos adotados são discutidos no Capítulo 2. Em seguida, a confecção e condução do mapeamento sistemático da literatura é descrito no Capítulo 3. As definições de funcionamento, arquitetura e tecnologias de implementação selecionadas para o desenvolvimento do CognIDE estão descritos no Capítulo 4. Mediante o Capítulo 5, é descrita a metodologia aplicada no desenvolvimento e condução da fase experimental, assim como a discussão acerca dos resultados obtidos. Por fim, as conclusões, considerações finais e trabalhos futuros estão dispostas no Capítulo 6.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados e discutidos os principais conceitos das teorias adotadas para o desenvolvimento deste trabalho. A fundamentação teórica deste estudo está disposta da seguinte maneira: na Seção 2.1 são discutidos os dados psicofisiológicos, os seus tipos, como são coletados e analisados e o que significam em seu contexto. Na Seção 2.2, apresenta-se a teoria da carga cognitiva e como esta relaciona-se ao processo de desenvolvimento de software. Em seguida, na Seção 2.3 são discutidas as anomalias de código e os seus impactos. Por fim, a Seção 2.4 visa explanar os principais conceitos e funcionalidades presentes nas IDEs, e como estas podem facilitar a integração de dados.

2.1 Dados Psicofisiológicos

Os dados psicofisiológicos[1] referem-se ao conjunto de métricas relacionadas aos processos fisiológicos executados pelo corpo humano, os quais surgem como resposta a estímulos oriundos de situações específicas. Tal conjunto de dados pode ser explorada de forma a relacionar as suas oscilações a determinados estados mentais ou processos cognitivos apresentados por um indivíduo por rótulos informacionais que lhe são atribuídos quando este é exposto às determinadas situações (LI; JAIN, 2015; ANDREASSI, 2007). Nas próximas subseções, serão apresentadas as principais métricas utilizadas na identificação de dados psicofisiológicos que podem ser encontradas com determinada frequência literatura especializada.

2.1.1 Métricas Oculares

Entre as principais técnicas para a obtenção de métricas oculares, destaca-se o uso de dispositivos sensores conhecidos por *Eye-trackers*. Estes dispositivos caracterizam-se por aplicarem feixes de *laser* ou luz no espectro infravermelho para mensurar séries de atributos provenientes dos globos oculares, tais como trajetória do movimento dos olhos, grau de dilatação da pupila, detecção de piscar de olhos e duração de ponto focal, ou fixação. Este último atributo caracteriza-se pelo período em que o olho humano permanece focado em um determinado ponto de interesse antes de mover-se novamente. Pesquisadores sugerem que informações visuais somente são processadas durante a fixação, sendo esta positivamente relacionada ao esforço cognitivo (FRITZ et al., 2014; PEITEK et al., 2018; FAKHOURY et al., 2018).

Estudos demonstram que os dados coletados por *Eye-trackers* podem indicar uma série de processos psicofisiológicos, tais como carga cognitiva, excitação, valência, atenção, ansiedade e *stress*. Os conhecimentos sob tais condições psicofisiológicas têm sido empregados no domí-

[1] Na literatura científica, comumente nos trabalhos escritos em inglês, pode-se encontrar o termo *biometric data* (ou *dados biométricos*, em tradução livre), entretanto, neste trabalho, adotou-se a terminologia *dados psicofisiológicos* a fim de reduzir a ambiguidade com o processo de atribuição de identidade a indivíduos por meio de dados biológicos, ou *biometria*. N. do A.

nio de Engenharia de Software, em processos como o desenvolvimento de programas (FRITZ et al., 2014), estratégias de depuração (LIN et al., 2016), usabilidade de Ambientes Integrados de Desenvolvimento (IDEs) (ROSTAMI et al., 2015), dificuldades no aprendizado de novas linguagens de programação, entre outros (HEJMADY; NARAYANAN, 2012).

2.1.2 Métricas Cerebrais

A atividade dos neurônios no cérebro durante a execução de tarefas diversas desencadeia uma série de flutuações na diferença de potencial, ou tensão, por toda a superfície craniana, a qual pode ser mensurada utilizando-se a técnica de eletroencefalografia (EEG) (ANDREASSI, 2007; ANTONENKO et al., 2010; FUNKE et al., 2013; FRITZ et al., 2014; ROSTAMI et al., 2015; LEE et al., 2016, 2018; KOSTI et al., 2018; SEGALOTTO, 2018). Pesquisadores identificaram a possibilidade de traduzir a atividade neural e suas flutuações de tensão em um espectro de ondas divididas em diferentes bandas de frequência, podendo cada banda ser relacionada a um respectivo conjunto de atividades (Tabela 1). A onda alfa (α), por exemplo, pode ser observada nos indivíduos em estado de relaxamento ou meditação, atenuando-se a medida que atividades físicas e/ou mentais vão aumentando (ANDREASSI, 2007; FRITZ; MULLER, 2016). As medidas advindas da atividade eletro dérmica, assim como da temperatura superficial da pele, já foram correlacionadas na literatura com processos psicológicos, como a excitação e algumas emoções específicas (FRITZ; MULLER, 2016)

Tabela 1: Espectro de ondas cerebrais e atividades frequentemente relacionadas.

Onda	Símbolo	Banda	Frequentemente ligada a:
Alfa	α	8 – 12Hz	relaxamento, meditação, reflexão
Beta	β	12 – 30Hz	estado de alerta, foco
Gama	γ	31 – 120Hz	atividade cerebral intensa, aprendizado
Delta	δ	0,4 – 4Hz	sono profundo
Teta	θ	4 – 7Hz	sonolência, meditação

Fonte: Elaborado pelo autor.

Assim como como as flutuações de tensão na superfície craniana, as atividades neurais desencadeiam outros processos biológicos conforme as atividades físicas e mentais exercidas, entre eles estão a hemodinâmica cerebral e a oxigenação. Tais processos podem ser medidos mediante técnicas como a Espectroscopia por Infravermelho Próximo (fNIRS) e Imagem por Ressonância Magnética Funcional (fMRI). Ao contrário da eletroencefalografia, as técnicas citadas são capazes de gerar imagens claras na atividade cerebral em regiões bem definidas, possibilitando assim as suas relações com processos físicos e cognitivos (FISHBURN et al., 2014; LI; LU, 2009; SOLOVEY et al., 2015).

2.1.3 Métricas Cardiorrespiratórias

Durante a formulação do referencial teórico, foram identificados quatro principais métricas relacionadas às características cardiorrespiratórias dos indivíduos analisados nos experimentos: pulso do volume sanguíneo (BVP), a frequência cardíaca (HR), a variabilidade da frequência cardíaca (HRV) e a frequência respiratória (RR).

A frequência cardíaca refere-se ao número de vezes que o músculo cardíaco realiza um ciclo cardíaco (movimento sistólico e diastólico) no período de um minuto, enquanto a variabilidade da frequência cardíaca indica a variação de tempo entre dois ciclos consecutivos. O pulso do volume sanguíneo, também mencionado na literatura sob o termo "pulso", mede o fluxo sanguíneo em partes específicas do corpo de acordo com mudanças comandadas pelo sistema nervoso simpático, especialmente em situações estressoras, tal como aquelas em que se observa elevado nível de carga cognitiva (ANDREASSI, 2007; FRITZ; MULLER, 2016).

2.1.4 Métricas Dérmicas

As métricas dérmicas são o conjunto de medidas relacionadas às variações de propriedades presentes na superfície da pele de um indivíduo, tais como a sua temperatura e a atividade eletro dérmica (EDA), frequentemente encontrada na literatura sob o termo de resposta galvânica da pele (GSR). Os sensores EDA coletam usualmente dados oriundos da variação da resistência da pele em reposta a algum estímulo externo, por exemplo, quando o indivíduo apresenta-se em excitação, as suas glândulas sudoríparas aumentam a produção de suor, que consequentemente reduzirá a resistência da pele, aumentando no que lhe concerne a condutibilidade elétrica da pele, ou atividade galvânica (FRITZ; MULLER, 2016; HAAG et al., 2004).

2.2 Teoria da Carga Cognitiva

O corpo humano, além de constantemente aportar recursos físico para o seu habitual funcional, também o faz com os recursos psicológicos do indivíduo – como memória, atenção, percepção, raciocínio e criatividade – para a execução de tarefas. Ao nível de alocação destes recursos pelo indivíduo, atribui-se o termo de carga cognitiva ou esforço cognitivo (SWELLER, 1988).

Visando a compreensão dos fatores envolvidos na formação da carga cognitiva, Sweller (1988) desenvolveu a chamada Teoria da Carga Cognitiva (TCC). Nesta teoria, o autor e seus colaboradores visavam fornecer orientações de como representar e apresentar informações, de forma a facilitar a execução de atividades e o aprendizado por parte dos indivíduos, fazendo o melhor uso possível dos seus recursos psicológicos. Esta teoria centra-se nas ideias de que (1) existem limitações provenientes da sobrecarga da memória de trabalho no processo de aprendizagem decorrente da instrução e, (2) os materiais instrucionais podem ser avaliados segundo a

forma como estes auxiliam na formação de estruturas mentais para representação de aspectos do mundo, ou esquemas.

A teoria da carga cognitiva ainda prevê um conjunto de orientações para a formulação de materiais instrucionais, visando o melhor aproveitamento da carga cognitiva destinada a formação de esquemas. Para tanto, Sweller (1988) dividiu-a em três categorias, segundo as suas peculiaridades e a forma como estas podem ser manipuladas pelo *designer* instrucional:

- **Intrínseca (*Intrinsic*):** este categoria de carga está intimamente associada ao grau de complexidade inerente ao material instrucional. Para Sweller (1988), toda e qualquer instrução possui uma dificuldade inerente associada. Esta dificuldade não pode ser removida dada a sua natureza intrínseca, entretanto o seu esquema pode ser fragmentado em esquemas de graus de dificuldades menores que, quando processados, podem retornar e reconstruir o esquema original.

- **Estranha (*Extraneous*):** este é uma categoria de carga cognitiva artificialmente induzida, que pode ser atribuída à forma como a informação é apresentada ao indivíduo por meio dos materiais instrucionais. Assumindo-se que os recursos cognitivos do indivíduo são limitados, alocá-los para o processamento da carga estranha, reduz a quantidade de recursos disponíveis para o processamento das cargas intrínseca e pertinente. Sendo assim, a carga estranha pode ser reduzida por meio da otimização do material instrucional pelo *designer*, implicando na melhor distribuição dos recursos para as cargas intrínseca e pertinente.

- **Pertinente (*Germane*):** é aquela destinada à construção e automatização de esquemas mentais, além do seu processamento. Ao passo que a carga cognitiva intrínseca assume-se como imutável, apesar de fragmentável, a carga cognitiva pertinente, assim como a estranha, pode ser otimizada por ações dos *designers* instrucionais. Tais ações visam limitar a carga estranha e, consequentemente, promover o aporte de recursos para a carga cognitiva pertinente, através de formação de esquemas pré-definidos, como, por exemplo, agrupamento de conteúdos, diversificação de conceitos e sequenciação de informações.

Ainda que a teoria da carga cognitiva tenha as suas origens na pedagogia e no processo de aprendizagem, tem-se observado a sua utilização também no contexto da engenharia de software. Segundo Mason et al. (2016), os desenvolvedores precisam lidar com aspectos inerentes à atividade de desenvolvimento, tais como interpretação de condições lógicas, formulação de algoritmos e a própria sintaxe da linguagem de programação utilizada. Além de se observar cada um destes aspectos até nos mais simples programas, a interação entre eles acaba por gerar componentes de interatividade adicionais. Sendo assim, pelo fato de terem que lidar com os componentes básicos presentes nas atividades de desenvolvimento, somado aos seus respectivos componentes de interatividade adicionais, é possível observar que desenvolvedores iniciantes

venham a apresentar altos níveis de carga cognitiva oriunda de fatores extrínsecos à tarefa. Portanto, denota-se que tais características venham a reduzir alocação da carga cognitiva pertinente na execução da atividade, passando a utilizar em maior parte da carga cognitiva estranha (MASON et al., 2016; Du Boulay, 1986).

Observando-se o uso da carga cognitiva de forma empírica no contexto da engenharia de software, Fakhoury et al. (2018) desenvolveu um experimento controlado (Figura 2) visando avaliar uma série de características inseridas no código-fonte e o seu impacto na carga cognitiva dos indivíduos durante atividades de compreensão do código. Como resultados emergentes, a autora constatou que: 1) determinados textos e identificadores no código-fonte podem aumentar significativamente as cargas cognitivas auto-reportada e mensurada; 2) a existência de anti-padrões linguísticos aumentou a carga cognitiva experienciada pelos indivíduos; 3) a presença de inconsistências estruturais não impactaram na carga cognitiva; 4) quando apresentados trechos de código com baixa legibilidade, assim como inconsistências estruturais, 60% dos indivíduos não conseguiram completar as atividades, entretanto, nos 40% restantes, foi constatado o aumento da carga cognitiva. Sendo assim, sugere-se que durante o entendimento do código, aspectos linguísticos, assim como baixa legibilidade podem impactar negativamente na carga cognitiva do indivíduo (FAKHOURY, 2018; FAKHOURY et al., 2018).

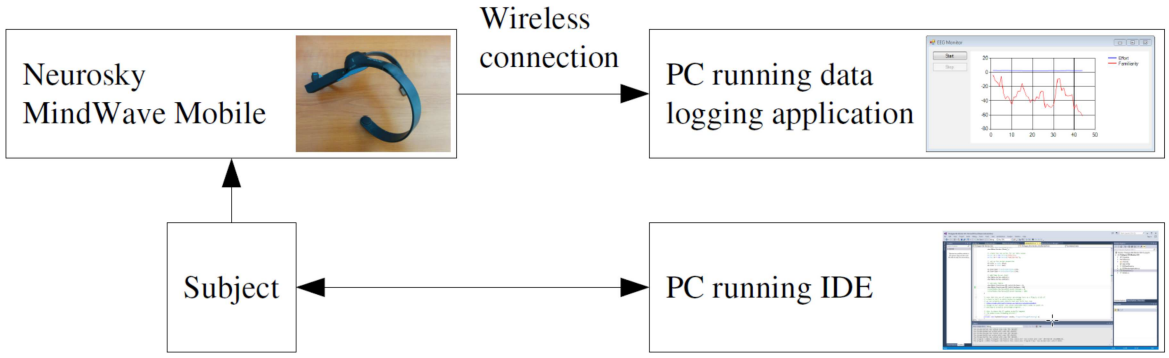
Figura 2: Visão geral do experimento realizado por Fakhoury et al. (2018)

Fonte: FAKHOURY et al. (2018)

Não obstante as evidências empíricas constatando alterações na carga cognitiva do indivíduo mediante aspectos oriundos do código-fonte, conforme abordado anteriormente pela teoria da carga cognitiva, não somente estes aspectos são responsáveis pelas alterações observadas no que tange a carga cognitiva. Características extrínsecas, como a apresentação de conteúdo informacional, podem impactar diretamente na alocação da carga cognitiva estranha (SWELLER, 1988). Neste ponto de vista, os ambientes integrados de desenvolvimento exercem o papel de agente informacional, apresentando o código-fonte para o desenvolvedor, sendo assim, seria possível inferir que, tanto fatores ligados à forma como o código é apresentado, assim como à própria estrutura das IDEs, viriam a produzir alterações na carga cognitiva do desenvolvedor (ROSTAMI et al., 2015; JACQUES; KRISTENSSON, 2015). No experimento executado por Rostami et al. (2015) (Figura 3), ficou evidente que, de um ponto de vista psicofisiológico, diferentes IDEs com *interfaces* de usuário distintas influenciam no aumento da carga cognitiva observada.

Por fim, experimentos realizados sugerem que não somente fatores ligados a composição do código-fonte e ao ambiente onde este é apresentado, como editores de códigos e IDEs, seriam

Figura 3: Visão geral do experimento realizado por Rostami et al. (2015)



Fonte: ROSTAMI et al. (2015)

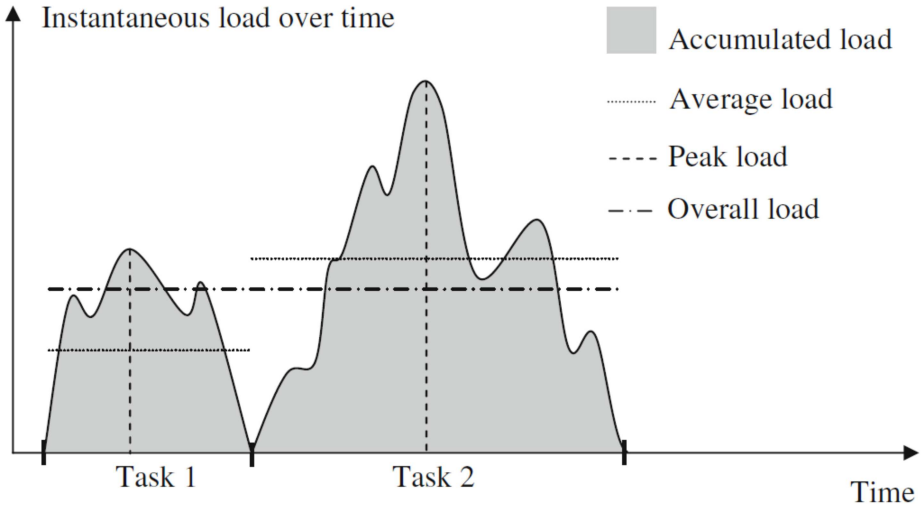
suficientes para explanar oscilações na carga cognitiva. Nestes seus estudos (SIEGMUND et al., 2014, 2017; CRK; KLUTHE, 2014) os autores observam que características do próprio desenvolvedor influenciam na eficácia neural que, por conseguinte, reduz os níveis observados de carga cognitiva. Estas características do desenvolvedor estão ligadas ao seu nível de experiência e escolaridade, contato prévio com códigos similares e ainda, às estratégias de compreensão de código, formuladas por estes (CRK; KLUTHE, 2014; SIEGMUND et al., 2017).

2.2.1 Medindo a Carga Cognitiva

Conforme apresentado anteriormente, observa-se que as variações na carga cognitiva e seus tipos podem manifestar-se de diversas maneiras no contexto da engenharia de software. A fim de obter-se observações mais precisas, deve-se fazer usos de técnicas capazes de mensurar os seus níveis, entretanto esta não é uma tarefa trivial. Segundo Xie e Salvendy (2010) e ilustrado pela Figura 4, a carga cognitiva pode ser dividida em quatro componentes observáveis: *carga instantânea (instantaneous load)*, *carga de pico (peak load)*, *carga média (average load)*, *carga acumulada (accumulated load)* e *carga geral (overall load)*. A carga instantânea reflete as flutuações da carga cognitiva do início ao fim de uma ou mais tarefas. A carga de pico é o valor máximo observado no período em que foi realizada uma atividade ou conjunto de atividades. A carga acumulada apresenta a quantidade total de esforco cognitivo em um atividade, já a carga média representa a intensidade média de carga cognitiva neste mesmo período. Por fim a carga geral, apesar de assemelhar-se com a carga acumulada e a média, indica o total de carga cognitiva **percebida** pelo indivíduo quando executadas as tarefas, sendo este tipo de carga a frequentemente avaliada pela maioria dos instrumentos (ANTONENKO et al., 2010).

Na literatura, constata-se cinco abordagens principais para a medição da carga cognitiva, as quais serão agrupadas em: 1) *medidas não-contínuas*, composta pelas *indiretas*, *subjetivas*, *de tarefas secundárias* e *de eficiência*; e 2) *medidas contínuas*, contendo todas as abordagens *fisiológicas*. Ambas as categorias e suas abordagens serão apresentadas nas seções subsequentes. (ANTONENKO et al., 2010; SWELLER; AYRES; KALYUGA, 2011; MORRISON; DORN;

Figura 4: Representação gráfica dos níveis de carga cognitiva no decorrer do tempo.



Fonte: ANTONENKO et al. (2010) adaptado de XIE; SALVENDY (2010)

GUZDIAL, 2014).

2.2.1.1 Medidas Não-contínuas

As *medidas indiretas* datam do início dos estudos acerca da carga cognitiva. Na época, Sweller (1988) e seus colegas teorizaram que estratégias de aprendizado mediante a busca de problema-solução, aumentavam significativamente a carga cognitiva dos alunos durante a fase de aprendizado. Tais teorizações foram posteriormente corroboradas por modelos computacionais de produção que comparavam as estratégias de busca. Sweller (1988) identificou que buscas mais complexas requeriam modelos mais complexos para simular o processo de problema-solução, correspondendo assim a mais informações sendo guardadas e processadas na memória de trabalho. Esta foi a primeira medida indireta de carga cognitiva observadas.

Estudos posteriores sugeriram que estratégias de aprendizado complexas viriam a aumentar a carga cognitiva do indivíduo, reduzindo assim seu desempenho, desta forma, constatou-que a complexidade do material instrucional e as estratégias de aprendizados serviriam de medidas indiretas para altos níveis de carga cognitiva. Ademais, em trabalhos subsequentes, foi constatado que, operações compostas por muitas etapas, somadas a presença de diversas variáveis, ocasionavam expressivo aumento nas taxas de erros apresentadas pelos participantes. Tal observação evidenciou que as taxas de erros seriam fortes indicadores de um aumento no uso da memória de trabalho, que por conseguinte, seria responsável pelas variações da carga cognitiva (SWELLER; AYRES; KALYUGA, 2011).

Com o aumento da necessidade de se mensurar diretamente a carga cognitiva e a ausência de mecanismos que o possibilitassem, surgiram então as *medidas subjetivas*. Paas (1992) concluiu que os sujeitos avaliados após processos de aprendizagem, seriam capazes de definir a

intensidade de esforço experienciada, e este valor poderia ser considerado como um indicador de carga cognitiva. Utilizando-se uma escala Likert de 9 pontos, os quais iam de "muito, muito baixo (1)"a "muito, muito alto (9)", grupos de indivíduos foram avaliados em vários aspectos durante o processo de aprendizagem e avaliação. Paas (1992) encontrou relações entre os níveis de carga cognitiva, apontados pelos participantes e o resultados dos testes de desempenho, marcando assim o surgimento da primeira medida subjetiva de carga cognitiva. O sucesso desta motivou outros pesquisadores a utilizá-lá em seus experimentos. Com o decorrer dos anos, variantes como a *medida de dificuldade* surgiram como alternativa à medida de esforço mental (SWELLER; AYRES; KALYUGA, 2011).

As medidas subjetivas são os instrumentos mais utilizados para mensurar-se a carga cognitiva. Apesar disso, o método tradicionalmente aplicado para a medição da carga cognitiva é de se utilizar uma tarefa secundária combinada a uma primária, também chamada de metodologia de dupla-tarefa (BRITTON; TESSER, 1982). Uma medição de tarefa secundária consiste em se aplicar uma atividade extra àquela que se deseja verificar. Sweller, Ayres e Kalyuga (2011) elucidam que quanto maior a carga cognitiva exigida na primeira tarefa, maior será a deterioração de desempenho na tarefa secundária. Geralmente, a segunda tarefa não possui relação com a primeira e requer menos memória de trabalho

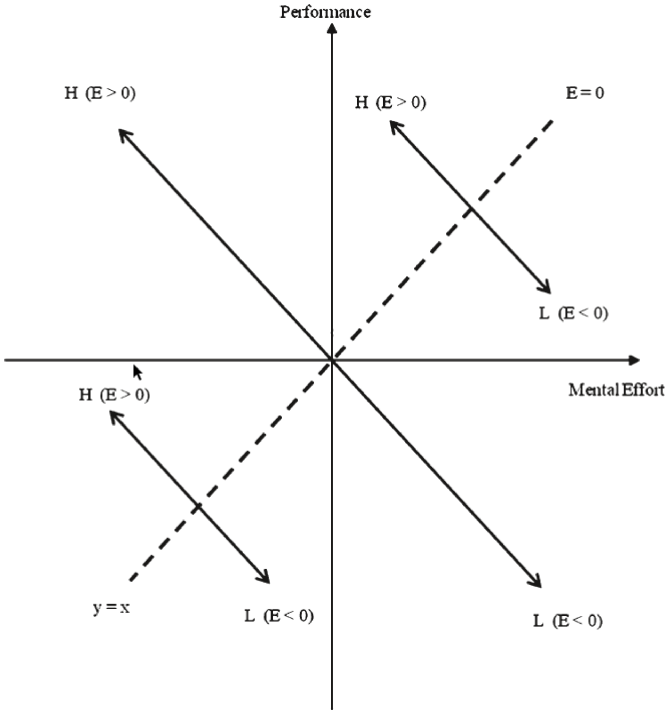
Inconsistências nas medidas subjetivas identificadas posteriormente motivaram o surgimento de novos métodos a fim mensurar a carga cognitiva. Para tanto, Paas e Van Merrienboer (1993) desenvolveram a primeira *medida de eficiência*, combinando o já conhecido esforço mental com os indicadores de desempenho de tarefas. Nesta abordagem, os autores argumentaram que, independente se duas estratégias instrucionais produzam o mesmo grau de desempenho, a estratégia que utilizar menos recursos cognitivos é considerada mais eficiente. A Eficiência (E) é calculada utilizando-se a seguinte fórmula proposta por Paas e Van Merrienboer (1993):

$$E = \frac{Z_{Pteste} - Z_{Eteste}}{\sqrt{2}}$$

onde Z_{Ptest} e Z_{Ptest} representam, respectivamente, o escore padronizado (*Z-score*) dos testes de desempenho e do esforço mental após a avaliação. Segundo os autores, a fórmula foi baseada no cálculo da distância perpendicular entre dois pontos, sendo assim, quando o escore padronizado para o desempenho e o esforço mental são iguais, o valor de eficiência (E) é zero, ilustrado pela linha diagonal na Figura 5. Todos os pontos nesta linha representam nenhuma eficiência ($E = 0$), enquanto os pontos acima ($E > 0$) indicam aprendizado eficiente e os abaixo ($E < 0$), ineficiente. Paas e Van Merrienboer (1993) explanam ainda que alta eficiência instrucional é resultado de alto desempenho nas tarefas e baixo esforço mental (zona H da Figura 5), enquanto a baixa eficiência instrucional é um produto de baixo desempenho nas atividades com alto esforço mental (zona L da Figura 5).

Apesar da forte adoção no uso da métrica de eficiência, Hoffman e Schraw (2010) identificaram algumas preocupações associadas com o cálculo da eficiência instrucional. Em uma

Figura 5: Representação gráfica da medida de eficiência.



Fonte: SWELLER; AYRES; KALYUGA (2011)

revisão da medida, Hoffman e Schraw (2010) categorizaram o modelo original de Paas e Van Merrienboer (1993) como sendo um modelo de desvio, já que ele é baseado nas diferenças entre os escores padronizados de desempenho e esforço mental. Os autores também alegaram ser que existe dificuldade em interpretar o significado de se subtrair duas variáveis com semânticas distintas, como é o caso do desempenho e do esforço mental. Apesar das críticas, Hoffman e Schraw (2010) defendem que modelo de desvio de Paas e Van Merrienboer (1993) ainda pode utilizado para diferentes objetivos de pesquisa, se o que se deseja é avaliar justamente a diferença entre as variáveis (SWELLER; AYRES; KALYUGA, 2011).

2.2.1.2 Medidas Contínuas

As medidas contínuas de carga cognitiva possibilitam adquirir dados em instantes específicos do tempo, permitindo uma interpretação mais precisa acerca dos efeitos de intervenções instrucionais, quando comparados às medidas únicas de carga acumulada ou geral. Neste aspecto, Antonenko et al. (2010) salienta que as medidas fisiológicas são a única natureza de medidas utilizadas para uma análise *online* das variações da carga cognitiva em todos os seus níveis (instantânea, pico, acumulada, média e geral).

Na engenharia de software empírica, as principais medidas fisiológicas utilizadas experimentos visando medir a carga cognitiva são: a variabilidade da frequência cardíaca (HRV) (COUCEIRO et al., 2019a,b), métricas oculares (FRITZ et al., 2014; PEITEK et al., 2018;

COUCEIRO et al., 2019a; FAKHOURY et al., 2020), métricas eletrodérmicas ((FRITZ et al., 2014; FRITZ; MULLER, 2016; FUCCI et al., 2019; FAKHOURY et al., 2020)). Além de tais abordagens para se mensurar a carga cognitiva, nos últimos anos tem se observado um aumento (GONCALES et al., 2019) na utilização de dispositivo de eletroencefalografia (ANTONENKO et al., 2010; FUNKE et al., 2013; FRITZ et al., 2014; ROSTAMI et al., 2015; LEE et al., 2016, 2018; KOSTI et al., 2018; SEGALOTTO, 2018), imagem por ressonância magnética funcional fMRI (SIEGMUND et al., 2014, 2017; FLOYD; SANTANDER; WEIMER, 2017; PEITEK et al., 2018) e espectroscopia por infravermelho próximo (FISHBURN et al., 2014; FAKHOURY, 2018; FAKHOURY et al., 2020), além da utilização combinada destas abordagens, conforme discutido na Seção 2.1.

Apesar da difundida adoção de diversas estratégias para aquisição de métricas fisiológicas objetivando medir a carga cognitiva, Antonenko et al. (2010) argumenta que, ao contrário de outras abordagens que requerem que o indivíduo permaneça em posições estáticas por longos períodos (fMRI), o EEG mostra-se como uma abordagem não-invasiva de se medir atividade neural ligada à carga cognitiva em cenários reais. Em contrapartida, Antonenko et al. (2010) aponta que o este possui baixa resolução espacial quando comparado a outros métodos de coleta, como a fMRI. Ademais, o EEG é suscetível à produção artefatos[2], como o piscar de olhos, movimentos do indivíduo ou interferências eletromagnéticas.

Não obstante as limitações apontadas, o uso do EEG para a medida contínua de carga cognitiva mostra-se como uma solução viável e acessível, dada a sua grande resolução temporal, capaz de medir com precisão de milissegundos as variações na atividade neural, refletindo continuamente o seu estado cognitivo (ANTONENKO et al., 2010).

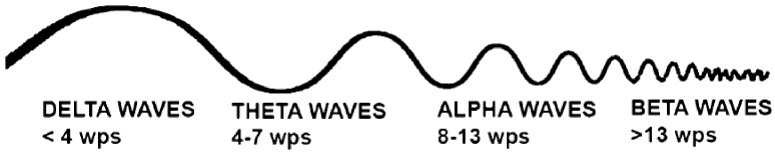
2.2.1.3 Medindo a Carga Cognitiva com Eletroencefalografia

Consoante ao apresentado na Seção 2.1.2, a atividade elétrica neural pode ser mensurada pelos EEGs em um espectro contínuo composto por cinco bandas (ou ritmos) principais (Figura 6). Segundo sintetizado por Antonenko et al. (2010), duas dessas bandas são sensíveis à manipulação da dificuldade de tarefas: *Alfa* (α) e *Teta* (θ). Medidas nas mudanças dos espectros alfa e teta refletem o que está acontecendo no processamento de informação de um indivíduo, mesmo que este não esteja ciente disto (KLIMESCH; SCHACK; SAUSENG, 2005). Resultados de experimentos sugerem que os espectros alfa e teta possuem relações opostas à dificuldade de uma tarefa: conforme esta aumenta, observa-se redução na atividade alfa (dessincronização), em contrapartida, é possível constatar-se um aumento na atividade da banda teta (sincronização) (ANTONENKO et al., 2010).

Mediante fatores como idade, volume cerebral e diferenças individuais, autores observaram que estes podem exercem influência sobre o comportamento dos espectros ondulatórios (NU-

[2]No processamento de sinais de EEG, um artefato pode ser definido como todo potencial elétrico proveniente de outra fonte que não seja o cérebro. Não confundir com os construtos do processo de desenvolvimento de software frequentemente mencionandos pelo autor.

Figura 6: Ritmos das ondas cerebrais humanas, em ondas por segundo (*waves per second*).



Fonte: ANTONENKO et al. (2010)

NEZ; CUTILLO, 1995; KLIMESCH, 1999; NIEDERMEYER; OTHERS, 1999). De modo a mitigar tal problema, foi proposto que as análises dos sinais capturados pelo EEG passassem a ser induzidas por determinado evento ou tarefa, no lugar ter o seu valor absoluto como objeto de análise. Para tanto, adotou-se o uso do ERD/ERS, ou (des-) sincronização relacionada ao evento[3], originalmente desenvolvido para detectar mudanças na banda alfa. Segundo Pfurtscheller e Lopes Da Silva (1999), essa medida indica a redução (dessincronização relacionada ao evento ou ERD) ou o aumento (sincronização relacionada ao evento ou ERS) da intensidade da banda durante um intervalo de atividade (ativação) quando comparado a um intervalo de base (referência). O intervalo de base refere-se a um período de controle pré-atividade, onde o indivíduo fica sem executar tarefas, enquanto o intervalo de atividade refere-se ao período onde as tarefas são executadas(ANTONENKO et al., 2010). O cálculo utilizado para obter-se o valor do ERD/ERS é descrito pela seguinte fórmula:

$$ERD|ERD = \frac{Intensidade_{base} - Intensidade_{atividade}}{Intensidade_{base}} \times 100$$

Em conformidade às recomendações de Antonenko et al. (2010), o cálculo demonstrado deve ser aplicado em cada amostra coletada de cada canal do EEG, e então deve-se calcular a média das amostras relativas aos seus respectivos canais, objetivando aumentar a confiabilidade do resultado. Valores positivos de ERD|ERS indicam redução na intensidade da banda (ERD, dessincronização), enquanto valores negativos denotam aumento na intensidade da mesma (ERS, sincronização).

Os indicadores ERD/ERS para as bandas alfa e teta mostraram-se responsivos em diversas tarefas com diferentes níveis de dificuldades. No estudo de Stipacek et al. (2003), evidenciou-se uma relação linear entre o aumento dos valores de ERD na banda alfa superior e da carga cognitiva experienciada.

[3]do inglês, *event-related (de-) synchronization*

2.3 Anomalias de Código

Durante o processo de desenvolvimento de software, é comum que seus componentes possam ser criados, alterados e excluídos pelos mais diversos desenvolvedores que venham a contribuir com a solução de software em questão. Em determinado ponto do desenvolvimento da solução, observa-se o surgimento de anomalias de código. Estas anomalias, por vezes referidas pelo termo *Code Smells* caracterizam-se como um trecho, ou ainda, um componente de software composto por código de baixa qualidade (VIDAL et al., 2018). Em contrapartida, autores (OIZUMI et al., 2018; YAMASHITA; COUNSELL, 2013) defendem que uma anomalia não necessariamente atua salientando a baixa qualidade de código, mas sim, servindo como um indicador de possíveis problemas de *design* nos componentes da solução observada.

As anomalias de código podem surgir em diversos níveis e formatos nos projetos de software. Apesar de existirem ferramentas automatizadas para a sua detecção, realizá-la manualmente é uma habilidade inerente aos desenvolvedores. No decorrer os anos, novas anomalias de código têm sido identificadas e outras deixam de existir, assim como determinados padrões de projeto passam a ser considerados anomalias e vice-versa. Quando estas tornam-se recorrentes na indústria de software, é comum que sejam catalogadas para a sua fácil identificação. Em seu livro *Refactoring Improving the Design of Existing Code*, Fowler (2018) sintetiza as principais anomalias de código identificadas e conhecidas na indústria de software, as quais serão brevemente apresentadas a seguir.

- ***Mysterious Name***: caracteriza-se pela nomenclatura incoerente, dúbia, sem sentido ou ainda, pela ausência desta, gerando dúvida e dificuldades de compreensão para que está trabalhando no contexto apresentado. Aplica-se, mas não limita-se a variáveis, métodos, classes ou até mesmo pacotes e *namespaces* inteiros.

- ***Duplicated Code:*** apresenta-se sob a forma de trechos de código, ou ainda, componentes completos, com lógica idêntica ou semanticamente similar, espalhando-se pelo projeto e demandando ações múltiplas quando este possui alterações em seu comportamento.

- ***Long Function*** ou ***God Method:*** esta anomalia é observada quando uma função estende-se além do necessário, concentrando a lógica de outras funções dentro de si, descaracterizando-se segundo a finalidade para a qual foi inicialmente criada.

- ***Long Parameter List:*** identificada quando um grande número de parâmetros é passado para uma função (ou outra entidade), prejudicando a sua legibilidade e aumentando a dificuldade de compreensão acerca da necessidade de tais parâmetros.

- ***Global Data:*** mesmo que variáveis globais sejam comumente utilizadas, estas são consideradas anomalias quando possuem mutabilidade fora de controle, gerando efeitos colaterais em componentes que as utilizam mesmo estando fora do escopo da alteração.

- ***Mutable Data:*** assim como a *Global Data*, esta anomalia ocorre quando não há garantias de que o dado necessário para a execução de determinado trecho de código seja imutável, ou seja, não tenha sofrido alterações em sua estrutura, levando a comportamentos imprevistos pelo componente que o utiliza.

- ***Divergent Change:*** é observada quando um componente demanda alterações internas independentemente se alteração origina-se de diferentes contextos que não possuem relação direta com ele.

- ***Shotgun Surgery:*** de forma oposta a *Divergent Change*, a *Shotgun Surgery* é identificada quando há a necessidade de alterações em diversos componentes distintos, mesmo com uma pequena alteração no contexto.

- ***Feature Envy:*** quando um programa é modularizado, deseja-se que as funções de um determinado módulo interajam o máximo possível entre si, maximizando o seu uso. A *Feature Envy* é observada quando uma função interage mais com módulos externos do que com o seu módulo de origem, aumentando assim o acoplamento modular.

- ***Data Clump:*** dados foram criados para interagirem entre si, entretanto, quando os mesmos conjuntos de dados são observados com frequência em locais distintos, como parâmetros em funções ou e construtores, caracteriza-se como *Data Clump*.

- ***Primitive Obsession:*** a presença de tipos primitivos como inteiros, numerais de ponto flutuante e cadeia de caracteres é inerente a quaisquer linguagens de programação modernas. Entretanto, quando seu uso é excessivo, evitando a criação de tipos derivados para a representação de dados específicos do domínio, denota-se uma anomalia do tipo *Primitive Obsession*.

- ***Lazy Elements:*** tal categoria de anomalias é atribuída a funções projetadas para uso frequente mas que são utilizadas uma única vez, ou ainda, classes compostas por apenas uma função. Tal anomalia é observada durante a fase inicial de um projeto, entretanto é encontrada com mais frequência após ciclos de refatoração.

- ***Speculative Generality:*** observada frequentemente em linguagens orientadas a objetos, ocorre quando utiliza-se excessivamente o uso de abstrações, criando-se interfaces e classes abstratas utilizadas uma única vez ou até mesmo vazias, apenas visando um desacoplamento artificial de suas implementações.

- ***Temporary Field:*** essencialmente uma classe é composta pelo seu conjunto de método e atributos. Porém, quando alguns destes necessitam de situações muito específicas para serem preenchidos, é possível observar a presença da anomalia *Temporary Field*.

- ***Message Chains:*** o encadeamento de mensagens (*Message Chains*) ocorre quando um determinado objeto solicita a outro por um terceiro, sendo que este, por sua vez, realiza a mesma sequência de operações até encontrar a função responsável por retornar o objeto originalmente requisitado. Este comportamento acaba por gerar um acoplamento profundo equivalente a toda a extensão de chamadas.

- ***Middle Man:*** este comportamento é encontrado nas situações em que, para responder às requisições solicitadas, um objeto demanda excessivamente delegá-las a um terceiro. Este concentra boa parte da lógica utilizada pelo objeto que o delega, não havendo a possibilidade de ser chamado diretamente sem que o objeto delegante seja acionado.

- ***Large Class*** ou ***God Class:*** assim como a *Long Method* ou *God Method*, mas apresentando-se sob a forma de classes, esta anomalia denota quando classes adquirem quase, senão todo, o comportamento de um determinado módulo do sistema, produzindo um forte acoplamento sobre apenas esta classe.

- ***Data Class:*** são classes que contém apenas atributos e os métodos utilizados para alterá-los, nada mais. Também são chamadas de *classes anêmicas*, por carecerem de comportamentos complexos, sendo sua única funcionalidade limitada a um contêiner para o transporte de dados entre as camadas de um sistema.

- ***Refused Request:*** pode ser observada quando a abstração da herança é utilizada de forma errada. Neste tipo de anomalia, é possível encontrar classes derivadas que não sobrescrevem todos os métodos de sua classe base, tampouco fazem uso dos atributos disponibilizados por esta.

- ***Comments:*** comentários não são uma anomalia, muito pelo contrário: são considerados uma boa prática no desenvolvimento de software. Entretanto, quando é observado um comentário extenso com explicações excessivas acerca da lógica de uma função ou do motivo de uma classe existir, é um sinal de uma possível anomalia na própria função/classe que este visa explanar.

Mediante a relação de anomalias apresentadas, é possível notar que estas podem ser encontradas em soluções corporativas e ainda sim, não oferecer riscos à estabilidade arquitetural da aplicação. É importante ressaltar que uma anomalia de código deve ser considerada um risco, quando esta representa retrabalho e a sua presença acarreta produção de defeitos e comportamentos inesperados. Mais importante do que saber identificar anomalias é saber como corrigi-las (FOWLER, 2018), portanto, a próxima seção focará na conceitualização e usabilidade das técnicas de refatoração.

2.3.1 Refatoração de Código

Como explanado anteriormente, as anomalias de código podem servir como indicadores de possíveis problemas de *design* na solução de software produzida. De modo a tratar tais problemas, é utilizada técnica conhecida como refatoração de código.

O termo refatoração é um anglicismo para o substantivo inglês *refactoring*, que denotaria o ato de reestruturar software sem alterar o seu comportamento observável (FOWLER, 2018). É possível encontrar textos em que o termo *refactoring* é substituído pela palavra *reestruturação*. Como defendido por Fowler (2018):

> Refatoração é muito similar às otimizações de desempenho, ambas envolvem a manipulação de código sem haver alterações no funcionamento do programa na totalidade. A diferença está no propósito: refatorar sempre realizado para fazer o código “fácil de se entender e barato de se modificar”. Isso pode tornar as coisas mais rápidas ou devagar. Já na otimização de desempenho, eu me preocupo somente em tornar o programa rápido, e estarei preparado para lidar com código difícil de trabalhar se eu realmente necessito melhoria de desempenho.

Em primeira instância a refatoração de código pode ser considerada como a "solução de todos os problemas", entretanto como salientado por Fowler (2018), a refatoração não visa ser uma solução universal para todas as anomalias presentes em um programa, porém o autor elucida os benefício de adotá-la:

- Melhora o *design* do software;
- Torna o software mais fácil de se entender;
- Auxilia o desenvolvedor na identificação de defeitos (*bugs*);
- Possibilita ao desenvolvedor uma programação mais rápida;

Uma vez identificadas as necessidades da solução de software, e se essa deve ser refatorada, é importante definir a sequência de passos lógicos que devem seguir-se, objetivando a garantia de uma refatoração eficaz sem a geração de efeitos colaterais que venham a alterar o comportamento da aplicação. Sendo assim, os autores Mens e Tourwé (2004) enumeram as seguintes etapas para a condução da refatoração de código:

1. **Identificar** onde o software poderia ser refatorado.
2. **Determinar** quais estratégias de refatoração poderiam ser aplicadas nos pontos identificados mediante o item anterior.
3. **Garantir** que a refatoração a ser aplicada preserve o comportamento programa.

4. **Aplicar** a(s) estratégia(s) de refatoração escolhida(s).

5. **Avaliar** se a refatoração afeta as características de qualidade do software (complexidade, legibilidade, compreensão e manutenibilidade) ou do processo (custo, produtividade, esforço) de forma positiva.

6. **Manter** a consistência entre o código do programa refatorado e outros artefatos de software, como documentações, documentos de arquitetura, requisitos, especificações, etc.

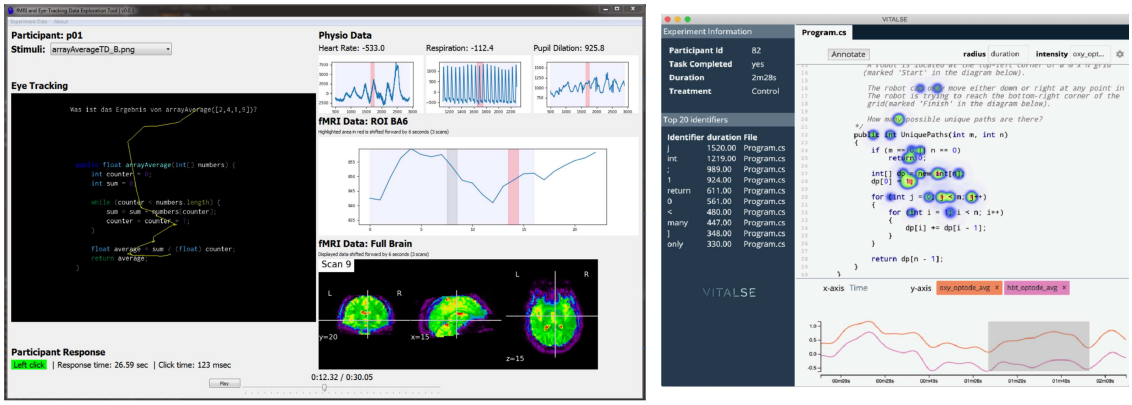
A primeira etapa do processo de refatoração consiste na procura por anomalias comuns (vide Seção 2.3) presentes no código-fonte. Em seguida, na etapa 2, o desenvolvedor deve analisar o código em questões, buscando pela presença de uma anomalia em particular de modo a determinar qual refatoração ou conjunto destas aplica-se-á de modo a remover a anomalia e melhorar a estrutura do código. Uma vez determinadas as estratégias de refatoração, é necessário garantir que o comportamento da aplicação na totalidade não seja alterado (etapa 3). Esta ação pode ser delegada às ferramentas automáticas de refatoração presentes na maioria dos ambientes integrados de desenvolvimento modernos. Ainda sim, é fortemente recomendada a utilização de testes unitários e de regressão visando garantir a consistência comportamental do programa (VIDAL et al., 2018). Caso sejam utilizadas ferramentas para a refatoração automatizada, a etapa 4 é contemplada por estas. Caso a refatoração seja realizada manualmente, é possível fazer uso das estratégias de refatoração presentes em catálogos já existentes e testados pela indústria (FOWLER, 2018). Para a coleta das métricas e verificações necessárias à contemplação da etapa 5, é possível utilizar-se de ferramentas especializadas, tais como SonarQube[4], Fortify[5], Coverity[6], entre outras. Por fim, a etapa 6 pode variar segundo o processo de software adotado pelo contexto, sendo assim, o desenvolvedor deve avaliar quais demais artefatos de software devem ser atualizados (testes, documentações, diagramas, etc.) visando manter a consistência destes as alterações performadas no código.

2.4 Ambientes Integrados de Desenvolvimento

Os Ambientes Integrados de Desenvolvimento (IDEs) podem ser considerados como o principal conjunto de ferramentas utilizado pelos desenvolvedores, tanto para tarefas de criação de novas funcionalidades de software, como para manutenção de código-fonte. Este conjunto de ferramentas engloba editores de código-fonte, navegação por arquivos, compiladores e depuradores, assistentes para criação e refatoração de artefatos, ferramentas de realce de sintaxe (*Syntax Highlighting*), entre outras funcionalidades (SNIPES et al., 2015). Para o desenvolvedor é essencial, além do domínio sobre a linguagem de programação em questão e os aspectos

[4]https://www.sonarqube.org/
[5]https://www.microfocus.com/en-us/products/static-code-analysis-sast/overview
[6]https://www.synopsys.com/software-integrity.html

Figura 7: CodersMUSE (esquerda) e VITALSE (direita): duas ferramentas para exploração de dados multimodais aplicados a experimentos na engenharia de software.



Fonte: Extraído e adaptado de Peitek et al. (2019) e Arnaoudova, Fakhoury e Roy (2020), respectivamente.

lógicos envolvidos na resolução do problema, possuir conhecimento sobre as ferramentas que serão utilizadas para a confecção da solução (ASTROMSKIS et al., 2017).

Nos últimos anos, autores têm realizado pesquisas a respeito do uso de IDEs por parte dos desenvolvedores, tanto no processo de desenvolvimento (ASTROMSKIS et al., 2017; ZAYOUR; HAJJDIAB, 2013), como na sua eficácia enquanto ferramenta de depuração (ZAYOUR; HAMDAR, 2016). Usualmente empregando *Eye-trackers* para coleta de dados quantitativos, os autores avaliam aspectos como a facilidade de uso, a organização da interface gráfica, o conjunto de ferramentas disponíveis, entre outros recursos variantes segundo o fornecedor.

Com o objetivo de evoluir as ferramentas e facilitar o desenvolvimento de estudos experimentais, trabalhos têm sido produzidos na tentativa de facilitar o processo de integração entre dados biométricos experimentais e ambientes integrados de desenvolvimento. Este é o caso de uso dos Eye-trackers aplicados nas IDEs, e ferramentas como o ITraceVis (CLARK; SHARIF, 2017), que visam relacionar os pontos de interesse identificados com o código-fonte produzido, assim como a interação do desenvolvedor com a ferramenta (Figura 7). Seguindo esta mesma linha de pesquisa, destacam-se o CodersMUSE (PEITEK et al., 2019) e a VITALSE (ARNAOUDOVA; FAKHOURY; ROY, 2020), duas ferramentas para a exploração de dados multimodais (Figura 7) aplicadas a experimentos na engenharia de software.

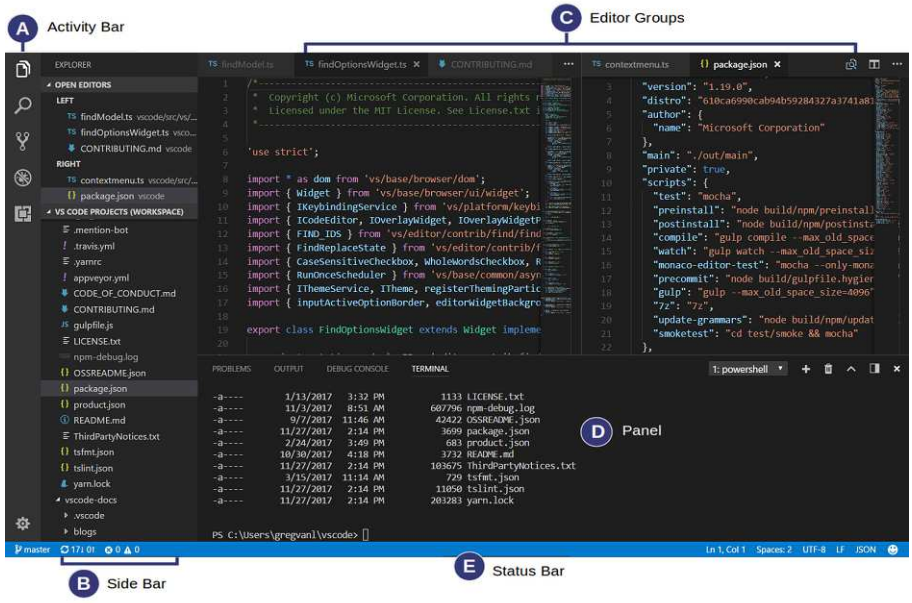
2.4.1 Visual Studio Code

O Visual Studio Code, popularmente referido como VSCode, é um editor de código-fonte de software livre e código-aberto[7] (Figura 8), criado e mantido pela Microsoft para a plataformas Windows, Linux e macOS . Além das ferramentas de edição, ele inclui suporte à depuração,

[7] Apesar de ser considerado como um software opensource, o download oficial do Visual Studio Code está sob uma licença proprietária

controle de versão incorporado, realce de sintaxe, complementação de código-fonte inteligente e sensível ao contexto, refatoração de código e inserção de trechos predefinidos (*snippets*). Ademais, devido a sua natureza extensível, diversas funcionalidades podem sem adicionadas ou modificadas, através de extensões que podem ser baixadas dentro a própria ferramenta.

Figura 8: Tela principal do Visual Studio Code



Fonte: https://code.visualstudio.com/docs/getstarted/userinterface

O Visual Studio Code foi construído sobre o Electron[8], um *framework* utilizado no desenvolvimento de aplicações para área de trabalho, que utiliza como base a *engine* de processamento JavaScript V8 do servidor de aplicações, Node.js[9], originalmente desenvolvida para o navegador web Google Chrome.

2.4.1.1 Language Server Protocol

O Protocolo de Servidor de Linguagens, ou LSP, é um protocolo aberto, baseado em JSON-RPC para a aplicação entre IDEs e servidores, com a finalidade de prover um conjunto de funcionalidades específicas consonante à linguagem de programação utilizada. O principal objetivo do protocolo, é permitir que o suporte às linguagens de programação seja implementado e distribuído independentemente do editor ou IDE utilizada.

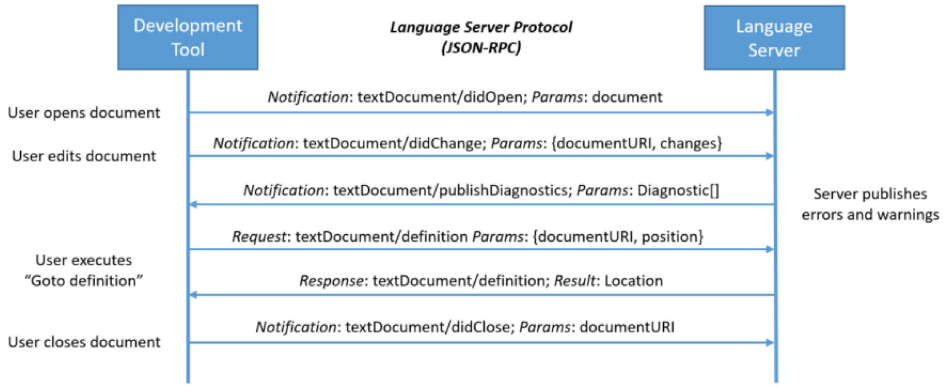
Originalmente desenvolvido pela Microsoft para o seu editor de código-fonte, Visual Studio Code, O LSP agora está disponível como um padrão aberto, passível de ser implementado em outros editores ou IDEs. Em 2016, a Microsoft anunciou uma ação conjunta com a Red Hat e Codenvy, a fim de padronizar as especificações do protocolo. Suas especificações e exemplos

[8]https://www.electronjs.org/
[9]https://nodejs.org/en/

de implementações estão disponíveis no repositório digital, GitHub[10]

Quando ativo, o LSP é executado como um processo independente e as ferramenta para edição de código comunicam-se com ele por mediante a utilização do JSON-RPC. A Figura 9 ilustra como um editor ou IDE comunica-se ao LSP durante uma sessão de edição de código-fonte:

Figura 9: Comunicação entre processos durante uma rotina de edição.



Fonte: https://microsoft.github.io/language-server-protocol/overviews/lsp/overview/

- **O usuário abre um arquivo:** a ferramenta conecta-se ao servidor, notificando-o que um arquivo (*document*) está aberto (mediante o evento `textDocument/didOpen`). Deste momento em diante, o arquivo carregado encontra-se na memória da ferramenta, e não mais no sistema de arquivos do disco. O conteúdo do arquivo agora permanece sincronizado entre a ferramenta e o servidor.

- **O usuário edita o arquivo:** A ferramenta notifica o servidor sobre um evento de modificação do arquivo (`textDocument/didChange`) e o servidor, portanto, atualiza a sua representação do arquivo com os dados informados pelo editor. Quando isto ocorre, o servidor analisa a informação e notifica a ferramenta com uma resposta, informando erros e avisos identificados (`textDocument/publishDiagnostics`).

- **O usuário executa o comando "Go To Definition" em um símbolo de um documento aberto:** a ferramenta envia o evento `textDocument/definition` em formato de requisição contendo dois parâmetros: (1) a URI do documento e (2) a posição do texto onde a requisição de "Go To Definition" foi iniciada para o servidor. Este responde à ferramenta informando a URI do documento e a posição da definição do símbolo no documento.

- **O usuário fecha um arquivo (*document*):** neste momento, uma notificação contendo um evento do tipo `textDocument/didClose` é enviada pela ferramenta informando

[10]https://microsoft.github.io/language-server-protocol/specifications/specification-current/

o servidor de linguagem que o documento não está mais presente na memória. O conteúdo do arquivo agora é atualizado no sistema de arquivos.

O fluxo de comunicação entre o editor e o servidor de linguagem, apresentado na Figura 9, ilustra como o Language Server Protocol opera ao nível de referências (URIs) e posições nos documentos utilizados. Os tipos de dados apresentados são agnósticos de linguagem de programação e podem ser aplicados a quaisquer outras linguagens.

3 TRABALHOS RELACIONADOS

Este capítulo foca em obter (1) um panorama acerca do estado-da-arte no uso de (2) dados psicofisiológicos e suas aplicações em (3) ambientes integrados de desenvolvimento. Como apontado pela problemática da Seção 1.1, apesar dos esforços na produção de estudos que buscam compreender o comportamento de processos psicofisiológicos durante o desenvolvimento e a compreensão de código, pouco ainda se sabe sobre tais fenômenos e como o conhecimento extraído pode ser aplicado como fator evolutivo no processo de desenvolvimento de software.

Muitos dos estudos limitam-se ao uso de apenas uma, eventualmente duas, abordagens quanto a utilização técnicas para a coleta de dados psicofisiológicos, tais como fMRI, fNIRS, EEG ou Eye-tracking. Ademais, tais estudos apresentam propósitos distintos quanto a aplicabilidade dos dados coletados, assim como pelos atributos representados, sem um consenso de como estes podem de fato serem aplicados na evolução da engenharia de software.

Ademais, para a obtenção de uma visão atual da literatura, torna-se importante (1) identificar quais os categorias de dados psicofisiológicos estão sendo objetos de estudos, (2) qual a sua aplicabilidade nos contextos em que são utilizados, (3) como estes dados estão sendo coletados, (4) quais ferramentas estão sendo utilizadas para a sua análise e integração nas IDEs utilizadas, e (5) como o estado-da-arte pode auxiliar na identificação de possíveis lacunas que venham a servir de oportunidades para o desenvolvimento de estudos futuros. Visando contemplar tais pontos, este estudo propõe o desenvolvimento de um mapeamento sistemático da literatura (SMS), cujas etapas metodológicas foram desenvolvidas e executadas seguindo um conjunto de protocolos bem aceitos na literatura (PETERSEN et al., 2008; PETERSEN; VAKKALANKA; KUZNIARZ, 2015).

Portanto, um mapeamento sistemático da literatura foi desenvolvido seguindo minuciosamente as diretrizes estipuladas pelo protocolo desenvolvido para este estudo. As seções que se sucedem estão organizadas da seguinte forma: a Seção 3.1 traz o plano de pesquisa do mapeamento desenvolvido. Em seguida, a Seção 3.2 descreve como se deu a condução do mapeamento mediante o protocolo adotado. Na Seção 3.3 são apresentados os resultados obtidos por meio da condução do mapeamento, além da interpretação destes, levando à Seção 3.4, a qual elucida as oportunidades identificada e como estas podem ser exploradas. Por fim, na Seção-3.5 estão descritas as ameaças à validade do mapeamento e quais estratégias foram adotadas para estas serem mitigadas.

3.1 Plano de Pesquisa

Esta seção descreve o protocolo de revisão adotado. Como mencionado anteriormente (Capítulo 3), foi escolhido o mapeamento sistemático da literatura como método de pesquisa para a identificação de oportunidades de pesquisa. Segundo DIANE (2016), esta metodologia tende a produzir resultados mais confiáveis visto que reduz vieses de autores mediante a aplicação de

um rigoroso processo de revisão. O protocolo utilizado neste estudo foi desenvolvido seguindo diretrizes bem aceitas na literatura (PETERSEN et al., 2008; PETERSEN; VAKKALANKA; KUZNIARZ, 2015), as quais têm sido amplamente utilizadas e validadas por estudos anteriores (GONÇALES et al., 2015; GONCALES et al., 2019; GONÇALES et al., 2019; VIEIRA; FARIAS, 2020a; CARBONERA; FARIAS; BISCHOFF, 2020; BISCHOFF et al., 2021)

A fase de planejamento (Seção 3.1) compreende todos os procedimentos necessários para a confecção do mapeamento. A Seção 3.1.1 detalha as questões de pesquisa as quais visa-se responder mediante os resultados do mapeamento. Na Seção 3.1.2 é descrita a estratégia adotada para a busca dos estudos-alvo. Em seguida, são definidos os critérios para a inclusão e exclusão de trabalhos durante o processo de filtragem (Seção 3.1.3). Por último, é definido como se dará a abordagem para a extração dos dados contido nos artigos filtrados.

3.1.1 Questões de Pesquisa

Além dos objetivos gerais e específicos apresentados na Seção 1.3, as Questões de Pesquisa guiarão o pesquisador no desenvolvimento do seu Mapeamento Sistemático da Literatura, auxiliando na identificação de possíveis estudos que venham a convergir com a temática do estudo proposto (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). As Questões de Pesquisa utilizadas neste estudo são apresentadas na Tabela 2, trazendo consigo os seus propósitos e as variáveis associadas a estas.

3.1.2 Estratégia de Busca

A segunda etapa na metodologia do Mapeamento Sistemático da Literatura é a definição da estratégia de busca, a qual pretende alcançar uma amostra representativa de estudos que possam responder de forma generalizada às Questões de Pesquisa definidas na Tabela 2. A Estratégia de Busca utilizada neste estudo foi concebida em duas etapas: (1) seleção das bases de dados digitais relacionadas à temática do estudo, e (2) construção de uma *string* de busca e suas variantes, respeitando as peculiaridades dos mecanismos de busca presentes em cada base de dados.

A Tabela 3 apresenta as bases de dados eletrônicas utilizadas. Elas foram escolhidas por serem amplamente utilizadas e por estudos anteriores de mapeamento da literatura terem demonstrado efetividade das mesmas (BISCHOFF et al., 2019; GONÇALES et al., 2019). Ademais, a seleção de bases de dados foi embasada na cobertura dos seus mecanismos de busca, os quais retornaram artigos dos principais periódicos de Engenharia de Software.

A *string* de busca é um construto oriundo da combinação de termos primários (*major terms*) e seus sinônimos (*minor terms*), sendo utilizada como insumo para os mecanismos de busca das bases de dados selecionadas. Tal insumo permite que os mecanismos de busca retornem uma coleção de estudos potencialmente relevantes contributivos para o desenvolvimento do

Tabela 2: Questões de Pesquisa (QPs), suas descrições e variáveis relacionadas

Questão de Pesquisa	Motivação	Variável
QP01: Como seria um **esquema de classificação** de estudos envolvendo IDEs e dados psicofisiológicos?	Construir um esquema de classificação passível de ser utilizados em estudos que apliquem o uso de dados psicofisiológicos e IDEs.	Esquema
QP02: Que tipos de **dispositivos** têm sido utilizados na coleta de dados psicofisiológicos?	Listar os dispositivos utilizados nos estudos selecionados para a coleta de dados.	Dispositivos
QP03: Que tipos de **atributos** são extraídos pelos dispositivos utilizados?	Elucidar as categorias de dados extraídos e o que representam no escopo dos estudos analisados.	Atributos
QP04: Como se dá a **integração** entra IDEs e os dados extraídos?	Explicar como os dados extraídos podem ser integrados nas IDEs.	Integração
QP05: Com quais **propósitos** os dados extraídos têm sido utilizados dentro das IDEs?	Trazer o que os dados extraídos visam explanar dentro das IDEs no contexto de Engenharia de Software.	Propósitos
QP06: Quais são as principais **contribuições** dos estudos?	Explicitar as contribuições de cada estudo no que tange a utilização de dados psicofisiológicos no contexto de Engenharia de Software.	Contribuições
QP07: Qual a **natureza** dos estudos analisados?	Compreender a natureza do estudo quanto ao seu objetivo.	Natureza
QP08: Em quais **veículos de publicação** os estudos selecionados foram divulgados?	Identificar os veículos de publicação onde os estudos foram publicados entre os anos de 2009 e 2019.	Veículos de Publicação

Fonte: Elaborado pelo autor.

Tabela 3: Bases de dados e seus endereços digitais

Base de Dados	Endereço Eletrônico
ACM Digital Library	http://portal.acm.org/
DBLP	https://dblp.uni-trier.de/
IEEE Xplore	http://ieeexplore.ieee.org
Science Direct	https://www.sciencedirect.com/
Scopus	https://www.scopus.com/
Semantic Scholar	https://www.semanticscholar.org/
Springer Link	http://www.springerlink.com/

Fonte: Elaborado pelo autor.

Mapeamento Sistemático da Literatura. A Tabela 4 traz a relação de termos principais e seus sinônimos que foram utilizados no desenvolvimento deste estudo.

Com o objetivo de produzir uma *string* de busca funcional, foram adotados os seguintes

passos para o desenvolvimento da mesma: (1) definir os termos principais; (2) pesquisar pelos sinônimos relacionados ou ainda, palavras com equivalência semântica, e (3) combinar os termos principais aos seus sinônimos utilizando operadores lógicos, tais como “AND” e “OR”. Aplicando-se as características citadas, chegou-se na seguinte *string* de busca:

> *(eeg* ***OR*** *electroencephalography* ***OR*** *neural* ***OR*** *brain* ***OR*** *cognitive* ***OR*** *psychometric* ***OR*** *bioinformatic)* ***AND*** *(software* ***OR*** *developer* ***OR*** *programmer* ***OR*** *professional* ***OR*** *“software engineering”)* ***AND*** *(ide* ***OR*** *code* ***OR*** *editor* ***OR*** *“integrated development environment”)*

Tabela 4: Termos primários da *string* de busca e os seus sinônimos

Termo Principal	Sinônimos
Neuroscience	eeg, fmri, electroencephalography, “functional magnetic resonance imaging”, neural, brain, cognitive, psychometric, bioinformatic
Software Engineering	software, developer, programmer, professional, software engineering
IDE	ide, code, editor, integrated development environment
Data Analysis	“data analysis”, “signal analysis”, analytic, “signal processing”

Fonte: Elaborado pelo autor.

Devido às peculiaridades sintáticas provenientes de cada mecanismo de busca contido nas bases de dados, a *string* de busca inicial necessitou ser adaptada individualmente de forma a prover um melhor resultado sem perder o seu valor semântico (Apêndice A).

3.1.3 Critérios de Inclusão e Exclusão

Uma vez selecionadas as bases de dados e construída a *string* de busca, a etapa subsequente no processo de desenvolvimento do Mapeamento Sistemático da Literatura foi composta pela definição dos Critérios de Inclusão e os Critérios de Exclusão (CI e CE, respectivamente).

Os Critérios de Inclusão foram aplicados com o objetivo de incluir estudos no processo de filtragem após a sua coleta inicial nas bases de dados (PETERSEN; VAKKALANKA; KUZNIARZ, 2015), em contrapartida, os Critérios de Exclusão agiram como um mecanismo de descarte de estudos que não representassem respostas qualificadas para as Questões de Pesquisa (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). Sendo assim, foram definidos como Critérios de Inclusão os seguintes itens:

- **CI01:** Publicação entre os anos de 2009 e 2019;
- **CI02:** Escrita na língua inglesa;
- **CI03:** Disponível de forma completa nas bases de dados digitais;

- **CI04:** Relação com a *string* de busca e Questões de Pesquisa;

Ademais, para a filtragem dos estudos previamente selecionados por meio dos Critérios de Inclusão, foram aplicados os seguintes Critérios de Exclusão:

- **CE01:** Coerência com a *string* de busca mas fora do contexto da pesquisa;
- **CE02:** Inexistência de resumo;
- **CE03:** Apenas um resumo ou *Call for Paper* de conferências;
- **CE04:** Sem relação com o domínio de Engenharia de Software;
- **CE05:** Cópias/duplicatas ou versões antigas de estudos já selecionados;
- **CE06:** Impossibilidade de acesso ao artigo completo;
- **CE07:** Em desacordo com a motivação da pesquisa;

3.1.4 Estratégia para a Extração de Dados

Uma vez definidos os critérios de inclusão de exclusão dos trabalhos selecionados, o próximo passo no processo de mapeamento foi a definição de como seriam extraídos os dados dos artigos potencialmente relevantes. Para tanto, foi criado um formulário para extração de dados (Figura 10) e um conjunto de esquemas de classificação, permitindo assim que as questões de pesquisa do mapeamento (Seção 3.1.1) fossem apropriadamente respondidas.

Esquema de classificação: Para Petersen, Vakkalanka e Kuzniarz (2015), a extração de dados dos estudos selecionados dá-se mediante esquemas de classificação previamente definidos, estes sendo responsáveis por definirem categorias nas quais os estudos identificados deverão ser classificados em concordância com o seu conteúdo. Para os autores, existem dois esquemas que são utilizados para a identificação de categorias utilizadas na classificação dos estudos:

- **Esquema Independente de Tópico:** categorias genéricas são utilizas de forma a prover uma visão geral em torno dos estudos selecionados e a sua temática. Usualmente os autores recomendam a utilização de categorias já adotadas em outros mapeamentos sistemáticos da literatura;
- **Esquema Dependente de Tópico:** questões específicas emergem enquanto os estudos são filtrados de forma iterativa, trazendo assim uma visão detalhada do seu conteúdo, sendo uma peculiaridade presente em cada mapeamento sistemático da literatura;

Durante a identificação das categorias utilizadas para a classificação dos estudos, adotou-se o *Esquema Independente de Tópico*, tomando-se como base para a extração de categorias, outros mapeamentos sistemáticos previamente produzidos e relacionados ao tópico de pesquisa

(GONCALES et al., 2019). Conforme as categorias eram identificadas, novas variantes, estas com maior especificidade, foram surgindo, aumentando assim tanto a gama, como a granularidade de possíveis categorias de classificação. Sendo assim, posteriormente adotou-se o *Esquema Dependente de Tópico*, dado que, segundo as definições de Petersen, Vakkalanka e Kuzniarz (2015), tal esquema de classificação é sustentado pela extração iterativa de categorias e pelas peculiaridades de cada estudo. Por intermédio de tal abordagem, foram identificadas as seguintes categorias utilizadas para a classificação dos trabalhos identificados:

- **Esquema de Classificação (QP01):** esta categoria tem por objetivo guiar o estudo na construção de um esquema capaz de auxiliar na identificação de estudos que utilizem dados psicofisiológicos integrados às IDEs no âmbito de Engenharia de Software. Ao contrário das demais Questões de Pesquisa, esta foi construída a medida que as respostas das demais emergiam.

- **Dispositivos (QP02):** representa quais tipos de dispositivos foram utilizados nos estudos selecionados para a coleta de dados psicofisiológicos e outras métricas biomédicas. Os itens a seguir representam as categorias extraídas utilizadas para a classificação dos estudos neste aspecto:

  - Eletroencefalograma (EEG): tipo de sensoriamento utilizado para medir e registrar a atividade elétrica do cérebro;
  - *Eye-tracker*: dispositivo utilizado para medir a posição ocular do indivíduo, assim como os seus movimentos;
  - Imagem por Ressonância Magnética Funcional (fMRI): mede a atividade cerebral por associação às alterações no fluxo sanguíneo e variações na saturação de oxigênio;
  - Espectroscopia Funcional de Infravermelho Próximo (fNIRS): assim como a fMRI, a fNIRS coleta medidas a atividade cerebral por meio das respostas hemodinâmicas;
  - Atividade Eletrodermal (EDA): realiza séries de medição em torno da variação galvânica da pele;
  - Combinação de Dispositivos: aplicável quando mais de um dispositivo foi utilizado no experimento, também aplicado à dispositivos de sensoriamento complementares, como pulseiras de medição e faixas peitorais equipadas com sensores de monitoramento cardíaco e respiratório;
  - Sem Dispositivos: quando a natureza do estudo não demandou que dispositivos sensores fossem utilizados.

- **Atributos (QP03):** refere-se às características baseadas no contexto associadas aos dados coletados por meio dos dispositivos sensores. Os atributos identificados no estudos selecionados foram classificados em:

  - Fixação/Atenção: visa indicar por quanto tempo o foco visual de um indivíduo permanece em uma Área de Interesse, geralmente associada com a atenção (BEELDERS; Du Plessis, 2016), entretanto, sem medidas complementares de Estados Mentais, não é possível inferir de forma clara a relação deste atributo com a atenção focada.
  - Estados Mentais: representa um conjunto de atributos responsáveis pela relação entre processos cognitivos (atenção, aprendizado, meditação, sonolência, etc...) e a atividade elétrica dos neurônios, geralmente expressados por ondas cerebrais, tais como Alfa, Beta, Gama, Delta e Teta (SZU et al., 2013).
  - Atividade Neural: esta categoria abrange os conjuntos de padrões de ativações neurais e as suas semânticas associadas (RD et al., 1997).
  - *Stress*/Carga Cognitiva: frequentemente associado com a memória de trabalho de um indivíduo enquanto este executa ou tarefa ou mesmo um conjunto de tarefas que demandem sua troca de contexto. (SWELLER, 1988).
  - Esforço: tem por objetivo explanar a coleção de métricas biológicas ligadas ao aporte de recursos providos pelo corpo durante da execução de tarefas cognitivas (VELTMAN; GAILLARD, 1998).
  - Atributos Combinados: aplicável aos estudos que realizaram a análise três ou mais atributos combinados.
  - Sem Atributos: categoria adotada nos estudos em que não foram avaliados atributos específicos em torno dos dados psicofisiológicos.

- **Integração (QP04):** esta categoria indica se o estudo analisado implementou algum mecanismo capaz de integrar os dados coletados às IDEs utilizadas.

  - Suportado: aplicável quando o estudo analisado utilizou alguma ferramenta capaz de realizar a integração entre os dados psicofisiológicos coletados e a IDE utilizada;
  - Não Suportado: quando um estudo recorreu a dados psicológicos, assim como de IDEs, mas a análise destes deu-se de forma independente em relação à IDE;
  - Não Aplicável: categoria adotada em estudos analisados que fazem uso de dados psicofisiológicos, mas não possuem em seu contexto o uso de IDEs;

- **Propósitos (QP05):** traz um ponto de vista geral em torno dos principais objetivos estipulados pelos autores em cada estudo, no que tange a utilização dos dados coletados e os seus correlatos atributos. Sendo assim, as categorias identificadas foram enumeradas da seguinte forma:

  - Compreensão de Código-fonte e Estratégias de Depuração: esta categoria de propósitos tenta quantificar a facilidade na compreensão de um programa, do ponto de

vista do desenvolvedor, ademais analisa as estratégias adotadas por este durante o processo de depuração;

- Interfaces Cérebro Computador (BCI[1]): categoria aplicada quando o principal propósito do estudo é o de desenvolver uma *interface* capaz de possibilitar a interação entre humanos e computadores por meio do uso de sinais oriundos das atividades neurais;
- Carga Cognitiva: relacionados aos estudos que conduzem medições ligadas ao esforço mental de um desenvolvedor enquanto este executa tarefas complexas;
- Melhoria na Produtividade: visa encontrar formas de prover melhores ferramentas e melhoras processos no contexto de Engenharia de Software e na qualidade de vida do desenvolvedor;
- Fora do Contexto: aplicável nos casos em que o propósito do estudo não era ligado diretamente à utilização de dados psicofisiológicos no contexto de Engenharia de Software.

- **Contribuições (QP06):** diferenciando-se das demais, esta categoria adotará duas abordagens: a primeira de caráter subjetivo, apresentará um texto sintético com as principais contribuições identificadas pelos autores para o desenvolvimento deste trabalho. Sob a mesma óptica, a segunda abordagem buscará generalizar as contribuições identificadas por meio das seguintes categorias, de acordo com o valor retornado para o desenvolvimento do presente estudo:

  - Métrica: o estudo apresenta, mesmo que indiretamente, alguma métrica oriunda dos dados psicofisiológicos que pudesse ser correlacionada a algum dos atributos identificados e com algum dos propósitos apresentados anteriormente;
  - Ferramenta: trouxe alguma ferramenta tanto para a coleta de dados, como para processamento/análise destes;
  - Modelo: quando o estudo contribui para a apresentação de um modelo conceitual ou prático de arquitetura ou de experimentação envolvendo o uso dos dados psicofisiológicos;
  - Método: apresenta uma etapa isolada que possa ser aproveitado nos aspectos de coleta, processamento ou análise de dados psicofisiológicos;
  - Processo: similar ao item *método*, entretanto esta categoria engloba uma sequência lógica e interdependente de métodos ou etapas para coleta, processamento e/ou análise dos dados coletados;
  - Integração: possibilita a identificação de alguma abordagem de integração dos dados psicofisiológicos coletados com os ambientes integrados de desenvolvimento.

[1] Brain Computer Interface

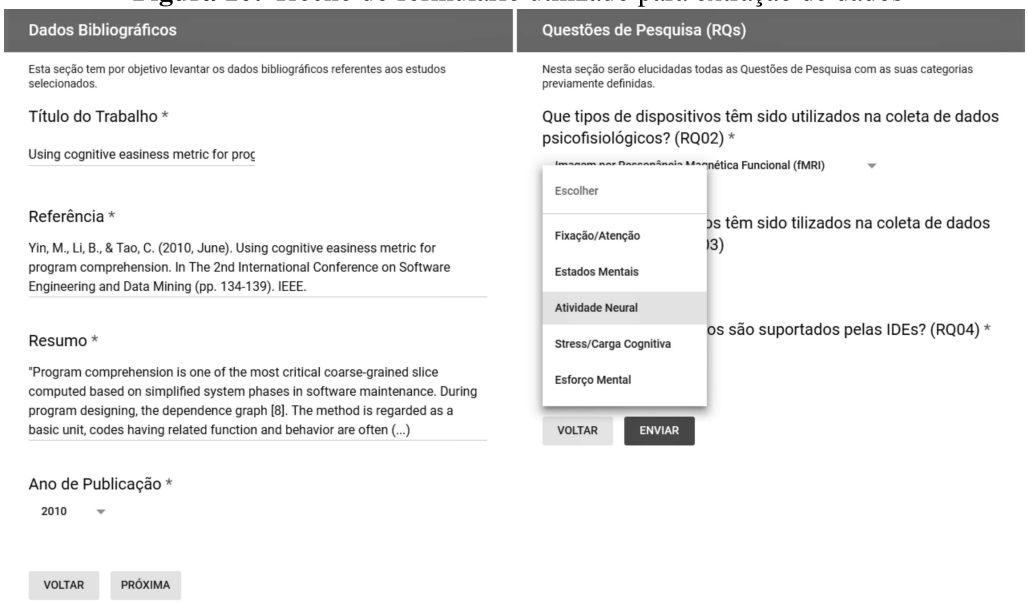
- **Natureza (QP07):** a natureza refere-se ao tipo de pesquisa desenvolvida quanto ao seu objetivo principal e desenvolvimento. O estilo de classificação frequentemente adotado na literatura é o de (WIERINGA et al., 2006), o qual determinam as seguintes categorias:
  - Proposta de Solução: relativo aos estudos que propõem novas técnicas para solução de problemas existentes, ou ainda, a revisão de técnicas já adotadas;
  - Pesquisa de Avaliação: nesta categoria aplicam-se os estudos que avaliam técnicas e práticas aplicadas na indústria em problemas específicos;
  - Pesquisa de Validação: estudos destinados à avaliação de técnicas inovadoras, mas que não foram ainda validadas quanto à sua aplicação na indústria;
  - Artigo Filosófico: estudos os quais visam trazer uma nova perspectiva no que se refere às estruturas de conhecimentos já consolidados;
  - Artigo de Experiência: este tipo de trabalho discute como outro autor resolveu um problema aplicando uma técnica, na prática, sem a necessidade de também utilizar a técnica durante o desenvolvimento do estudo.
  - Artigo de Opinião: engloba todos os estudos que apresentam opiniões dos autores, sendo estas embasadas por outros estudos acerca de um determinado tópico de pesquisa.
  - Experimento Controlado: nesta natureza de estudos, os autores realizaram experimentos em ambiente controlado seguindo metodologias bem consolidadas e reprodutíveis.
- **Veículos de Publicação (QP08):** identifica em quais veículos de publicação os estudos selecionados tem sido publicados, como conferências, oficinas, periódicos científicas, etc., como o objetivo de inferir o nível te maturidade do tópico na comunidade científica.

Formulário de extração de dados: O processo de extração de dados consiste na leitura completa dos artigos previamente filtrados e na tabulação dos dados encontrados em concordância com os esquemas de classificação definidos. Para isto, foi utilizado um formulário de extração de dados (Figura 10) inspirado no modelo adotado por GONCALES et al. (2019), permitindo uma fácil classificação dos estudos selecionados e geração de dados quantitativos sobre estes. A utilização de formulários para a extração dos dados em mapeamentos sistemáticos da literatura tem se mostrado como uma solução bem recebida nesta natureza de estudos (GONÇALES et al., 2015; GONCALES et al., 2019; GONÇALES et al., 2019; VIEIRA; FARIAS, 2020a; CARBONERA; FARIAS; BISCHOFF, 2020; BISCHOFF et al., 2021).

3.2 Execução do Mapeamento

Esta seção tem por finalidade descrever como foi conduzido o processo de filtragem dos estudos potencialmente relevantes obtidos nas bases de dados eletrônicas apresentadas na Tabela 3

Figura 10: Trecho do formulário utilizado para extração de dados



Fonte: Elaborado pelo autor, baseado em GONCALES et al. (2019)

da Seção 3.1.2. O processo de filtragem utilizado neste trabalho deu-se em oito etapas distintas, apresentadas por meio da Figura 11. Excetuando-se a etapa de Pesquisa Inicial, a qual utilizou apenas os Critérios de Inclusão na triagem, as demais utilizaram agrupamentos dos Critérios de Exclusão como diretivas para a remoção dos estudos inicialmente selecionados.

A seguir estão descritas as etapas adotadas, assim como as suas diretrizes de condução.

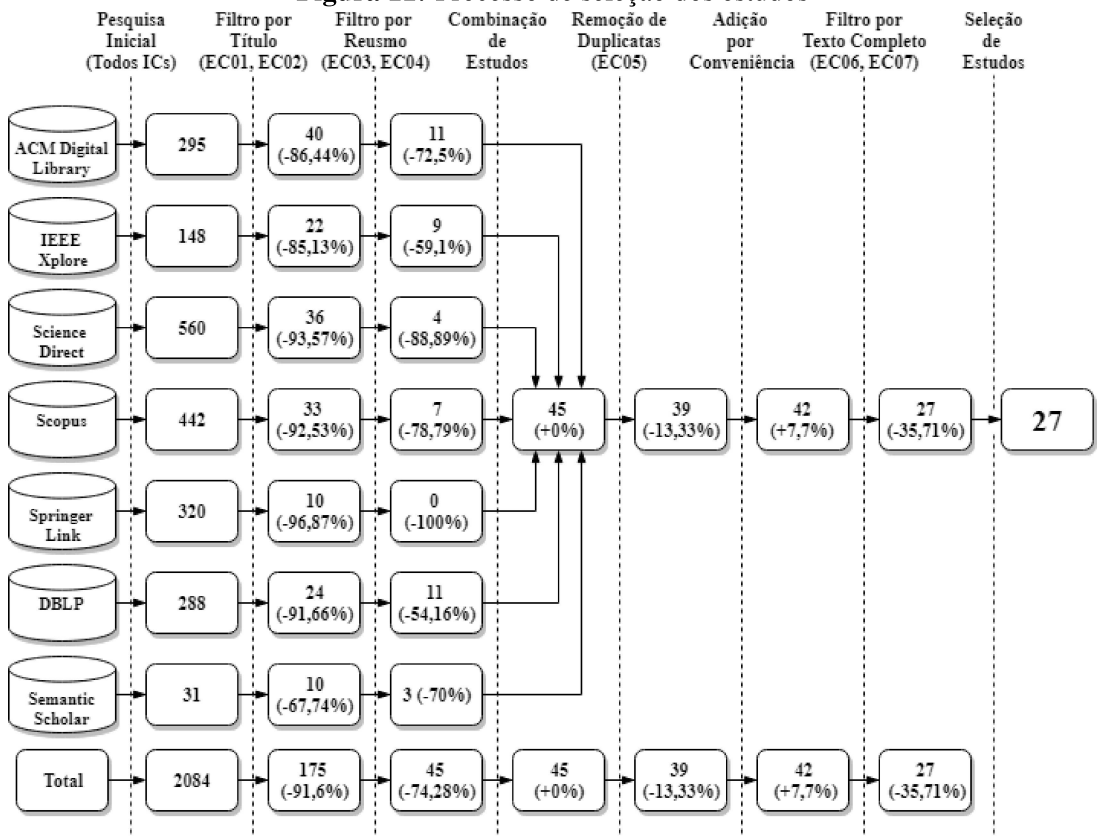
- **Pesquisa Inicial:** esta etapa caracterizou-se pela inserção da *string* de busca (Apêndice A) nas bases de dados escolhidas (Tabela 3 da Seção 3.1.2), combinada aos filtros compostos parametrizados segundo os Critérios de Inclusão (Seção 3.1.3), com isto, a busca retornou um montante de 2084 artigos a serem filtrados, agrupados segundo suas respectivas bases de dados, conforme ilustrado na Figura 11.

- **Filtro por Título:** nesta etapa do processo de filtragem foram analisados os títulos dos artigos resultantes da Pesquisa Inicial. Foram utilizados para filtragem os Critérios de Exclusão CE01 e CE02 (vide Seção 3.1.3). Desta forma, o montante inicial de 2084 artigos foi reduzido para um total de 175, representando um redução de 91,6% do volume de estudos inicialmente selecionados (Figura 11).

- **Filtro por Resumo:** responsável pela precisão na escolha de estudos relevantes para o desenvolvimento do Mapeamento Sistemático da Literatura, esta etapa consistiu na leitura dos resumos de todos os 175 artigos resultantes. Foram aplicados como filtros os Critérios de Exclusão CE03 e CE04 (Seção 3.1.3), reduzindo o volume de estudos em 74,28%, chegando assim a um total de 45 artigos (Figura 11).

- **Combinação de Estudos:** conforme ilustrado pela Figura 11, as etapas de filtragem ante-

Figura 11: Processo de seleção dos estudos



Fonte: Elaborado pelo autor.

riores aplicaram-se de forma individual à cada uma das bases de dados eletrônicas. Foram removidos os rótulos que ligavam os artigos às suas bases de dados de origem, fazendo com que estes compusessem um único *corpus* literário, permitido assim a execução da etapa subsequente.

- **Remoção de Duplicatas:** com os artigos já combinados, aplicou-se o Critério de Exclusão CE05, responsável pela remoção de duplicatas de estudos e/ou versões antigas dos mesmos. Tal filtragem acarretou na redução de 13,39% do total de estudos, resultando assim em um *corpus* literário de 39 artigos (Figura 11).

- **Adição por Conveniência:** visando um aumento no volume de estudos analisados, foram adicionados por conveniência 3 artigos relacionados com a temática e dentro dos Critérios de Inclusão e Exclusão utilizados anteriormente, implicando em um aumento de 7,7% do total, ou ainda, 42 artigos (Figura 11). Tal adição deu-se por conhecimento dos autores e através de análises encadeadas das referências presentes nos artigos já analisados (*snowballing*), seguindo assim as recomendações de (PETERSEN; VAKKALANKA; KUZNIARZ, 2015) para seleção.

- **Filtro pelo Texto Completo:** nesta fase, os artigos provenientes da etapa anterior foram lidos de forma integral, aplicando-se os Critérios de Exclusão CE06 e CE07 (Seção 3.1.3)

acarretando remoção de 35,71% dos artigos e gerando um montante final de 27 trabalhos (Figura 11).

- **Seleção dos Estudos:** a fase final do processo resultou na seleção de 27 artigos, nomeados neste trabalho como Estudos Primários, os quais estão dispostos no Apêndice B. Ademais, por meio das estratégias descritas na Seção 3.1.4, foram extraídos os dados de forma a possibilitar a obtenção das respostas para as Questões de Pesquisa.

3.3 Resultados do Mapeamento

Nesta seção, estão dispostos os resultados obtidos por meio da execução processo de filtragem descrito na Seção 3.2, segundo o delineamento discutido na Seção 3.1. Para apresentação dos dados quantitativos referentes a cada Questão de Pesquisa, foram utilizadas tabelas contendo a distribuição Estudos Primários, representados pelos seus identificadores enumerados no Apêndice B em suas respectivas categorias de respostas.

3.3.1 QP01: Como seria um esquema de classificação de estudos envolvendo IDEs e dados psicofisiológicos?

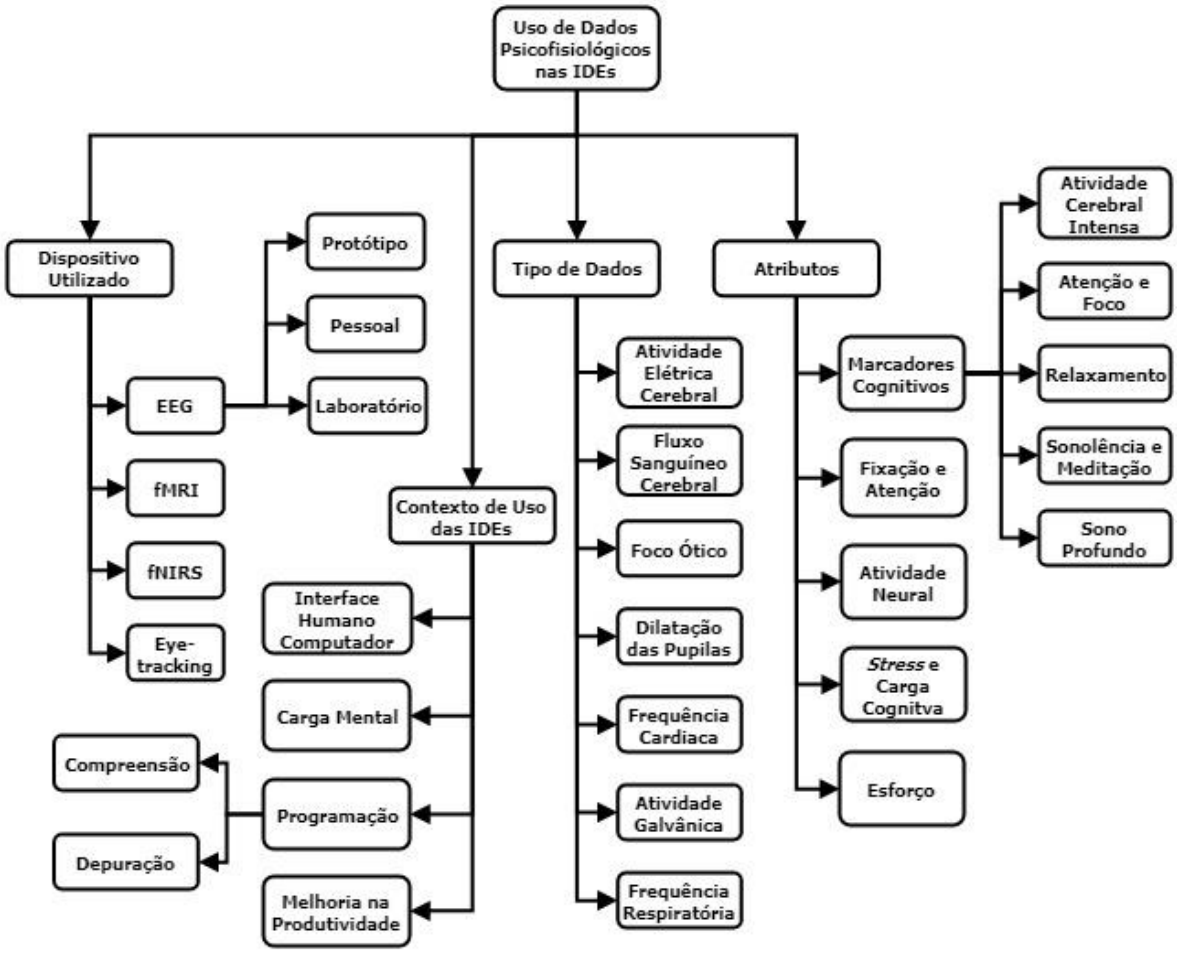
Segundo ilustrado na Figura 12, por meio do desenvolvimento deste estudo, foi possível produzir um esquema de classificação que pudesse ser utilizado em trabalhos cujo objeto de estudo fossem os dados psicofisiológicos e os ambientes integrados de desenvolvimento. É possível observar que o segundo nível do esquema de classificação foi desenvolvido tendo como base os critérios de classificação dos estudos abordados no desenvolvimento deste trabalho, sendo eles: (1) dispositivos que foram utilizados nos estudos para coletas de dados psicofisiológicos; (2) o contexto em que os dados coletados foram utilizados no escopo das IDEs; (3) os tipos de dados que foram coletados pelos dispositivos utilizados e (4) o que os dados coletados são capazes de indicar.

Ainda que tal esquema de classificação possa trazer uma visão geral no âmbito deste trabalho, e facilitar o processo de classificação e identificação de estudos, é válido ressaltar que tal ferramenta foi desenvolvida com o propósito de ser aplicada neste estudo, sendo assim, é necessária a sua validação e/ou adaptação quando utilizada no desenvolvimento de trabalhos similares.

3.3.2 QP02: Que tipos de dispositivos têm sido utilizados na coleta de dados psicofisiológicos?

A Tabela 5 apresenta os resultados referentes à QP02. É possível observar que na maioria dos estudos envolvendo a coleta de dados psicofisiológicos (29,63%), são utilizados eletroencefalógrafos (EEG) como dispositivo de coleta de sinais. Tal fenômeno pode ser atribuído à

Figura 12: Esquema de classificação de estudos relacionados ao uso de dados psicofisiológicos e IDEs



Fonte: Elaborado pelo autor.

redução do custo dos artefatos de eletroencefalografia, além do desenvolvimento de versões acessíveis pelo público não especializado, democratizando o seu uso em diversos contextos (ROSTAMI et al., 2015). Ademais, estudos sinalizam que a eletroencefalografia, por ondas cerebrais provenientes da ativação neural, pode traduzir processos cognitivos durante o desenvolvimento de tarefas diversas, inclusive àquelas ligadas ao desenvolvimento de software, compreensão de código e estratégias de depuração (ROSTAMI et al., 2015; FRITZ et al., 2014; FRITZ; MULLER, 2016; RADEVSKI; HATA; MATSUMOTO, 2015).

Em seguida, **Eye-trackers** e **Imagem por Ressonância Magnética Funcional** representaram, igualmente, 18,52% dos dispositivos utilizados nos Estudos Primários. Nota-se ainda o uso de **Espectroscopia Funcional de Infravermelho Próximo** (7,41%) como dispositivo de coleta.

Representaram 14,81% dos estudos analisados, àqueles que utilizaram mais de um dispositivo, usualmente EEGs combinados com **Eye-trackers** ou ainda, dispositivos complementares, tais como pulseiras de medição e faixas peitorais equipadas com sensores de frequência car-

Tabela 5: Distribuição dos estudos por dispositivos utilizados (QP02)

Dispositivo	Trabalhos	Percentual	Estudos Primários
Sem Dispositivos	3	11,11%	S15, S26, S27
Eye-tracker	5	18,52%	S02, S03, S12, S16, S20
Eletroencefalograma	8	29,63%	S01, S04, S11, S14, S18, S19, S23, S25
Imagem por Ressonância Magnética Funcional	5	18,52%	S05, S08, S17, S21, S22
Espectroscopia Funcional de Infravermelho Próximo	2	7,41%	S13, S24
Combinação de Dispositivos	4	14,81%	S06, S07, S09, S10

Fonte: Elaborado pelo autor.

diorrespiratória, sendo estes responsáveis por complementar as medições em cenários onde o **Eye-tracker** e o EEGs não são capazes de indicar, de forma isolada, atributos como *stress* e atenção focada (BEELDERS; Du Plessis, 2016).

Dada a natureza de alguns dos Estudos Primários, assim como a metodologia adotada, não foi constatada a utilização de dispositivos para a coleta de dados. Tal gama de estudos representaram 11,11% do volume total.

3.3.3 QP03: Que tipos de atributos são extraídos pelos dispositivos utilizados?

No que diz respeito aos atributos analisados, a Tabela 6 elucida as suas distribuições nos Estudos Primários. Observa-se a predominância (29,63%) de trabalhos que tiveram como objeto de estudo a atividade neural dos indivíduos. Tais trabalhos recorreram a dispositivos como EEGs, fNIRs e fMRI para avaliar quais regiões cerebrais eram ativadas durante a execução de seus experimentos, além de relacionar determinadas semânticas às atividades cerebrais observadas (RD et al., 1997).

Tabela 6: Distribuição dos estudos por atributos analisados (QP03)

Atributo	Trabalhos	Percentual	Estudos Primários
Sem Atributos	3	11,12%	S15, S26, S27
Fixação/Atenção	5	18,52%	S02, S03, S12, S16, S20
Estados Mentais	7	25,93%	S01, S04, S14, S18, S19, S23, S25
Atividade Neural	8	29,63%	S05, S08, S11, S13, S17, S21, S22, S24
Atributos Combinados	2	7,40%	S09, S10
Fixação/Atenção, Atividade Neural	2	7,40%	S06, S07

Fonte: Elaborado pelo autor.

Estudos que dissertaram em torno dos **Estados Mentais** representaram 25,93% do Estudos Primários. Nestes estudos optou-se primordialmente pelo uso de EEGs para a obervação dos estados mentais, tendo em vista a íntima ligação destes à ondas cerebrais (Alfa, Beta, Gama, Delta e Theta), além de ser um tópico que se mostrou muito presente na literatura especializada (SZU et al., 2013).

A **Fixação/Atenção** foi o atributo estudado em 18,52% dos estudos presentes no *corpus* literário adotado neste mapeamento sistemático. Usualmente a fixação foi atribuída pelos estudos ao foco ocular em pontos de interesse, predominantemente mensurado através de **Eye-trackers**, vide dados apresentados na Tabela 5 entretanto, nos casos em que a atenção era objeto de estudos, fez-se o uso de sensores complementares de forma concomitante.

Em 7,40% do trabalhos, realizou-se a análise combinada dos atributos de **Fixação/Atenção** e **Atividade Neural**. Neste estudos observou-se que o enfoque dos autores permeava a compreensão de código-fonte, para tanto, foram utilizados **Eye-trackers** para a coleta de dados relativos ao posicionamento e/ou foco ocular e fNIRS, a fim de identificar a ativação neural de áreas cerebrais ligadas ao aprendizado e leitura (FAKHOURY et al., 2018; FAKHOURY, 2018).

Estudos que analisaram mais de três atributos em seus experimentos correspondem a 7,40% do total. Nestes trabalhos foram utilizadas situações de maior especificidade se comparados aos demais, tais como a mensuração de **Fixação/Atenção**, **Estados Mentais** e atributos complementares (frequência cardiorrespiratória) durante a execução de tarefas cotidianas por parte de um desenvolvedor (FRITZ et al., 2014), ou ainda, a combinação de **Estados Mentais** com atributos oriundos de sinais mioelétricos (piscar de olhos, tensão temporomandibular), a fim utilizá-los na comparação do nível de compreensão de código-fonte entre desenvolvedores iniciantes e experientes (LEE et al., 2016).

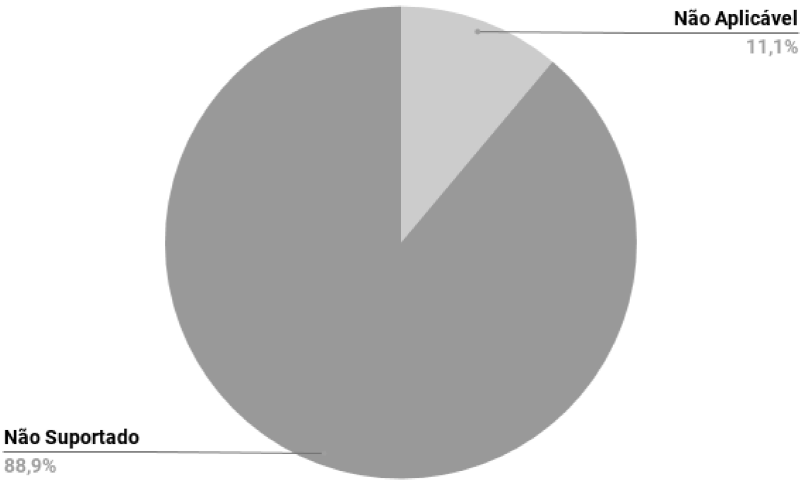
Ainda que 88,88% dos estudos tenham aplicado análises de atributos relacionados aos dados psicofisiológicos coletados, em 11,12% destes não houve a presença de análises em torno de tais atributos. Tal característica se deu pela natureza destes trabalhos, além do processo metodológico adotado em seu desenvolvimento.

3.3.4 QP04: Como se dá a integração entre IDEs e os dados extraídos?

Esta Questão de Pesquisa visava obter como resposta quais os tipos de suporte que as IDEs utilizadas nos experimentos proveram para a integração dos dados psicofisiológicos coletados, possibilitando assim a sua análise no contexto de Engenharia de Software pelos pesquisadores. A Figura 13 ilustra como se deu a divisão das respostas para esta Questão de Pesquisa.

Observa-se que, do total de 27 estudos analisados, 11,1% destes fizeram algum tipo de uso de dados psicofisiológicos, entretanto, sem a utilização de IDEs em seus experimentos, conforme discriminado anteriormente na Seção 3.1.4 e ilustrado por meio da Figura 13. Por outro lado, cerca de 88,9% dos estudos foram enquadrados na categoria **Não Suportado**, ou seja, re-

Figura 13: Suporte à integração de dados psicofisiológicos nas IDEs (QP04)



Fonte: Elaborado pelo autor.

alizaram experimentos envolvendo IDEs e também a análise de dados psicofisiológicos. Ainda sim, nenhum destes estudos especificou mecanismos capazes de integrar os dados coletados e os Ambiente Integrados de Desenvolvimento (IDEs), sendo assim, as análises foram realizadas de forma externa e por meio de ferramentas especializadas.

3.3.5 QP05: Com quais propósitos os dados extraídos têm sido utilizados dentro das IDEs?

Neste item estão dispostos os propósitos adotados pelos autores nos Estudos Primários, relativos ao uso de dados psicofisiológicos no contexto de Engenharia de Software. Mediante os dados elucidados pela Tabela 7, constatou-se que, em sua maioria (66,70%), os Estudos Primários tinham como enfoque analisar a **Compreensão de Código-fonte e as Estratégias de Depuração** utilizadas pelos desenvolvedores nos experimentos realizados.

Tabela 7: Distribuição dos estudos por propósito adotado (QP05)

Propósito	Trabalhos	Percentual	Estudos Primários
Compreensão de Código-fonte e Estratégias de Depuração	18	66,70%	S02, S03, S05, S06, S07, S08, S09, S12, S13, S14, S16, S17, S20, S21, S22, S25, S26, S27
Interfaces Cérebro Computador	2	7,41%	S01, S23
Carga Cognitiva	4	14,81%	S04, S11, S19, S24
Melhoria na Produtividade	2	7,41%	S10, S18
Fora do Contexto	1	3,70%	S15

Fonte: Elaborado pelo autor.

Observou-se ainda que, em 14,81% dos Estudos Primários, o propósito focal do trabalho foi

a avaliação da do desenvolvedor enquanto este executava um série de atividades relacionadas ao seu contexto. É válido ressaltar que, apesar da possibilidade de avaliar a Compreensão de Código-fonte e a estratégia de depuração de forma individual, as medições relacionadas a estes propósitos possuem uma maior validade quando atreladas à **Carga Cognitiva**, segundo levantamentos desenvolvidos por outros autores (GONCALES et al., 2019).

Constituindo 7,41% dos Estudos Primários estavam àqueles cujo propósito principal era a busca pela **Melhoria na Produtividade** dos desenvolvedores mediante o uso de dados psicofisiológicos e IDEs.

Trabalhos envolvendo o desenvolvimento de **Interfaces Cérebro Computador** corresponderam a também 7,41% dos Estudos Primários. Esta categoria engloba a construção de um protótipo de dispositivo para o controle de periféricos domésticos através de ondas cerebrais (SZU et al., 2013) e a idealização de um EEG portátil capaz de comunicar-se com *smartphones* (ALSHBATAT et al., 2014).

3.3.6 QP06: Quais são as principais contribuições dos estudos?

Esta seção tem por objetivo apresentar as principais contribuições identificadas em cada um dos Estudos Primários. Tais contribuições foram extraídas durante a leitura do texto completo (Seção 3.2) e visavam responder à esta Questão de Pesquisa. Os textos foram sintetizadas e dispostos por meio da listagem a seguir, a fim de possibilitar uma leitura estruturada, mesmo que a origem dos dados tenha sido de natureza qualitativa e com aspectos subjetivos.

- **S01** e **S18** Possibilitou a aquisição e o processamento de sinais de atividades neurais em tempo real, além da extração de *features* a fim de utilizar tais sinais como BCI (*Brain-Computer Interface*).
- **S02:** Trouxe uma visão de como a exibição das informações referentes ao código são interpretadas por programadores iniciantes e experientes.
- **S03:** O estudo indicou que, apesar de o tempo de fixação ser maior nos casos em que o código é apresentado em preto-e-branco, se comparado com o mesmo trecho com realce de sintaxe, essa diferença não é significativa no âmbito de compreensão do código.
- **S04:** Neste estudo vale salientar duas principais contribuições: (1) a metodologia para utilização de EEG na avaliação de carga cognitiva e (2) a evidência de que programadores mais experientes resolvem as tarefas de forma mais ágil, gerando uma menor carga cognitiva no mesmo espaço de tempo que os iniciantes.
- **S05:** Como principal contribuição deste estudo, os autores trouxeram um mapeamento da ativação cerebral durante o processo de depuração.
- **S06:** Os autores propuseram uma metodologia para relacionar a carga cognitiva do desenvolvedor com a baixa qualidade do código-fonte.

- **S07:** Este estudo foi importante por trazer uma revisão de como outros autores estão utilizando o *Eye-tracker* e o fNIRs como ferramentas para a mensuração da carga cognitiva/mental de desenvolvedores no quesito de compreensão de código-fonte.

- **S08:** Entre as contribuições deste estudo, duas são de grande valia para o desenvolvimento deste mapeamento: (1) elucida que a linguagem de programação e a linguagem natural são interpretadas em regiões diferentes do cérebro e (2) com o aumento da *expertise* do programador, a linguagem de programação começa a ser interpretada como uma linguagem natural pelo seu cérebro.

- **S09:** O trabalho trouxe como contribuição a possibilidade de se utilizar sinais neurofisiológicos para a avaliação da dificuldade em atividades de programação, além de trazer a possibilidade do desenvolvimento de ferramentas para a melhoria no processo de desenvolvimento de software.

- **S10:** Os autores trouxeram uma revisão de grande valia em torno de dados psicofisiológicos e seus dispositivos sensores. Ainda apresentaram as técnicas utilizadas para a análise dos dados utilizados e os algoritmos para predição de casos semelhantes.

- **S11:** Como resultado os autores apresentaram que, com o aumento da carga cognitiva, a produtividade degrada-se no decorrer do tempo.

- **S12:** Ainda que tenha sido realizado um experimento no âmbito do uso de linguagens de programação, o estudo apresentou como resultado os movimentos oculares dos participantes, sem traçar paralelos com a literatura a fim de encontrar padrões ou ainda explicações para os padrões identificados.

- **S13** e **S14:** Os resultados são úteis no entendimento do que acontece no cérebro do programador, quando observados determinados aspectos (lógica, fluxo de código) durante o processo de compreensão de linguagens de programação. Por meio do entendimento dos padrões de ativação cerebral, é possível criar métricas de avaliação das habilidades de programadores novatos e experientes. Ainda é possível determinar fatores que tornam o ensino de programação mais eficaz em termos de aumento da habilidade cognitivo.

- **S15:** Foi desenvolvida uma IDE sensível ao contexto de trabalho.

- **S16** e **S20:** Os autores expuseram o estado-da-arte em torno da utilização de *eye-tracking* na Engenharia de Software, além dos avanços da tecnologia no decorrer dos anos.

- **S17:** Desenvolveu diretrizes para estabelecer a fMRI como um instrumento padrão para observar a compreensão de código-fonte e outros processos relacionados.

- **S19:** O trabalho trouxe como contribuições a possibilidade de uso de EEGs de baixo custo como instrumentos para a medição da facilidade de uso de IDEs por parte dos

desenvolvedores, permitindo o aprimoramento da ferramenta, possibilitando ainda uma melhora na sua usabilidade.

- **S21:** Como contribuição do estudos, os autores evidenciaram que existe um padrão de ativação associado à memória de trabalho e ao processamento de linguagem natural. Ainda como contribuição, o trabalho trouxe uma metodologia para utilização de fMRI no contexto de compreensão de linguagens de programação.

- **S22:** Os autores identificaram que o uso de sugestões semânticas reduzem a ativação cerebral durante o processo de compreensão do código-fonte, além de reproduzir os estudos de outros autores e validar os seus resultados.

- **S23:** O trabalho em questão trouxe a viabilidade do desenvolvimento de um EEG doméstico, tanto para o uso como equipamento de monitoramento, como para BCI, dada a possibilidade de análise de dados em tempo real. Além disso elucidou os estados mentais vinculado aos espectros das ondas cerebrais.

- **S24:** O presente estudo contribuiu com a evidência de que, durante o processo de ensino de programação orientada à objetos, IDEs visuais como o BlueJ, reduzem a carga cognitiva, possibilitando inferir um processo de aprendizado mais facilitado. Ainda os autores fomentam estudos em torno do uso de IDEs e fNIRS.

- **S25:** Ainda que o trabalho tenha avaliado o comportamento das ondas Theta e Alpha, os autores não trouxeram uma perspectiva psicofisiológica dos processos cognitivos envolvidos durante as atividades.

- **S26:** Propôs uma métrica para avaliar complexidade cognitiva baseada em grafos de chamadas de métodos.

- **S27:** Os autores mostraram que as metodologias de depuração utilizada pelos desenvolvedores variam conforme as funcionalidades das IDEs, opondo-se às formas de depuração tradicionais.

Em adição aos resultados apresentados na forma de textos sintéticos, foram dispostas as métricas conforme as categorias as quais as respectivas contribuições foram enquadradas, conforme apresentado na Tabela 8.

Denota-se um maior número (37,04%) de estudos que contribuíram de forma significativa fornecendo algum tipo de **Métrica** acerca da utilização dos dados psicofisiológicos aplicados aos atributos elencados na QP03 e propósitos da QP05. Por conseguinte, observa-se que **Método** e **Modelo** apresentaram as mesmas frequências (22,22% cada), trazendo ambas contribuições que pudessem ser aplicados para a definição de um modelo de arquitetura ou método experimental. A categoria **Processo** trouxe contribuições (11,11%) para a definição de processos

Tabela 8: Distribuição dos estudos por tipo de Contribuições (QP06)

Contribuição	Trabalhos	Percentual	Estudos Primários
Métrica	10	37,04%	S03, S04, S05, S10, S12, S13, S14, S21, S22, S26
Ferramenta	2	7,41%	S09, S15
Modelo	6	22,22%	S01, S16, S18, S19, S20, S23
Método	6	22,22%	S02, S06, S07, S08, S24, S25
Processo	3	11,11%	S11, S17, S27

Fonte: Elaborado pelo autor.

que pudessem ser adotados no que tange coleta, processamento e análise de dados psicofisiológicos. Por último, apenas 2 estudos (7,41%) apresentaram alguma ferramenta que possuísse aplicabilidade no contexto deste trabalho.

É válido ressaltar que nenhum dos estudos analisados foi enquadrado na categoria *Integração*, já que estes não apresentaram mecanismos capazes de suportarem a integração dos dados coletados com os ambientes integrados de desenvolvimento ou outras ferramentas utilizadas.

3.3.7 QP07: Quais as naturezas dos estudos analisados?

Nesta Questão de Pesquisa buscou-se trazer um panorama acerca da distribuição das naturezas de cada estudo selecionado, utilizando-se os critérios definidos na Seção 3.1.4. Os resultados obtidos pelo processo de condução do mapeamento sistemático estão organizados e dispostos na Tabela 9.

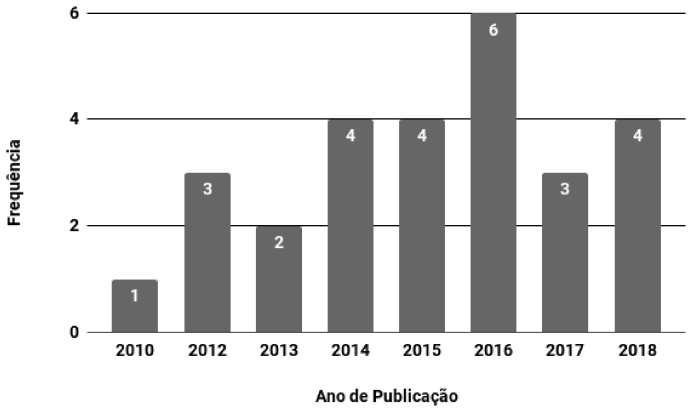
Tabela 9: Distribuição dos estudos por Natureza (QP07)

Natureza	Trabalhos	Percentual	Estudos Primários
Experimento Controlado	15	55,56%	S02, S03, S04, S05, S06, S08, S09, S10, S11, S12, S13, S14, S21, S22, S25
Proposta de Solução	7	25,93%	S07, S15, S18, S23, S24, S26
Pesquisa Avaliação	1	3,70%	S19
Artigo de Opinião	4	14,81%	S16, S17, S20, S27

Fonte: Elaborado pelo autor.

Observa-se que predominantemente os estudos selecionados (55,56%) são de natureza **Experimento Controlado**, ou seja, estudos em que os autores realizaram algum tipo de experimento de forma a validar uma técnica já existente ou ainda, corroborar algum conceito encontrado na literatura. Em seguida estão os estudos de natureza **Proposta de Solução**, representando 25,93% do montante total. Nesta natureza de estudos, estão àqueles em que o principal objetivo foi o de apresentar uma solução para um problema já existente ou uma versão melhorada de uma solução já apresentada, por meio de uma prova de conceito. Os **Artigos de Opinião** compuseram 14,81%, trazendo trabalhos que expressam a opinião dos autores acerca

Figura 14: Distribuição dos estudos pelos anos de publicação



Fonte: Elaborado pelo autor.

de determinado tópico, instigando a discussão e abrindo oportunidade para novos trabalhos. Por fim, 3,70% dos trabalhos foram de natureza **Pesquisa Avaliação**, onde buscou-se avaliar uma ferramenta, metodologia ou ideia apresentada por outros autores.

3.3.8 QP08: Em quais veículos de publicação os estudos selecionados foram divulgados?

Esta Questão de Pesquisa objetivava quantificar os veículos nos quais os Estudos Primários foram publicados. Os dados referente às respostas adquiridas mediante o desenvolvimento deste estudo estão dispostos na Tabela 10.

Tabela 10: Distribuição dos estudos por veículos de publicação (QP08)

Veículo	Trabalhos	Percentual	Estudos Primários
Conferência	16	59,26%	S03, S04, S06, S07, S08, S09, S10, S11, S12, S13, S14, S15, S17, S21, S25, S26
Periódico	6	22,22%	S01, S02, S16, S20, S23, S24
Oficina	2	7,41%	S18, S19
Simpósio	3	11,11%	S05, S22, S27

Fonte: Elaborado pelo autor.

Constata-se que em sua maioria (59,26%), os Estudos Primários foram publicados em conferências da área, indicando uma maior incipiência no que tange a temática dos trabalhos. Por conseguinte, estão os trabalhos que foram veiculados em periódicos (22,22%), caracterizando assuntos consolidados na comunidade científica. Por último são apresentados os trabalhos que foram publicados em eventos menores, tais como simpósios (11,11%) e oficinas (7,41%).

Em adição aos resultados apresentados acercado dos **Veículos de Publicação**, realizou-se também o levantamento da distribuição de publicação dos estudos no decorrer dos anos, conforme quantificado por meio da Figura 14.

É possível constatar que, no decorrer da última década – período avaliado no desenvolvimento deste mapeamento sistemático – houve um aumento no número de publicações referente ao uso de dados psicofisiológicos aplicados no contexto da Engenharia de Software.

3.3.9 Análise Comparativa

Foram definidos cinco Critérios de Comparação (CC) para auxiliar no processo de identificação das similaridades e diferenças entre o trabalho proposto e os artigos selecionados. Esta comparação é crucial para tornar o processo de identificação de oportunidades de pesquisa utilizando critérios objetivos, ao invés de subjetivos. Os critérios são descritos a seguir:

- **Mapeamento Sistemático (CC01):** estudos que desenvolveram um mapeamento sistemático da literatura visando adquirir uma visão do atual estado da arte.
- **Integração de Dados (CC02):** trabalhos que pesquisaram como os dados psicofisiológicos estão sendo suportados pelos ambientes integrados de desenvolvimento.
- **Métricas Psicofisiológicas (CC03):** nesta categorias estão os estudos que pesquisaram em torno de métricas biológicas que estivessem ligadas a processos cognitivos.
- **Aplicações na Engenharia de Software (CC04):** estudos que buscaram avaliar como dados psicofisiológicos são aplicados na Engenharia de Software.
- **Ambientes Integrados de Desenvolvimento (CC05):** pesquisas relacionadas a ambientes integrados de desenvolvimento, tanto em torno do seu uso como na sua melhoria.

Tabela 11: Análise comparativa dos Trabalhos Relacionados selecionados

Trabalho Relacionado	CC1	CC2	CC3	CC4	CC5
Trabalho proposto	●	●	●	●	●
FRITZ et al. (2014)	○	○	●	◐	○
FRITZ; MULLER (2016)	○	○	●	●	○
FUCCI et al. (2019)	○	○	●	●	○
GONCALES et al. (2019)	●	○	●	●	○
GUI et al. (2019)	○	○	◐	○	○
RADEVSKI; HATA; MATSUMOTO (2015)	○	◐	◐	●	○
ROSTAMI et al. (2015)	○	◐	◐	●	●

● Atende Completamente ◐ Atende Parcialmente ○ Não Atende

Fonte: Elaborado pelo autor.

3.4 Oportunidades de Pesquisa

Esta seção discute as principais oportunidades de pesquisa que podem ser exploradas para o desenvolvimento de estudos futuros. Para isto, foi utilizado o gráfico de bolhas ilustrado

na Figura 15, o qual combina as questões de pesquisa QP03 (Atributo), QP05 (Propósito) e QP06 (Contribuição). Mediante a síntese das questões apresentadas, somadas aos resultados discutidos na Seção 3.3, chegou-se a duas principais oportunidades, as quais serão discutidas abaixo.

3.4.1 Carência de Suporte à Integração de Dados Psicofisiológicos

Analisando-se os dados sintéticos apontados pela Figura 15, é possível observar-se na faceta da contribuição (eixo horizontal, à esquerda) que não foram identificados estudos que propusessem ferramentas com o propósito de melhoria na produtividade, análise de carga cognitiva, tampouco que atuassem como *interface* cérebro-computador. Quando esta análise é combinada aos dados apresentados na Figura 13 (Seção 3.3), denota-se uma carência por ferramentas que suportem a integração de dados psicofisiológicos, não obstante, com propósitos alinhados à melhoria na produtividade, medição e uso de carga cognitiva, ou ainda, que possam realizar a comunicação entre cérebro e computador.

Ademais, estudos que realizaram experimentos utilizando ambientes integrados de desenvolvimento e dados psicofisiológico mostraram que não existem mecanismos nativos (ROSTAMI et al., 2015; SHARIF et al., 2016) de suporte a tais tipos de dados por parte das IDEs, tampouco através de extensões que adicionem tal funcionalidade. Neste aspecto, os dados eram coletados por ferramentas proprietárias que se comunicavam com os dispositivos sensores, para posteriormente serem pré-processados e analisados.

Baseado nas constatações discutidas, o capítulo 4 foi criado visando contemplar a oportunidade de desenvolvimento de uma ferramenta capaz integrar e utilizar os dados psicofisiológicos oriundos dos desenvolvedores de forma a contemplar os propósitos supracitados.

3.4.2 Conhecimento empírico sobre o impacto de se exibir os dados psicofisiológicos nas IDES

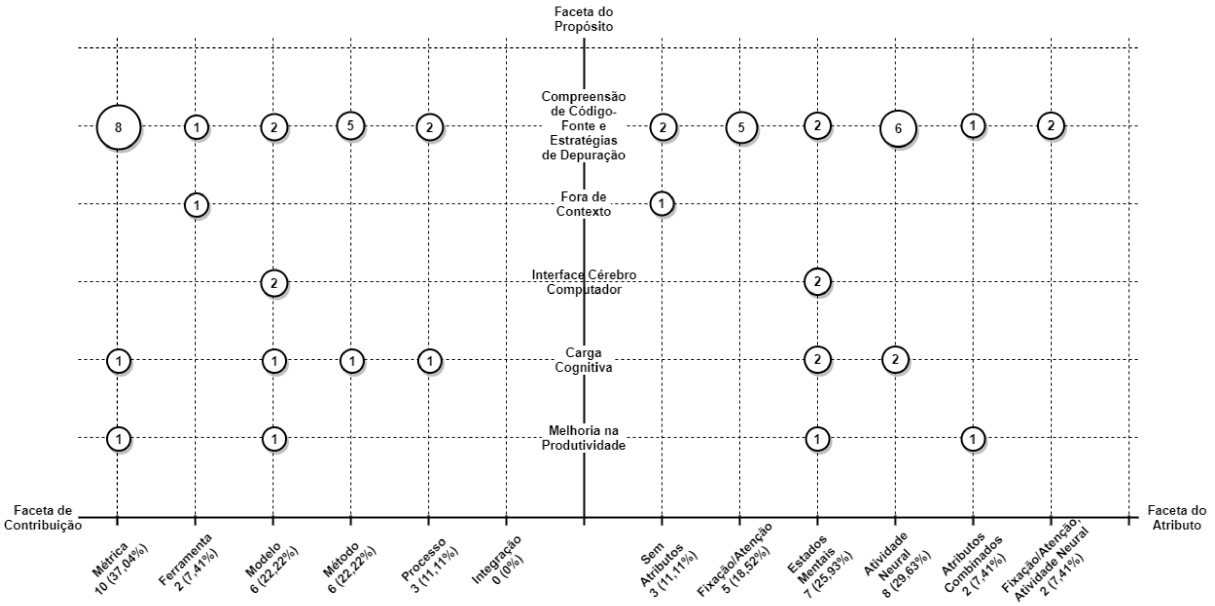
Ainda que a temática tenha sido objeto de estudo com crescente interesse nos últimos anos (GONCALES et al., 2019; VIEIRA; FARIAS, 2020a), a utilização de dados psicofisiológicos na engenharia de software tem se mostrado limitada à compreensão dos processos psicofisiológicos apresentados pelos desenvolvedores quando sujeitos a diversas condições. Tal observação poder ser corroborada analisando-se a faceta dos atributos (eixo horizontal, à direita) na Figura 15. Denota-se que apenas dois estudos utilizam Estados Mentais e Atributos Combinados visando melhorias na produtividade.

Ainda que a maioria dos estudos concentre-se no propósito de Compreensão de Código-fonte e Estratégias de Depuração, conforme discutido na Seção 3.3.6, os estudos analisados mostraram pouca aplicabilidade dos resultados obtidos como um fator de melhoria no processo de desenvolvimento de software. Tal constatação mostra-se evidente quando analisadas as com-

parações presentes na Tabela 11.

Assim como na oportunidade de pesquisa anterior, estudos que recorreram a IDEs em seus experimentos não tiraram proveito da utilização de tais dados nas próprias IDEs a fim de avaliar o impacto de se apresentar tais dados. Sendo assim, no Capítulo 5 desenvolveu-se um experimento controlado abrangendo a oportunidade de avaliação do impacto da exibição de dados psicofisiológicos dentro de IDEs, e qual a sua contribuição para o processo de desenvolvimento de software.

Figura 15: Combinação das QP03, QP05 e QP06



Fonte: Elaborado pelo autor.

3.5 Ameaças à Validade e Contramedidas

No que se refere ao desenvolvimento de Mapeamentos Sistemáticos, mesmo que estes possuam uma metodologia bem definida e passível de reprodução, as ameaças à validade do estudo são inerentes ao mesmo (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). Neste quesito, foram enumeradas três naturezas de ameaças: *construção*, *condução* e *conclusão*.

A primeira natureza de ameaças se refere às definições presentes na metodologia, em especial às *strings* de busca adaptadas para cada base de dados, as quais podem não representar uma retorno de mesmo valor para cada uma das bases. De modo a mitigar esta ameaça, foram construídas *strings* de busca individuais utilizando as peculiaridades de cada motor de busca.

A natureza subsequente de ameaças implica na condução do processo de filtragem, seleção e classificação dos estudos, o qual pode ter adotado uma classificação muito genérica ou, em contrapartida, muito específica, comprometendo a formação de grupos de estudos. Neste aspecto buscou-se realizar uma classificação iterativa, ou seja, no decorrer na classificação dos

estudos, novas categorias foram adicionadas, em detrimento à exclusão de outras, demandando novos ciclos de classificação de estudos já classificados previamente.

Por último, a natureza de conclusão traz consigo vieses dos autores aplicados aos resultados, podendo gerar inclinações à conclusões convenientes. Para tanto, buscou-se utilizar o embasamento teórico de estudos similares a fim de corroborar com os resultados, reforçando assim as conclusões obtidas.

4 COGNIDE

Neste capítulo será apresentada a ferramenta CognIDE. O desenvolvimento de desta visa contemplar um dos objetivos de pesquisa definidos neste trabalho (Seção 1.3) em concordância com as oportunidades de pesquisa identificadas por intermédio da análise de trabalhos relacionados à temática (Seção 3.4).

Para tanto, este capítulo organiza-se em três seções da seguinte forma: a Seção 4.1 busca apresentar uma visão geral da ferramenta e da abordagem adotada por esta. Em seguida, na Seção 4.2, são definidos os aspectos relacionados a arquitetura utilizada pelo CognIDE e, por fim, a Seção 4.3 descreve os detalhes da implementação da ferramenta.

4.1 Visão Geral da Ferramenta

A Figura 16 busca ilustrar a visão geral acerca do processo de funcionamento adotado na criação do CognIDE. Originalmente idealizado como uma extensão e proposta de arquitetura para o editor Visual Studio Code[1]. A escolha de tal IDE por parte do autor se deu por conveniência, levando-se em considerações dois pontos principais, sendo eles: 1) o suporte desta às diversas linguagens de programação existentes no mercado e 2) a extensibilidade de suas funções e a criação de novos recursos mediante o desenvolvimento de extensões em TypeScript.

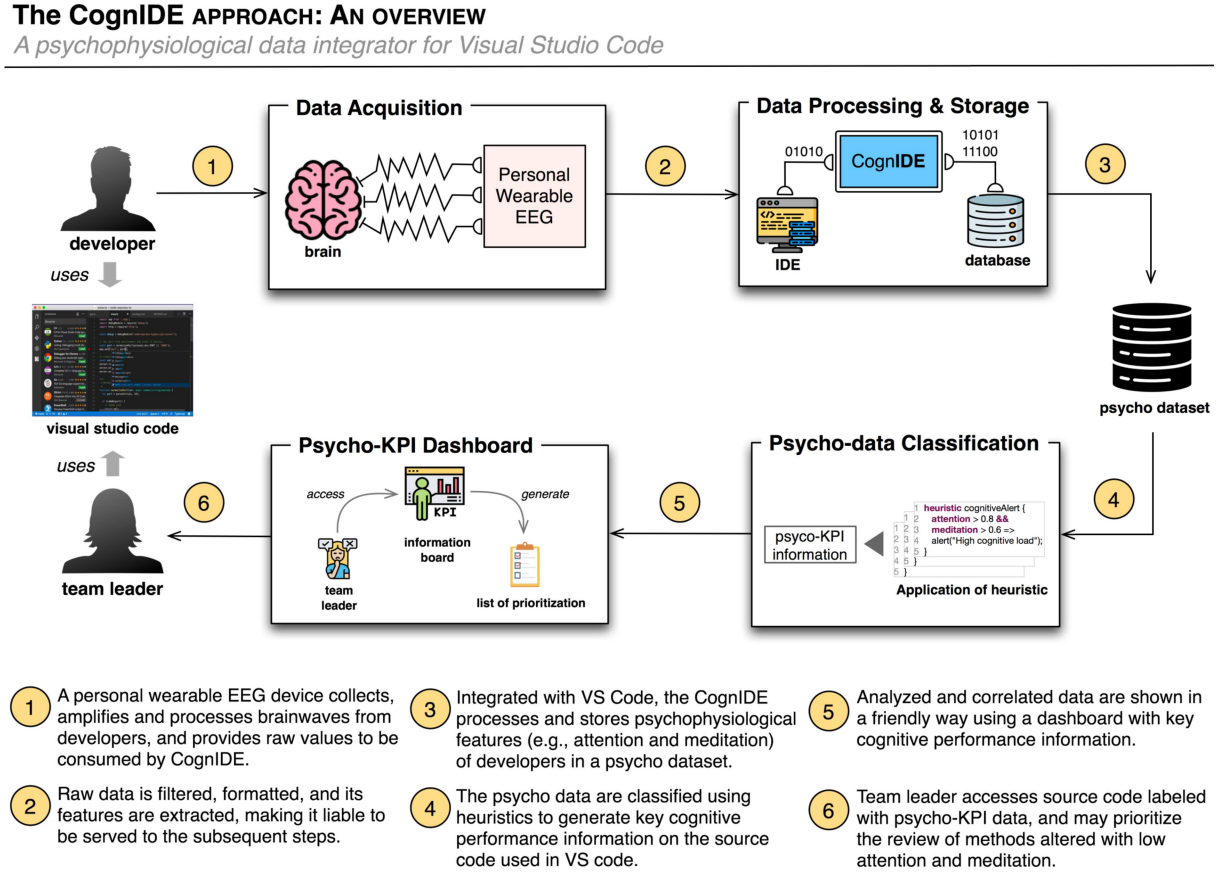
O CognIDE busca possibilitar a interação com dispositivos coletores de dados psicofisiológicos, tais EEGs, EDAs e pulseiras inteligentes de baixo custo. Tal abordagem foi idealizada com o propósito de consumir tais dados e atribuir-lhes informações contextualizadas geradas pelas IDEs durante o processo de desenvolvimento de software.

Os dados processados pelo CognIDE, além de serem apresentados no editor da IDE como um recurso contextualizado ao código-fonte que esta sendo produzido, podem ser armazenados externamente em bancos de dados para posteriores análises. Ademais, uma vez armazenados, os dados podem ser manipulados e visualizados através de painéis. Tais funcionalidades visam facilitar a integração de dados provenientes de experimentos envolvendo dados psicofisiológicos e ambientes integrados de desenvolvimento. Para isto, o CognIDE utiliza um fluxo de operação para a manipulação e beneficiação dos dados, o qual foi ilustrado pela Figura 16 e cujas etapas são descritas a seguir:

- **Etapa 1: aquisição.** Esta etapa visa a coleta dos sinais elétricos produzidos pela atividade neural do desenvolvedor durante uma tarefa de compreensão de código-fonte ou depuração de defeitos. Para isto, dispositivos EEG vestíveis de uso pessoal, como NeuroSky Mindwave Mobile 2[2], foram utilizados para a aquisição dos valores puros oriundos da atividade neural.

[1] https://code.visualstudio.com/
[2] https://store.neurosky.com/pages/mindwave

Figura 16: Uma visão geral do modelo de operação do CognIDE.



Fonte: Elaborado pelo autor.

- **Etapa 2: processamento.** Os dados puros oriundos da fase de aquisição são formatados e filtrados, objetivando a remoção de ruídos e impurezas (artefatos), e as suas *features* são extraídas por meio de algoritmos específicos, servindo assim de insumo para a próxima etapa do processo.

- **Etapa 3: armazenagem.** Neste ponto, os dados puros coletados da atividade elétrica neural já foram convertidos em indicadores contextualizados de atividade neural, tais como atenção, meditação, sonolência, etc., recorrendo ao conjunto de bibliotecas e APIs disponibilizadas pelo dispositivo EEG adotado. Estes indicadores são formatados e enviados para o seu armazenamento em um mecanismo de banco de dados de séries temporais.

- **Etapa 4: classificação.** Antes do armazenamento propriamente dito dos conjuntos de indicadores coletados de cada desenvolvedor observado, estão são previamente rotulados com uma série de metadados oriundos da IDE que está sendo utilizada, tais como a linha de código e o arquivo que estavam sendo editados no exato momento em que a métrica foi coletada. Sendo assim o CognIDE não é só responsável por coletar e processar os dados oriundos da atividade neural do desenvolvedor, mas também contextualizar métricas com os respectivos eventos relacionados a elas.

- **Etapa 5: criação dos painéis de visualização de indicadores.** Em posse das informações contextuais devidamente persistidas no banco de dados, o próximo passo seria a apresentação destes de forma amigável para o usuário. Neste aspecto, o CognIDE possibilita a criação de conjuntos de painéis de visualização (*dashboards*) dos indicadores coletados, visando apresentar informações relevantes para o responsável por realizar a análise.

- **Etapa 6: visualização.** Esta etapa envolve a renderização, visualização e análise dos painéis de visualização criados pelo usuário do CognIDE. Por meio destes, o usuário poderá criar visualizações personalizadas de métricas e suas relações, além de configurar alerta quando estas atingirem valores pré-determinados, possibilitando uma análise próxima a tempo-real.

4.2 Arquitetura

A Figura 17 ilustra a arquitetura baseada em componentes adotada para suportar a abordagem proposta pelo CognIDE. Juntos, os componentes são responsáveis por implementar as etapas descritas no processo. Para uma melhor compreensão da arquitetura, cada componente é descrito de um ponto de vista abstrato, visando descrever o seu comportamento geral em detrimento da tecnologia utilizada em sua implementação, como mostrado a seguir:

- **Connector:** este componente recebe os dados binários oriundos do dispositivo adotado na coleta de dados psicofisiológicos, utilizando comunicação de baixo nível, como, por exemplo, a comunicação serial (SSP) sobre Bluetooth, e aplicando o protocolo de comunicação específico adotado pelo EEG. Ademais, este componente realiza a transformação dos dados e os disponibiliza em um formato de alto-nível passível de ser consumido por outras aplicações, tais como JSON e XML.

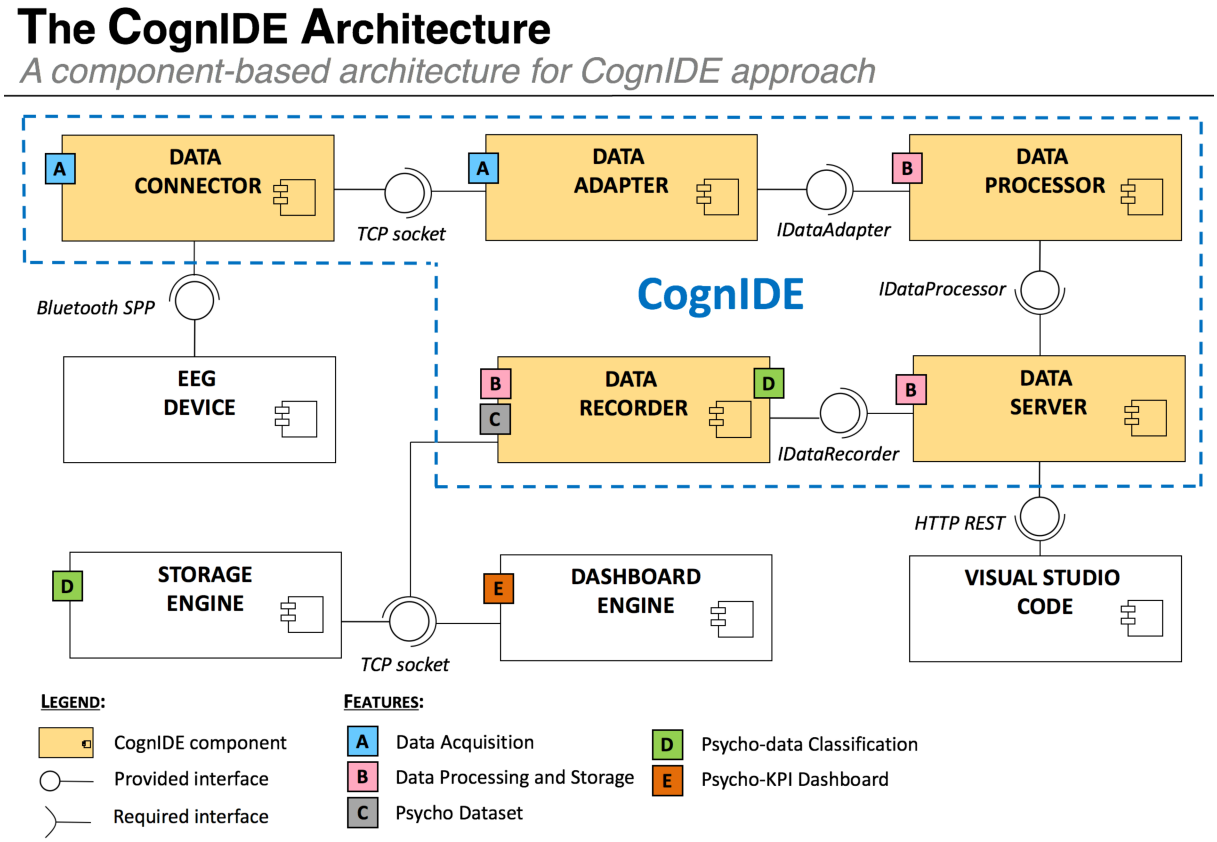
- **Adapter:** responsável por prover uma interação agnóstica de formato entre o Connector e o Processor, este componente visa reduzir o acoplamento arquitetural caso o Connector seja substituído, permitindo que este seja intercambiável por meio de configurações, conforme a necessidade do dispositivo que será utilizado na coleta de dados.

- **Processor:** aplica uma sequência predeterminada de cálculos sobre os valores dos dados puros disponibilizados pelo Connector, de forma a extrair as suas *features* e transformá-los em informações passíveis de serem analisadas.

- **Recorder:** tem como principal função, abstrair a camada de persistência de dados, tornando-a disponível para as operações de gravação e leitura dos dados previamente processados.

- **Server:** atua como o componente principal da arquitetura, recebendo os dados do Connector, processando-os utilizando o Processor e os enviando para serem consumidos

Figura 17: Uma visão geral da arquitetura adotada baseada em componentes.



Fonte: Elaborado pelo autor.

e visualizados pela IDE, além relacioná-los aos trechos de código-fonte produzidos e persistindo-os no sistema de armazenamento por meio do componente Recorder.

4.3 Implementação

Definidos o comportamento e a arquitetura adotada, iniciou-se a implementação do CognIDE. O dispositivo sensor escolhido para a coleta de dados do desenvolvedor foi o EEG NeuroSky Mindwave Mobile 2[3] (Figura 18). Este dispositivo foi escolhido devido ao seu baixo custo, quando comparado a outras soluções de mercado, portabilidade e um SDK gratuito e bem documentado[4], o qual traz uma API (*application interface*) que pode ser utilizada no desenvolvimento de aplicações para a interação cérebro-computador (BCI). Além disso, a precisão e aplicabilidade deste dispositivo foi validada em estudos recentes. Este mostrou-se como uma alternativa viável de EEG de baixo-custo e portátil, podendo ser aplicados em estudos emergente e validações de hipóteses, desde que não sejam demandas análises profundas (MORSHAD; MAZUMDER; AHMED, 2020).

Para o Data Connector, não houve a necessidade de implementação, uma vez que o au-

[3]http://neurosky.com/biosensors/eeg-sensor/biosensors/
[4]http://developer.neurosky.com/

Figura 18: EEG Headset MindWave Mobile 2.

Fonte: Adaptado de https://store.neurosky.com/pages/mindwave.

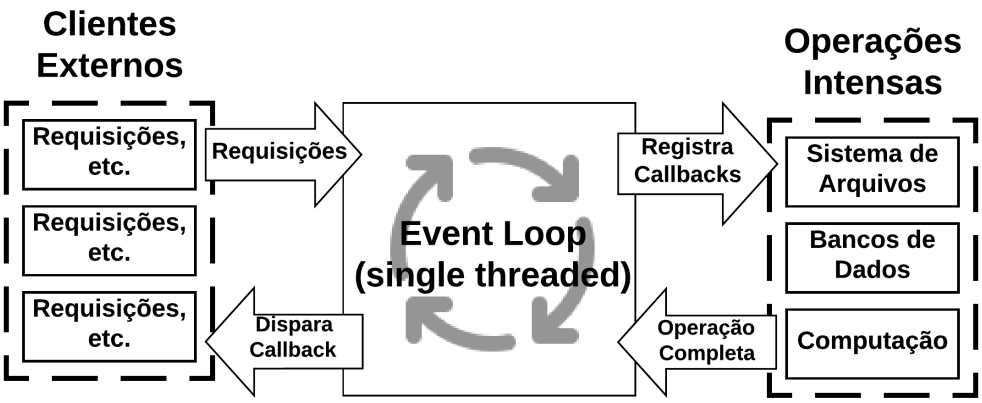
tor utilizou o ThinkGear Connector[5], uma opção gratuita disponível no mercado e que suporta dispositívos MindWave Mobile, uma linha de EEGs móveis de baixo-custo produzidos pela NeuroSky. O NeuroSky Connector comunica-se com os dispositivos EEG utilizando-se a comunicação serial sobre Bluetooth, e provê os resultados das coletas mediante a utilização de dados codificados no formato JSON e disponibilizados através de soquetes TCP.

A implementação do Data Server (Figura 20), nomeado de CognIDE Server, deu-se pela utilização do servidor de aplicação Node.js. A escolha pelo Node.js foi baseada na capacidade deste de manipular requisições sem bloquear a sua *thread* principal (chamada de *event loop*[6]), conforme ilustrado na Figura 19, além da sua capacidade de operar de forma orientada a eventos. Todas as transações realizadas entre a IDE e o CognIDE Server se dão por meio de trocas de mensagens JSON-RPC sobre o protocolo HTTP, gerenciado pelo Express[7], um *framework web* implementado em JavaScript para tratamento de requisições HTTP disponível para Node.js.

De forma a armazenar tanto os dados psicofisiológicos coletados pelo EEG, como os dados contextuais oriundos dos eventos da IDE, optou-se pela utilização do InfluxDB[8], um mecanismo de código aberto para gerenciamento banco de dados de séries temporais (TSDB). Esta classe de gerenciadores de bancos de dados caracteriza-se pelas altas taxas de operações de inserção de dados, ademais, facilita a sua consulta para análises de variações de valores no decorrer do tempo.

[5]https://store.neurosky.com/products/thinkgear-connector
[6]https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/
[7]https://expressjs.com/
[8]https://www.influxdata.com/products/influxdb-overview/

Figura 19: Visão geral da ordem de operações do *event loop*.



Fonte: https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/.

Os dados armazenados no InfluxDB foram consumidos pelo mecanismo de painéis de visualização Chronograf[9], o qual é responsável pela apresentação de forma intuitiva dos dados processados (Figura 21). Tal ferramenta foi selecionada devido à sua integração nativa com o InfluxDB, uma vez que ambas as soluções são desenvolvidas e mantidas pela mesma, a Influx Data[10]. Na Figura 21 é apresentado um exemplo de painel de visualização utilizando-se o Chronograf. Na parte superior é exibido um gráfico de área indicando as variações das métricas de Atenção e Meditação em função do tempo, conforme estas eram coletadas durante a atividade do desenvolvedor. Na parte inferior, os medidores apresentam o valor médio da Atenção (esquerda) e Meditação (direita) dentro de um intervalo de tempo pré-definido.

Além da apresentação dos dados processados mediante o uso de painéis de visualização do Chronograf, implementou-se também o CognIDE Extension, uma extensão desenvolvida para o retorno visual de dados simultaneamente à apresentação do código-fonte na ferramenta de edição. Para o seu desenvolvimento, utilizou-se como base o editor de código-fonte da Microsoft, Visual Studio Code[11], o qual provê uma API de alto-nível para o desenvolvimento de extensões (ou *plug-ins*) de forma a possibilitar a adição de novas funcionalidades ou alterações das já presentes. Implementado utilizando-se o TypeScript, um *superset* de funcionalidades e tipagem estática para a linguagem de programação Javascript, o CognIDE Extension aproveitou-se da API de apresentação de dados sensíveis ao contexto, CodeLens[12], a qual é responsável por reagir a eventos provenientes da interação do usuário no editor de código-fonte, além de permitir a apresentação de conjuntos de informações definidas pelo usuário próximo ao espaço do editor. Conforme ilustrado pela Figura 22, as métricas de Atenção e Meditação são apresentadas na parte superior do respectivo trecho de código-fonte que estava sendo editado no momento em que as métricas foram coletadas. Na figura são exibidos os mesmos indicadores para todas as

[9]https://www.influxdata.com/time-series-platform/chronograf/
[10]https://www.influxdata.com/
[11]https://code.visualstudio.com/
[12]https://code.visualstudio.com/blogs/2017/02/12/code-lens-roundup

Figura 20: Trecho de código mostrando a implementaçao do Data Server.

```
var net = require('net')
var express = require("express");
var app = express();
var client = new net.Socket();

import { Measurement } from "./models/Measurement";···

var adapter: DeviceAdapter = new MockedAdapter();
var recorder: IRecorder = new InfluxDBRecorder();


// ================================== COGNIDE SERVER ================================== //

app.listen(3000, () => {
    console.log("Server running on port 3000");
});

// ================================  CLIENT REQUESTS  ================================ //

// http://localhost:3000/metrics?clientId=123&artifactName=helloWorld.js&lineNumber=24
app.get("/metrics", (req: any, res: any, next: any) => {

    var request: MetricsRequest = req.query;
    console.info(`Received request from CognIDE Extension: ${JSON.stringify(request)}`);

    var measurement: Measurement = adapter.getMeasurement();
    console.debug(`Retrieved from adapter: ${measurement}`);

    var metric: IMetricEntity = new MetricEntity(
        request.clientId,
        request.artifactName,
        request.lineNumber,
        measurement.attention,
        measurement.meditation
    );

    recorder.Save(metric);
    res.send(JSON.stringify(measurement));

});

// ============================  CLIENT DISCONNECTION  ============================ //

client.on("close", () => {
    console.log(`${Date.now()} - Disconnected`);
});
```

Fonte: Elaborado pelo autor.

linhas do código. Tal apresentação deu-se devido à forma como a granularidade dos eventos foram configurados, neste caso, para cada linha editada.

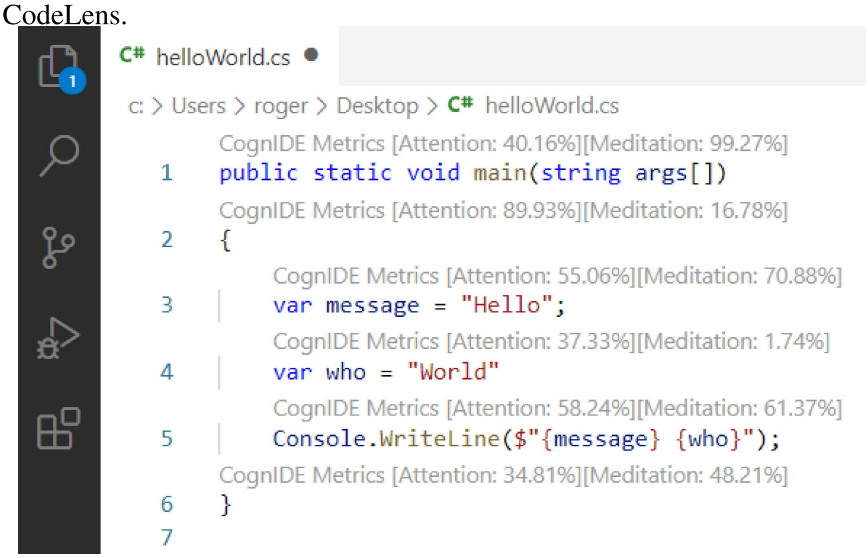
Conforme o desenvolvedor digita os trechos de código-fonte, o módulo dos CodeLens fica responsável por capturar os eventos de digitação, não sendo limitado somente a este tipo, envia-os para Data Server junto aos metadados relativos ao contexto de geração do evento, tais como nome do arquivo sendo editado, linha posicionada, palavra selecionada, etc., utilizando JSON-RPC. O Data Server, por sua vez, comunica-se com o EEG via o Data Connector. Na sequência, o dispositivo devolve as medições já previamente filtradas, os dados psicofisiológicos coletados são então processados, rotulados e combinados com os metadados provenientes do CognIDE Extension, para então serem enviados ao mecanismo de banco de dados e enfim armazenados. Neste momento, os dados persistidos são consumidos de duas formas: (1) o Data Server recebe o retorno do processamento de persistência e responde à requisição do CognIDE Extension

Figura 21: Exemplo de um painel criado no Chronograf para visualização de dados psicofisiológicos.

Fonte: Elaborado pelo autor.

com as respectivas métricas coletadas; e (2) mediante a integração nativa, os agendadores do Chronograf ficam responsáveis por consumir os dados armazenados e atualizar os painéis de visualização de acordo com suas configurações realizadas pelo usuário.

Figura 22: Trecho de código-fonte exibindo dados psicofisiológicas integrados utilizando-se o CognIDE Extension e o CodeLens.



Fonte: Elaborado pelo autor.

5 AVALIAÇÃO

Mediante as oportunidades de pesquisa identificadas na Seção 3.4 este capítulo visa apresentar a definição, desenho e planejamento do modelo de avaliação e apresentação dos resultados da ferramenta CognIDE. Ademais, será detalhada a condução do método de avaliação utilizado no experimento controlado desenvolvido neste estudo visando a produção de resultados empíricos. Tais resultados visam contemplar a carência de estudos empíricos que possam explicar as relações e aplicabilidade da utilização de métricas psicofisiológicas no processo de desenvolvimento de software.

Inicialmente, um experimento controlado foi desenvolvido de modo a avaliar e analisar os impactos de se apresentar a métrica psicofisiológica de Carga Cognitiva junto ao código-fonte produzido em ambientes integrados de desenvolvimento, em especial os seus efeitos sobre a percepção de Anomalias de Código e o Nível de Intenção de Refatoração apresentado pelo desenvolvedor.

Neste experimento, utilizou-se o conceito de integração de dados psicofisiológicos em ambientes integrados de desenvolvimento proposto pela ferramente CognIDE no Capítulo 4. Para a sua avaliação, utilizou-se como referência o método adotado por RICCA et al. (2010) em seu estudo sobre o impacto do uso de notações UML na compreensão dos desenvolvedores. Na Tabela 12, é apresentada a sumarização utilizada para a confecção do experimento controlado adotado neste estudo.

Tabela 12: Visão geral do experimento.

Objetivo	Verificar se a presença da métrica de Carga Cognitiva junto ao código fonte pode influenciar na percepção de Anomalias de Código e/ou na intenção de refatoração por parte do desenvolvedor.
Contexto	Análise estática de código-fonte mediante a presença ou ausência da métrica psicofisiológica de Carga Cognitiva.
Hipótese Nula	Não há influência do indicador de Carga Cognitiva, quando apresentado junto ao código fonte, tanto na percepção de anomalias como na intenção de refatoração do código proposto.
Variáveis Independentes	Indicador de Carga Cognitiva (N/A, Alto e Baixo) e Número de Anomalias (0, 1 e 2).
Variáveis Dependentes	Percepção de Anomalias de Código (Sim ou Não) e Nível de Intenção de Refatoração (1, 2, 3, 4 e 5).

Fonte: Elaborado pelo autor.

Este capítulo é composto por três seções principais que contemplam todas as etapas adotadas na confecção do experimento controlado. Inicialmente, a Seção 5.1 descreve o desenho experimental utilizado para o delineamento das etapas necessárias para a confecção do experimento controlado. Em seguida, a Seção 5.2 detalha como se deu a condução do experimento

controlado propriamente dito. A Seção 5.3 traz os resultados consolidados referente à estatística descritiva, assim como os destes das hipóteses avaliadas no decorrer do experimento. Por fim, na Seção 5.4, os resultados obtidos e apresentado serão discutidos à luz da literatura especializada.

5.1 Desenho Experimental

Esta seção endereça o processo adotado no experimento controlado, tal como objetivos, formulação de hipóteses, variáveis e cenários testados objetivando avaliar a aplicabilidade do CognIDE. Não obstante, pouco se sabe acerca do impacto de se apresentar métricas psicofisiológicas em IDEs, especialmente a Carga Cognitiva no processo de desenvolvimento de software.

Conforme descrito na Seção 2.2.1, altos níveis de Carga Cognitiva durante o desenvolvimento podem indicar dificuldades enfrentadas pelo desenvolvedor, tanto em fatores sintáticos, como lógicos (ANTONENKO et al., 2010; FAKHOURY et al., 2018; ROSTAMI et al., 2015). A ideia principal por trás da condução do experimento controlado seria a de avaliar como a presença do indicador psicofisiológico de Carga Cognitiva poderia influenciar em aspectos relacionados ao desenvolvimento de software, tais como Percepção de Anomalias e Nível de Intenção de Refatoração, quando observados durante a revisão de código.

O desenho experimental definido nesta seção para a avaliação do CognIDE foi dividido em quatro seções menores contendo o detalhamento do processo. A primeira delas, Seção 5.1.1, descreve os objetivos que visam serem alcançados com a condução do experimento. Na Seção 5.1.2, estão descritas as hipóteses conjecturadas acerca dos efeitos da presença do indicador de Carga Cognitiva junto ao código-fonte durante o processo de revisão. Em seguida, mediante a Seção 5.1.3, foram definidas as variáveis dependentes e independentes avaliadas utilizadas na validação das hipóteses. Por fim, a Seção 5.1.4 apresenta a descrição dos cenários de teste das hipóteses, assim como quais questões os compuseram e quais variáveis foram avaliadas.

5.1.1 Objetivos

Esta seção formaliza os objetivos deste experimento. Neste é explorado os efeitos de se apresentar o indicador de Carga Cognitiva em uma IDE sobre a Percepção de Anomalias e Nível de Intenção de Refatoração do ponto de vista do indivíduo que está revisando o código-fonte. Estes efeitos serão avaliados mediante diferentes cenários de testes compostos por combinações de anomalias de código e diferentes níveis de carga cognitiva apresentada. O propósito deste experimento é descrito por intermédio da convenção GQM (WOHLIN et al., 2012):

Analisar *a apresentação da métrica de Carga Cognitiva em IDEs*
com o propósito de *investigar os seus efeitos*
em relação à *Percepção de Anomalias de código e Nível de Intenção de Refatoração*
com a perspectiva de *um indivíduo que está revisando código-fonte*

no contexto de *identificar aplicabilidades para o indicador de Carga Cognitiva nas IDEs.*

5.1.2 Formulação das Hipóteses

Nesta seção, as hipóteses formuladas no desenvolvimento deste estudo são listadas. Cada hipótese, além da sua descrição, é seguida pelas conjecturas que estas produzem e que servirão de subsídio para a confecção do desenho experimental.

5.1.2.1 Primeira Hipótese: o impacto de apresentar o indicador de Carga Cognitiva na percepção de Anomalias de Código por parte do desenvolvedor

Nesta primeira hipótese, conjectura-se que a presença da métrica de Carga Cognitiva em ambientes integrados de desenvolvimento, quando esta assume seu valor como "Alto", pode levar o desenvolvedor a aumentar a sua atenção e, consequentemente, as chances deste perceber Anomalias de Código criadas durante a fase de desenvolvimento. Tais conjecturas surgiram mediante observações que relacionam os altos níveis de Carga Cognitiva, produzidos por certas características de código-fonte, tais como variáveis declaradas com nomenclaturas dúbias, código-fonte com baixa legibilidade, altos níveis de complexidade ciclomática, entre outros fatores.

Baseado nas observações discutidas no parágrafo anterior, é hipotetizado que, dado um alto nível de Carga Cognitiva apresentado durante o desenvolvimento de certos artefatos de software, seria possível conjecturar que o desenvolvedor apresentou dificuldade em seu desenvolvimento, assumindo-se assim que existiria uma possibilidade deste serem observadas anomalias no código-fonte produzido. Sendo assim, a primeira hipótese visa avaliar se, quando apresentada a métrica de Carga Cognitiva "Alta"junto ao código-fonte produzido contendo ou não anomalias, poderia influenciar na percepção destas por parte do desenvolvedor. Esta hipótese é sumarizado a seguir:

> **Hipótese Nula 1, H_{1-0}:** Não há significância estatística na percepção de Anomalias de Código devido à presença da métrica de Carga Cognitiva.
> **Hipótese alternativa 1, H_{1-1}:** É observada significância estatística na percepção de Anomalias de Código devido à presença da métrica de Carga Cognitiva.

Testando-se a primeira hipótese, é avaliado se há significância estatística na percepção de Anomalias de Código mediante a presença da métrica de Carga Cognitiva. Este teste de hipótese visa demonstrar se a métrica de Carga Cognitiva, quando exibida junto ao código-fonte, poderia auxiliar o desenvolvedor na identificação de possíveis problemas de código ocasionados pela presença de anomalias durante a fase de desenvolvimento

5.1.2.2 Segunda Hipótese: impacto de apresentar o indicador de Carga Cognitiva no Nível de Intenção de Refatoração por parte do desenvolvedor

Como mencionado anteriormente, anomalias no código-fonte podem levar o desenvolvedor a enfrentar dificuldades durante a implementação, as quais podem levar à produção de artefatos de software defeituosos. Nesta hipótese, portanto, os autores conjecturam que, mesmo que não exerça influência na percepção de Anomalias de Código por parte dos desenvolvedores, a presença do indicador psicofisiológico de Carga Cognitiva próximo ao trecho de código em que este foi observado, quando combinado a conhecimentos prévios e experiências adquiridas pelo desenvolvedor, pode exercer certa influência no nível de intenção de refatorar o código em questão. Sendo assim, hipotetiza-se que a presença da métrica de Carga Cognitiva pode influenciar na decisão do desenvolvedor em refatorar o código-fonte produzido. Tal influência, portanto, poderia auxiliar na identificação precoce de determinados artefatos defeituosos de software em tempo de desenvolvimento, evitando assim perdas de tempo durante a implementação e reduzindo o tempo total do ciclo de produção de software. A hipótese levantada e suas alternativas são descritas abaixo:

Hipótese Nula 2, H_{2-0}: Não há significância estatística no Nível de Intenção de Refatoração mediante a presença do indicador de Carga Cognitiva. **Hipótese alternativa 2, H_{2-1}:** É observada significância estatística no Nível de Intenção de Refatoração devido a presença do indicador de Carga Cognitiva.

Testando-se a segunda hipótese, tem-se como objetivo identificar se o indicador de Carga Cognitiva, quando exibido em ambientes integrados de desenvolvimento junto ao código fonte utilizado durante a coleta da métrica, poderia auxiliar o desenvolvedor na decisão quanto a priorização da refatoração do código fonte em questão.

5.1.3 Seleção das Variáveis

Esta seção discute as variáveis dependentes e as independentes envolvidas neste estudo. Esta pesquisa visa identificar se a métrica apresentada pela ferramenta CognIDE possui relevância no processo de desenvolvimento de software. Com isto em mente, irá verificar-se se uma métrica psicofisiológica que indique o nível de Carga Cognitiva do indivíduo durante a escrita de código, pode ou não influenciar na percepção de anomalias e/ou Nível de Intenção de Refatoração indicado pelo sujeito que revisa o código escrito. A Tabela 13 detalha as variáveis investigadas neste estudo com o objetivo de validar a relevância do CognIDE.

O experimento controlado desenvolvido é composto por duas variáveis dependentes: Percepção de Anomalias e Nível de Intenção de Refatoração. A primeira visa indicar se o indivíduo, quando executando o papel de revisor de código no contexto da pesquisa, percebeu ou não

Tabela 13: Relação das variáveis independentes e dependentes selecionadas a avaliação da relevância do CognIDE.

Natureza	Variável	Tipo	Valores
Independente	Anomalias de Código	Quantitativa Discreta	0, 1 e 2
	Carga Cognitiva	Qualitativa Nominal	N/A, Alta e Baixa
Dependente	Percepção de Anomalias	Qualitativa Nominal (Binária)	Sim e Não
	Nível de Intenção de Refatoração	Qualitativa Ordinal	1 - Certamente não recomendaria; 2 - Não recomendaria; 3 - Talvez recomendasse; 4 - Recomendaria; 5 - Certamente recomendaria

Fonte: Elaborado pelo autor.

a presença de uma anomalia no código-fonte analisado. A segunda indica, independentemente se o revisor percebeu ou não alguma anomalia, o nível com que este deseja enviar o código em questão para ser refatorado.

Como variáveis independentes, foram utilizadas as Anomalias de Código e a Carga Cognitiva. Numericamente, a variável Anomalias de Código informa a quantidade de anomalias presentes no trecho de código que é analisado, entretanto, este valor não fica visível para o revisor, sendo utilizado para fins de testes de hipóteses e controle. Por outro lado, a variável de Carga Cognitiva é apresentada ao revisor e exibe, textualmente, os níveis de Carga Cognitiva medidos pelo desenvolvedor no momento em que este escreveu o trecho de código que está sob revisão pelo participante.

5.1.4 Cenários

Como descrito na Seção 5.1.3, um experimento controlado foi executado de modo a avaliar a Percepção de Anomalias e o Nível de Intenção de Refatoração. A fim de garantir o isolamento das variáveis e impedir a interação entre elas, esta foram separadas em cenários de avaliação. Esta seção, portanto, tem como finalidade descrever como se deu a composição de tais cenários e quais variáveis os compuseram. A Tabela 14 resume os cenários de avaliação, as questões testadas, assim como as suas variáveis relacionadas.

Tabela 14: Resumo da composição dos cenários avaliados.

Cenário	Questões Testadas	Variáveis Testadas
C01	Q01, Q03 e Q04	Percepção de Anomalias e Nível de Intenção de Refatoração
C02	Q02, Q05 e Q06	
C03	Q03 .. Q08	

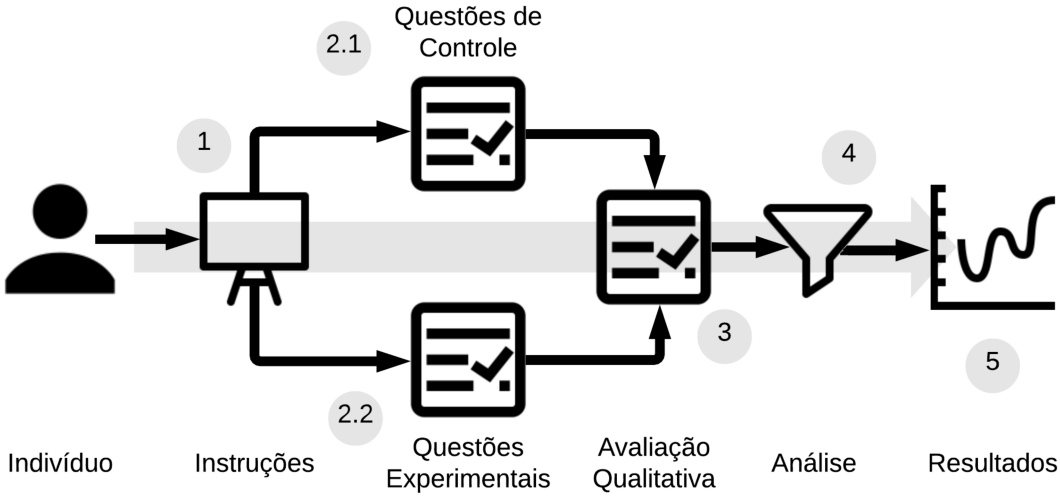
Fonte: Elaborado pelo autor.

O primeiro cenário (C01) apresentado na Tabela 14 é composto pelas questões Q01, Q03 e Q04, descritas mediante a Tabela 15 da Seção 5.2. Este cenário objetiva avaliar a influência da variável Carga Cognitiva em questões nas quais não há a presença de anomalias, garantindo assim que esta seja a única variável manipulada. Já no segundo cenário (C02), ocorre a presença de apenas uma anomalia de código em toda as questões e, assim como no primeiro cenário, a variável de Carga Cognitiva é a única que é manipulada entre as questões, sendo assim, este cenário visa avaliar se a Carga Cognitiva seria o único fator ou se a presença das Anomalias de Código também exerceria influência nas variáveis dependentes. O terceiro e último cenário (C03) foi desenhado para avaliar apenas as variáveis do grupo experimental. Neste cenário, a variável Anomalias de Código pode ser observada nenhuma, uma ou duas vezes entre as questões, assim como a variação dos valores apresentados pela Carga Cognitiva. Portanto, o terceiro cenário tem por finalidade avaliar se tanto o número de Anomalias de Código, como os valores assumidos pela variável Carga Cognitiva influenciam tanto na Percepção de Anomalias como no Nível de Intenção de Refatoração reportados pelo indivíduo revisor.

5.2 Processo Experimental

Nesta seção, são descritas as etapas adotadas na confecção do experimento controlado. Não obstante o desenvolvimento de experimentos controlados utilizando-se dados psicofisiológicos, assim como a utilização de ambientes integrados de desenvolvimento, não foi observado desenhos experimentais que contemplassem ambos aspectos. Neste estudo, portanto, foi utilizada uma versão adaptada do desenho experimental aplicado no estudo de RICCA et al. (2010). O modelo referido é ilustrado pela Figura 23.

Figura 23: Fluxo de execução do experimento controlado.



Fonte: Elaborado pelo autor.

A primeira etapa da fase experimental caracterizou-se pela introdução à temática e fornecimento das instruções. Inicialmente foi ao indivíduo uma breve conceituação acerca dos temas

de Anomalias de código e Carga Cognitiva, ambos ambientados no contexto da engenharia de software, podendo estes serem consultados no Apêndice C.

Em seguida, foram fornecidas as instruções necessárias para a condução do questionário, descrevendo o cenário hipotético em que o indivíduo estava inserido, qual o seu papel neste e as ações pelas quais, era responsável. Neste cenário, foi descrito o que era o CognIDE e qual a sua finalidade no contexto apresentado.

A segunda etapa da fase experimental caracterizou-se pela aplicação do questionário do experimento controlado. Este compôs-se por oito questões objetivas acerca de duas variáveis: Percepção de Anomalias e Nível de Intenção de Refatoração. Cada questão continha um trecho de código-fonte onde o indivíduo, sob o papel de revisor de código, deveria ler e responder às questões que seguiam, conforme ilustrado pela Figura 24.

Figura 24: Exemplo de questão apresentada ao indivíduo durante o experimento controlado.

Fonte: Elaborado pelo autor.

É importante ressaltar que, mesmo havendo distinção entre as questões do grupo de controle e do grupo experimental, um mesmo indivíduo acaba por responder às questões de ambos os grupos. Esta característica dá-se por se tratar de um experimento auto-pareado (ou auto-controlado), onde o indivíduo participa tanto do grupo controle, como do grupo experimental

(HOCHMAN et al., 2005). Além disso, como o estudo visa avaliar o impacto da manipulação das variáveis independentes, eventualmente chamadas de "tratamento", no decorrer das questões respondidas pela mesma unidade amostral, este também caracteriza-se como sendo um estudo de medidas repetidas (HOCHMAN et al., 2005). Mediante a Tabela 15, são descritas as questões aplicadas no questionário do experimento controlado, as variáveis independentes manipuladas, assim como o grupo a qual cada questão pertence.

Tabela 15: Composição e distribuição das questões utilizadas no questionário do experimento controlado.

Grupo	ID	Anomalias de Código	Indicador Carga Cognitiva
Controle	Q01	Nenhuma	Não se aplica
	Q02	Speculative Generality	Não se aplica
Experimental	Q03	Nenhuma	Alta
	Q04	Nenhuma	Baixa
	Q05	Speculative Generality	Alta
	Q06	Speculative Generality	Baixa
	Q07	Speculative Generality e God Method	Alta
	Q08	Speculative Generality e God Method	Baixa

Fonte: Elaborado pelo autor.

A última fase da experimentação deu-se pela aplicação de um questionário qualitativo. Este tinha por objetivo avaliar o experimento perante quatro dimensões: *Clareza*, *Percepção*, *Efeitos* e *Percepção de Uso*, segundo discriminado pela Tabela 16. Cada questão podia ser respondida segundo a escala de Likert indicando o grau de concordância do sujeito mediante a questão aplicada, indo de *1 - Discordo totalmente* à *5 - Concordo totalmente*.

Tabela 16: Relação das questões apresentadas no questionário pós-experimento.

Contexto	ID	Questão
Clareza	Q09	Tive tempo o suficiente para responder o questionário
	Q10	O treinamento me capacitou para executar o experimento.
	Q11	As instruções do treinamento foram claras.
	Q12	As questões apresentadas foram claras.
Percepção	Q13	Não tive dificuldade em compreender o código apresentado.
	Q14	Não tive dificuldade em compreender a métrica de Carga Cognitiva apresentada.
Efeitos	Q15	Compreendi o que são anomalias (Bad Smells) de código.
	Q16	Compreendi o que é Carga Cognitiva.
Percepção de Uso	Q17	A métrica de Carga Cognitiva auxiliou na percepção de Anomalias de Código.
	Q18	A métrica de Carga Cognitiva influenciou na recomendação da refatoração do código apresentado.

Fonte: Elaborado pelo autor.

A primeira dimensão avaliada, *Clareza* quantificava o grau de entendimento das instruções e

do texto das questões do ponto de vista do respondente. Em seguida foi avaliada a percepção do indivíduo em relação aos trechos de código apresentado e acerca da métrica de Carga Cognitiva por meio da *Percepção*. A dimensão dos *Efeitos* quantificava se os conceitos de Anomalias de Código e Carga Cognitiva foram assimilados pelo indivíduo após a execução do experimento. Por fim, a dimensão de *Percepção de Uso* buscava elucidar se, conforme a percepção do indivíduo, a métrica de Carga Cognitiva, presente em alguns trechos de código, exerceu alguma influência na sua decisão enquanto este respondia ao questionário.

5.2.1 Seleção dos Participantes

Este experimento controlado foi realizado com 61 voluntários estudantes e profissionais da área de Tecnologia da Informação, com enfoque em desenvolvimento de software, os quais foram recrutados por e-mail e indicação do professor orientador durante o programa de mestrado. Além do experimento controlado, dados sociodemográficos foram coletados para fins de delineamento da amostra, dados estes que foram anonimizados com consentimento dos participantes.

A Tabela 17 mostra a faixa etária dos participantes, os quais foram divididos por grupos, 42 indivíduos possuem entre 20 e 29 anos (68,85%), 16 entre 30 e 39 anos (26,23%), 2 entre 40 e 49 anos (3,28%) e 1 (1,64%) com menos de 20 anos. Observa-se que a maioria dos participantes possuía entre 20 e 29 anos, compondo assim 68,85% do total (n=61).

Tabela 17: Faixa etária dos participantes.

Faixa Etária	Quantidade	Percentual
de 20 a 29 anos	42	68.85%
de 30 a 39 anos	16	26.23%
de 40 a 49 anos	2	3.28%
menos de 20 anos	1	1.64%

Fonte: Elaborado pelo autor.

A Tabela 18 detalha o nível de escolaridade dos participantes, distribuindo-os em cursos profissionais (Técnico), superior (Completo e Incompleto), pós-graduação *lato sensu* (Especialização e/ou MBA) e pós-graduação *stricto sensu* (Mestrado e/ou Doutorado/Ph.D.). Observa-se que a maior parte dos participantes avaliados (95,98%) estavam cursando ou já haviam concluído curso superior. Do montante restante, 7 participantes (11,48%) possuíam algum tipo de curso de pós-graduação.

Dentre todos os participantes, por intermédio da Tabela 19, é possível notar-se que a maioria destes atua como desenvolvedor de software/programador, caracterizando 55,37% do total (n=61), onde, em relação ao montante total, 40,98% possuem até quatro anos de experiência na função. Do total de participantes, ressalta-se que 24,59% dos indivíduos possuem mais de oito anos de experiência nas respectivas funções, representando cerca de $1/4$ da quantidade total (n=61).

Tabela 18: Nível de escolaridade dos participantes.

Escolaridade	Quantidade	Percentual
Ensino Superior Incompleto	43	70.49%
Ensino Superior Completo	9	14.75%
Especialização/MBA	4	6.56%
Mestrado/Doutorado	3	4.92%
Ensino Técnico	2	3.28%

Fonte: Elaborado pelo autor.

Tabela 19: Ocupação e experiência dos participantes.

Ocupação	Tempo	Quantidade	Percentual
Analista de Sistemas	até 2 anos	3	4,92%
	de 3 a 4 anos	1	1,64%
	de 5 a 6 anos	2	3,27%
Analista de Suporte	de 3 a 4 anos	3	4,92%
Analista de qualidade de sistemas	acima de 8 anos	1	1,64%
Arquiteto de Software	acima de 8 anos	4	6,55%
	de 3 a 4 anos	1	1,64%
	de 7 a 8 anos	1	1,64%
Coordenador de Suporte	de 3 a 4 anos	1	1,64%
Desempregado	de 5 a 6 anos	1	1,64%
Desenvolvedor	acima de 8 anos	5	8,19%
	até 2 anos	13	21,31%
	de 3 a 4 anos	12	19,67%
	de 5 a 6 anos	3	4,92%
	de 7 a 8 anos	1	1,64%
Engenheiro de Software	acima de 8 anos	3	4,92%
Iniciação Científica	até 2 anos	1	1,64%
Product Owner	acima de 8 anos	1	1,64%
QE	acima de 8 anos	1	1,64%
Suporte	de 3 a 4 anos	1	1,64%
Web designer	de 5 a 6 anos	1	1,64%
Estudante de ADS	até 2 anos	1	1,64%

Fonte: Elaborado pelo autor.

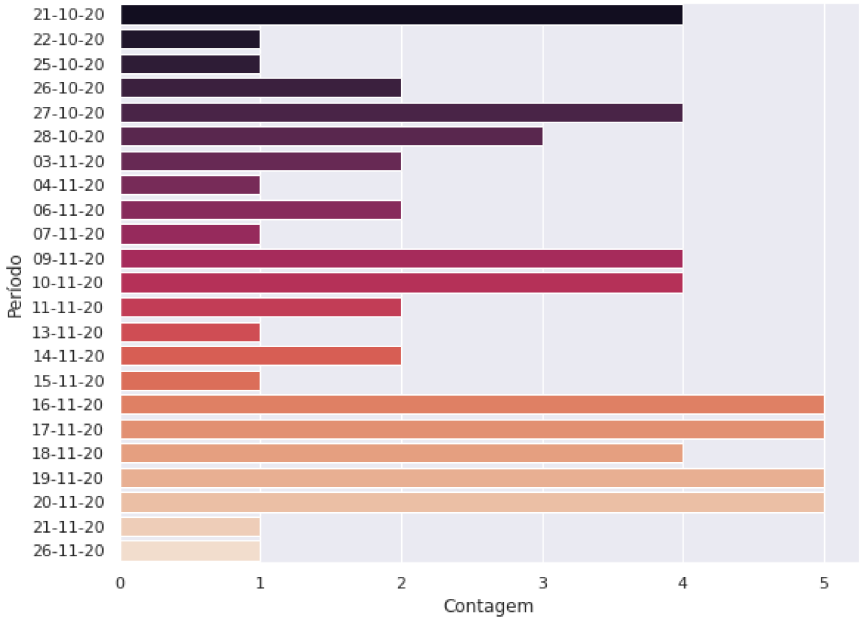
5.2.2 Execução

Por restrições ambientais, todas as etapas executadas na fase experimental se deram de forma *online*. Para isto, fez-se o uso da ferramenta de formulários eletrônicos CognitoForms[1] enviada diretamente por e-mail ou mensageiros instantâneos para os participantes. Uma vez recebido o formulário, todos os participantes possuíam sete dias corridos para o preenchimento do

[1] https://www.cognitoforms.com/

formulário, caso esse tempo fosse excedido, eram automaticamente excluídos do experimento. Ademais, questionários incompletos até a data de corte, ou alterados após esta, também foram excluídos. O período em que a coleta foi realizada deu-se entre 21 de outubro de 2020 a 29 de novembro de 2020, totalizando 40 dias de execução. A frequência de contribuições pode ser vista mediante a Figura 25. Ressalta-se que os dias com aumento no número de participantes são decorrentes de ações de engajamento realizadas por e-mail e mensageiros instantâneos.

Figura 25: Quantidade de respondentes pelo período do experimento.



Fonte: Elaborado pelo autor.

5.3 Resultados

Esta seção organiza a apresenta os resultados obtidos durante a execução do experimento controlado. Enquanto desenvolvido o desenho experimental, cenários de teste foram escolhidos de modo a determinar avaliar as diferenças estatísticas entre diferentes conjuntos de valores apresentados pelas variáveis selecionadas. Os resultados obtidos contemplaram os objetivos definidos na Seção 1.3 desta pesquisa. Os dados coletados pela ferramenta CognitoForms foram importados e analisados utilizando-se a plataforma *online* de edição de *notebooks*, Google Colab[2], a linguagem de programação Python[3] e as bibliotecas Matplotlib[4], Numpy[5], Pandas[6], SciPy[7], statsmodel[8], entre outras. O *notebook* completo utilizado para a importação dos dados

[2]https://colab.research.google.com/
[3]https://www.python.org/
[4]https://matplotlib.org/
[5]https://numpy.org/
[6]https://pandas.pydata.org/
[7]https://www.scipy.org/
[8]https://www.statsmodels.org/

e sintetização dos resultados pode ser consultado no Apêndice D deste trabalho.

Os resultados considerados para a rejeição das hipóteses nulas são apresentados nas seções subsequentes (Seções 5.3.1 e 5.3.2). Seguindo a mesma linha, as evidências apontando para a validade das hipóteses alternativa também serão apresentadas.

5.3.1 Hipótese 1: Indicador de Carga Cognitiva e Percepção de Anomalias

A primeira hipótese investiga os impactos da presença do indicador psicofisiológico de Carga Cognitiva na Percepção de Anomalias experienciada pelo indivíduo revisor. Por se tratar de uma variável qualitativa com comportamento binário dicotômico, para o teste de significância estatística foi utilizado o teste não paramétrico Q de Cochran, eventualmente chamado de Teste Q, no lugar do teste de McNemar, visto que o McNemar é realizado apenas entre dois grupos de variáveis binárias, não sendo possível aplicá-lo em quantidades maiores de grupos ou em estudos com medidas repetidas, como é o caso do teste Q de Cochran (COCHRAN, 1950; CONOVER, 1999).

A Tabela 20 sumariza os resultados obtidos após a aplicação do teste Q de Cochran para cada um dos cenários de testes desenvolvidos na Seção 5.1.4. No cenário C01, o qual testava as questões controle Q01 e experimentais Q03 e Q04, observa-se a impossibilidade de rejeição da hipótese nula, dada a significância de 0,4966 (valor-p $>$ 0,05). De forma semelhante, o cenário C02, o qual avaliava a diferença estatística entre as questões Q02, Q05 e Q06, apresentou significância de 0,5250 (valor-p $>$ 0,05), não sendo possível rejeitar a hipótese nula. Em contrapartida, o teste realizado no cenário C03 apresentou resultados distintos. Este avaliava a diferença estatística entre as questões de 03 a 08, ou seja, todas as pertencentes ao grupo experimental. Neste cenário observou-se significância estatística de 0,0007 (valor-p $\leq$ 0,05) possibilitando assim a rejeição da hipótese nula.

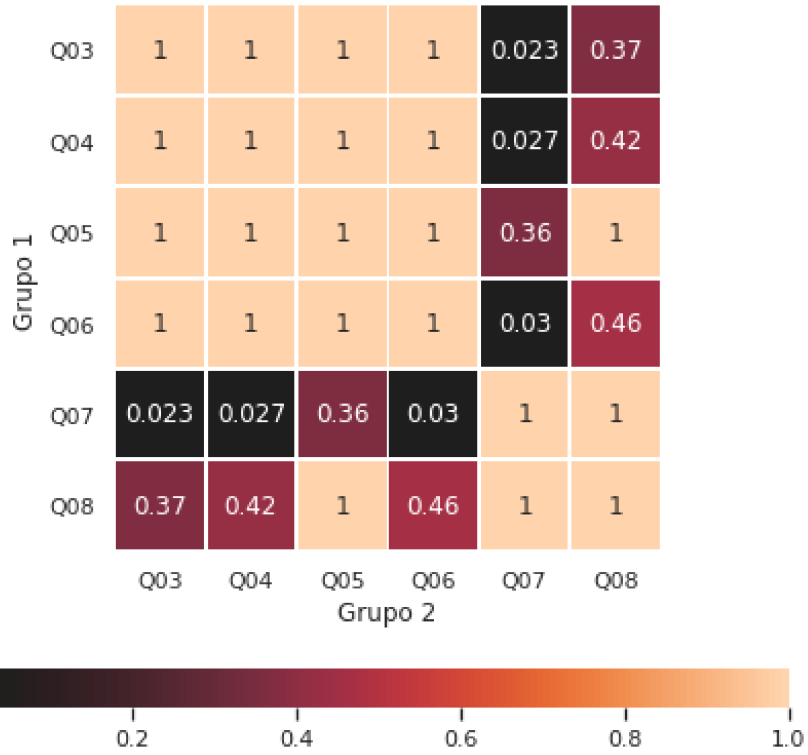
Tabela 20: Resultados dos testes de Q de Cochran avaliando a Percepção de Anomalias para os três cenários da Hipótese Nula H_{1-0}

Cenário	Q de Cochran	Valor-p	Rejeita Hipótese
C01	1,4000	0,4966	Não
C02	1,2857	0,5258	Não
C03	21,2373	0,0007	Sim

Fonte: Elaborado pelo autor.

Não obstante, os resultados positivos apresentados para a rejeição da hipótese nula no cenário C03, o teste Q de Cochran mesmo indicando diferença estatisticamente significante entre os grupos de teste, não indica entre quais componentes foi realmente observada diferença, portanto se fez necessário um teste de análise *post-hoc*. Sendo assim, aplicou-se o teste de Dunn, ajustado segundo Correção de Bonferroni, para análise de múltiplas comparações entre grupos (DUNN, 1964; SHAFFER, 1995).

Figura 26: Resultado do teste de Dunn entre as questões.



Fonte: Elaborado pelo autor.

O mapa de calor exibido na Figura 26 ilustra uma matriz de confusão entre todas as questões componentes do cenário C03 e os respectivos valor-p oriundos do Teste de Dunn. Nele é possível observar que houve significância estatística (valor-$p \leq 0{,}05$) entre as questões Q03 e Q07 (valor-p = 0,023), Q04 e Q07 (valor-p = 0,027) e entre Q06 e Q07 (valor-p = 0,03). É possível observar que a questão Q07 mostrou-se presente em todos os testes que apresentaram significância.

Tabela 21: Resumo das diferenças significantes do teste de Dunn.

Questão	Anomalia de Código	Carga Cognitiva
Q03	Nenhuma	Alta
Q04	Nenhuma	Baixa
Q06	Speculative Generality	Baixa
Q07	Speculative Generality e God Method	Alta

A Tabela 21 busca resumir a composição de todas as questões que apresentaram significância estatística nas comparações múltiplas do teste de Dunn. Analisando-se de forma geral, é possível constatar que a questão Q07, apesar de possuir a variável de Carga Cognitiva assumindo o valor "Alta", assim como a questão Q03, diferencia-se desta e das demais apenas na quantidade de anomalias, possuindo duas.

Conclusão de H_1: este resultado sugere que, a métrica de *Carga Cognitiva*, quando seu valor é exibido como "*Alta*"junto ao trecho de código-fonte revisado, associada a um maior número de *Anomalias de Código* presentes no mesmo trecho, pode influenciar o revisor no que tange a sua *Percepção de Anomalias*.

5.3.2 Hipótese 2: Indicador de Carga Cognitiva e Nível de Intenção de Refatoração

A segunda hipótese foi conjecturada buscando analisar qual seria a influência de se apresentar o indicador de Carga Cognitiva em uma IDE sobre o Nível de Intenção de Refatoração de quem está revisando o código-fonte. Por ser obtida mediante medidas repetidas de uma escala de Likert, a variável Nível de Intenção de Refatoração teve o seu teste de significância estatística realizado por meio da análise de variância (ANOVA) de Friedman. Este caracteriza-se por ser um teste não paramétrico utilizado para comparar três ou mais grupos dependentes quando avaliados mediante uma variável qualitativa ordinal ou quantitativa que não apresente comportamento normal (FRIEDMAN, 1937, 1940; ZIMMERMAN; ZUMBO, 1993).

Tabela 22: Resultados dos testes ANOVA de Friedman avaliando o Nível de Intenção de Refatoração para os três cenários da Hipótese Nula H_{2-0}

Cenário	qui-quadrado (X^2)	Valor-p	Rejeita Hipótese
C01	4,7605	0,0925	Não
C02	2,7312	0,2552	Não
C03	30,9403	0,000009	Sim

Fonte: Elaborado pelo autor.

Na Tabela 22 é apresentada a relação de resultados obtidos segundo a aplicação do teste ANOVA de Friedman para cada cenário delineado na Seção 5.1.4. Conforme apresentado, o cenário C01, composto pela questão de controle Q01 e questões experimentais Q03 e Q04 mostrou significância de 0,092 (valor-$p > 0{,}05$) impossibilitando a rejeição da hipótese nula. O mesmo ocorreu para o cenário C02, oriundo da composição da questão de controle Q02 e das questões experimentais Q05 e Q06. A significância de 0,2552 (valor-$p > 0{,}05$), assim como no cenário anterior, impediu que a hipótese nula fosse rejeitada. Não obstante, os resultados demonstrados, a significância de 0,000009 (valor-$p \leq 0{,}05$), obtida no teste do cenário 03 levou, à rejeição da hipótese nula.

Semelhante ao que foi observado nos testes da hipótese anterior, o teste ANOVA de Friedman, quando aplicado como único método de análise, apenas elucida que há diferença estatisticamente significante entre os grupos de teste, delegando a evidência de quais grupos apresentaram diferença para algum método de análise complementar. Neste caso, utilizaram-se as múltiplas comparações de médias do teste de Diferença Honestamente Significativa (HSD) de Tukey, ajustando-se segundo a Correção de Bonferroni (TUKEY, 1949; SHAFFER, 1995).

Os dados provenientes das comparações múltiplas estão sintetizados na Tabela 24. Nela são explicitadas as questões que compuseram o cenário C03 e os resultados dos valor-p oriundos da

Tabela 23: Resultado do teste Tukey HSD entre as questões.

Grupo 1	Grupo 2	Diferença Média	Valor-p Ajustado	Mínimo	Máximo	Rejeita Hipótese
Q03	Q04	0,0	0,9	-0,6159	0,6159	Não
Q03	Q05	0,1803	0,9	-0,4356	0,7962	Não
Q03	Q06	-0,0164	0,9	-0,6323	0,5995	Não
Q03	**Q07**	**0,6721**	**0,0233**	**0,0562**	**1,288**	**Sim**
Q03	Q08	0,4754	0,2349	-0,1405	1,0913	Não
Q04	Q05	0,1803	0,9	-0,4356	0,7962	Não
Q04	Q06	-0,0164	0,9	-0,6323	0,5995	Não
Q04	**Q07**	**0,6721**	**0,0233**	**0,0562**	**1,288**	**Sim**
Q04	Q08	0,4754	0,2349	-0,1405	1,0913	Não
Q05	Q06	-0,1967	0,9	-0,8126	0,4192	Não
Q05	Q07	0,4918	0,2016	-0,1241	1,1077	Não
Q05	Q08	0,2951	0,7162	-0,3208	0,911	Não
Q06	**Q07**	**0,6885**	**0,0184**	**0,0726**	**1,3044**	**Sim**
Q06	Q08	0,4918	0,2016	-0,1241	1,1077	Não
Q07	Q08	-0,1967	0,9	-0,8126	0,4192	Não

Fonte: Elaborado pelo autor.

aplicação do teste Tukey HSD. Foi constatada significância estatística (valor-$p \leq 0{,}05$), identificada pelos valores em negrito, entre as questões Q03 e Q07 (valor-p ajustado = 0,0233), Q04 e Q07 (valor-p ajustado = 0,0233) e finalmente entre as questões Q06 e Q07 (valor-p ajustado = 0,0184). Assim como constatado na hipótese H1, a questão Q07 foi observada em todos os testes cujos resultados provenientes do Tukey HSD indicaram diferença estatisticamente significante.

Tabela 24: Resumo das diferenças significantes do teste Tukey HSD.

Questão	Anomalia de Código	Carga Cognitiva
Q03	Nenhuma	Alta
Q04	Nenhuma	Baixa
Q06	Speculative Generality	Baixa
Q07	Speculative Generality e God Method	Alta

Todas as questões que apresentaram significância estatística proveniente do teste Tukey HSD estão dispostas na Tabela 24, assim como as variáveis relacionadas e os seus respectivos valores. Tendo em vista os dados apresentados, denota-se que, seguindo o comportamento da hipótese H_1, a questão Q07 apresenta o valor da variável de Carga Cognitiva também sendo "Alta", tendo como único fator de distinção, quando comparada às questões Q03, Q04 e Q06, a presença de duas Anomalias de Código.

Conclusão de H_2: o resultado obtido preconiza que a métrica de *Carga Cognitiva*, quando apresentada junto ao código-fonte e assumindo o seu valor como "*Alta*", associada a um maior número de *Anomalias de Código* presentes no mesmo trecho analisado, pode influenciar no *Nível de Intenção de Refatoração* do revisor.

5.4 Discussão

Nesta avaliação, o experimento controlado buscou explorar o impacto da presença do indicador de Carga Cognitiva em diferentes cenários de revisão de código-fonte. Esta seção destaca pontos da discussão baseados nos resultados encontrados e objetivos contemplados mediante a condução do experimento.

Os dados coletados previamente foram extraídos de forma a se obter uma visão detalhada da composição da amostra em relação ao tempo de experiência dos indivíduos nas suas respectivas funções no mercado de tecnologia, tais dados são utilizados de subsídio para a discussão Seção 5.4.1. Ademais, dados qualitativos avaliando o experimento em diferentes dimensões foram coletados durante a execução do mesmo, conforme listado na Seção 5.2.2, de modo a se obter uma visão qualitativa acerca da percepção dos participantes em relação à qualidade do experimento, aspecto este discutido na Seção 5.4.2.

5.4.1 Experiência na Função

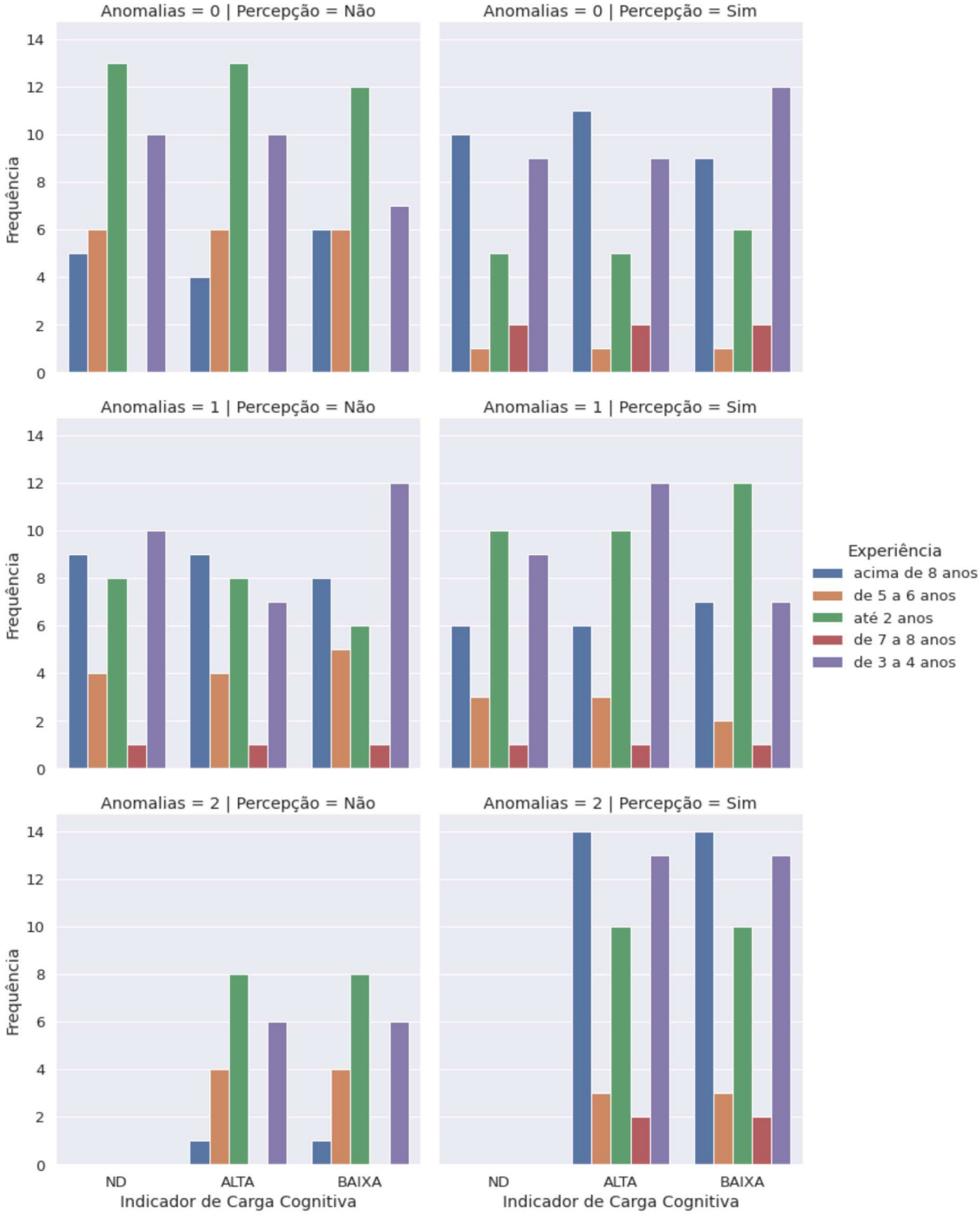
A experiência prévia é conhecida como um dos fatores determinantes que influenciam nos resultados quando os participantes de um experimento tal como este são confrontados uns com os outros. Sendo assim, o objetivo desta seção é realizar observações acerca das variáveis de Percepção de Anomalias e Nível de Intenção de Refatoração quando observadas mediante a consideração do tempo de experiência dos participantes. A Tabela 25 foi criada de forma a sumarizar a relação dos tempos de experiência observados.

Tabela 25: Sumarização da experiência dos participantes.

Experiência	Quantidade	Percentual
de 3 a 4 anos	19	31,15%
até 2 anos	18	29,51%
acima de 8 anos	15	24,59%
de 5 a 6 anos	7	11,47%
de 7 a 8 anos	2	3,28%

Mediante a Figura 27 são apresentados os tempos de experiência em relação à quantidade Anomalias de Código, Indicador de Carga Cognitiva e Percepção de Anomalias. Cada linha do gráfico indica um valor possível da variável Anomalias de Código (0, 1 ou 2), em contraste com cada coluna, que apresenta a variável de Percepção de Anomalias (Sim ou Não). Além disso, em cada gráfico consta os valores do indicador de Carga Cognitiva (ND, Alta, Baixa).

Figura 27: Tempo de experiência em relação à Percepção de Anomalias.



Fonte: Elaborado pelo autor.

Observando-se o gráfico no cenário onde não há Anomalias de Código (primeira linha, colunas 1 e 2, Anomalias = 0), denota-se que, independente dos valores do Indicador de Carga Cognitiva, os profissionais com experiência acima de oito anos, representados pela barra azul, reportaram a presença de Anomalias de Código. Em contrapartida, os desenvolvedores com até

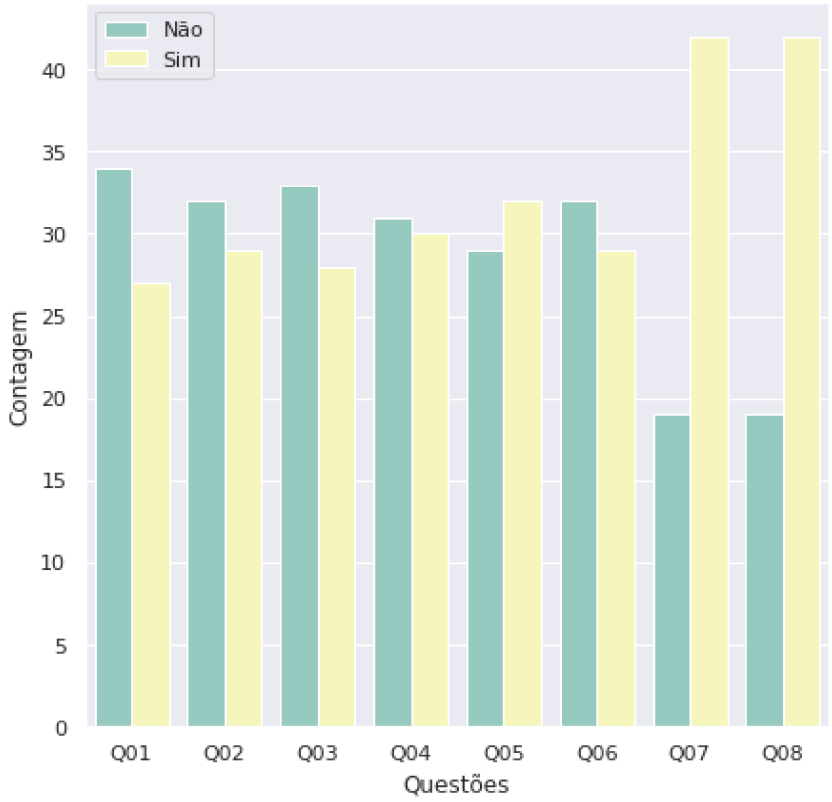
2 anos de experiência, ilustrados pela barra verde, em discordância com o indicador de Carga Cognitiva, não perceberam Anomalias de Código (Coluna 1, Percepção = Não). Baseado nesta constatação, é possível sugerir-se que os desenvolvedores com experiência acima de 8 anos identificaram outras características, consideradas por eles, como anomalias de código, que não as inseridas propositalmente nas questões aplicadas. Em direção oposta à constatação apresentada, quando presente uma Anomalia de Código (segunda linha, colunas 1 e 2, Anomalias = 1), é possível observar-se que os desenvolvedores com mais de 8 anos de experiência não reportaram a presença da anomalia (Coluna 1, Percepção = Não), ao contrário dos desenvolvedores com até 2 anos de experiência (Coluna 2, Percepção = Sim).

No cenário onde foram apresentadas duas Anomalias de Código e houve a Percepção de Anomalias (última linha, Coluna 2, Anomalias = 2 e Percepção = Sim), fica claro que a métrica de Carga Cognitiva não teve influência para a percepção, visto que, tanto para o valor "Alta", como para "Baixa", a quantidade de respondentes que afirmaram terem percebidos as anomalias mantêm-se semelhantes (Coluna 2, Percepção = Sim). Tal observação pode ser corroborada mediante as questões Q07 e Q08 presentes no gráfico ilustrado pela Figura 28. Entretanto, tal constatação mostra-se como válida somente quando analisado isoladamente o cenário onde existem duas Anomalias de Código em conjunto com a presença da métrica de Carga Cognitiva (Q07 e Q08), portanto, conforme apresentado na Tabela 23 e ilustrado na Figura 26 das Seções 5.3.1 e 5.3.2, as conclusões provenientes da hipótese H1 permanecem válidas, visto que esta sugere que existe influência da presença da métrica nas variáveis observadas quando testada entre as questões (de Q03 a Q08).

Por intermédio do gráfico ilustrado na Figura 29 estão dispostos os intervalos de experiência dos participantes em relação ao Nível de Intenção de Refatoração apontados por testes. Cada linha do gráfico apresenta um valor do Indicador de Carga Cognitiva (ND, Alta e Baixa) e cada coluna, a quantidade de Anomalias de Código (0, 1 e 2). Consonante ao que foi apresentado, quando presentes duas Anomalias de Código (coluna 3, linhas 2 e 3, Anomalias = 2) independente dos valores apresentados pelo indicador de Carga Cognitiva (Alta ou Baixa), nota-se que os profissionais com mais de 8 anos de experiência (barra azul) mostram um alto Nível de Intenção de Refatoração (5 - Certamente recomendaria). Já para os profissionais com experiência de até 2 anos (barra verde) e entre 3 e 4 anos, percebe-se aumento no Nível de Intenção de Refatoração quando apresentado o indicador de Carga Cognitiva com o valor "Alta"(coluna 3, linha 2) em comparação ao mesmo cenário com a Carga Cognitiva sendo apresentada como "Baixa"(coluna 3, linha 3).

Aditivamente ao que foi discutido, a Figura 30 sintetiza distribuição dos Níveis de Intenção de Refatoração em relação às questões que foram aplicadas no experimento controlado. Segundo o que é exibido, constata-se que da questão Q01 a Q06 os Níveis de Intenção de Refatoração concentram-se em torno dos valores centrais da Escala de Likert (2, 3 e 4), entretanto, quando observadas as questões Q07 e Q08, as quais apresentam a métrica de Carga Cognitiva como "Alta"e "Baixa", respectivamente, além de possuírem duas Anomalias de Código cada,

Figura 28: Resultado das Percepções de Anomalias observadas no experimento.



Fonte: Elaborado pelo autor.

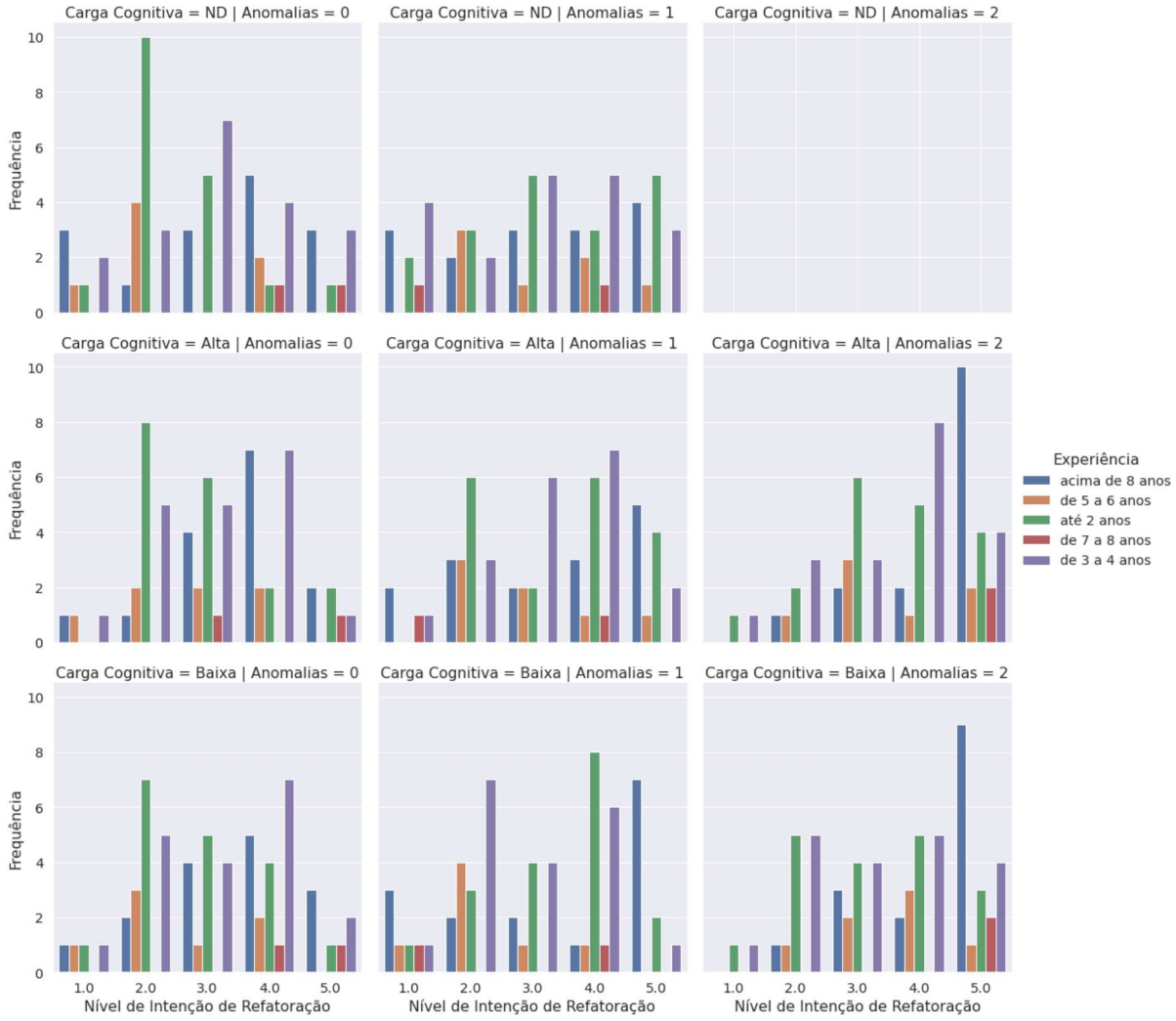
contata-se que a mediana desloca-se para o valor 4 da escala de Likert (Refatoraria), enquanto o terceiro quartil encontra-se extrema-direita, atingindo o valor 5 (Certamente recomendaria), sugerindo que o aumento no número de Anomalias de Código, independente da métrica de Carga Cognitiva,

Não obstante, as discussões decorrentes das relações apresentados pelas Figuras 27 e 29, além das adições provenientes das Figuras 28 e 30, as observações sugerem que a presença de métrica de Carga Cognitiva quando associada a mais Anomalias de Códigos pode influenciar no Nível de Intenção de Refatoração e Percepção de Anomalias, entretanto, mediante o que foi observado, estes valores podem variar conforme o nível de experiência do indivíduo, sendo os níveis superiores menos suscetíveis à influência da métrica de Carga Cognitiva. Por fim, o desenvolvimento deste estudo promoveu a produção de um mecanismo de integração de dados psicofisiológicos, neste caso a Carga Cognitiva, que, em determinadas circunstâncias pode impactar positivamente o processo de desenvolvimento no que tange a etapa de revisão de código.

5.4.2 Avaliação Qualitativa

Na etapa final do experimento controlado, os participantes foram submetidos a um questionário pós-experimento de modo a avaliar o que foi apresentado e as suas percepções acerca

Figura 29: Tempo de experiência em relação ao Nível de Intenção de Refatoração.



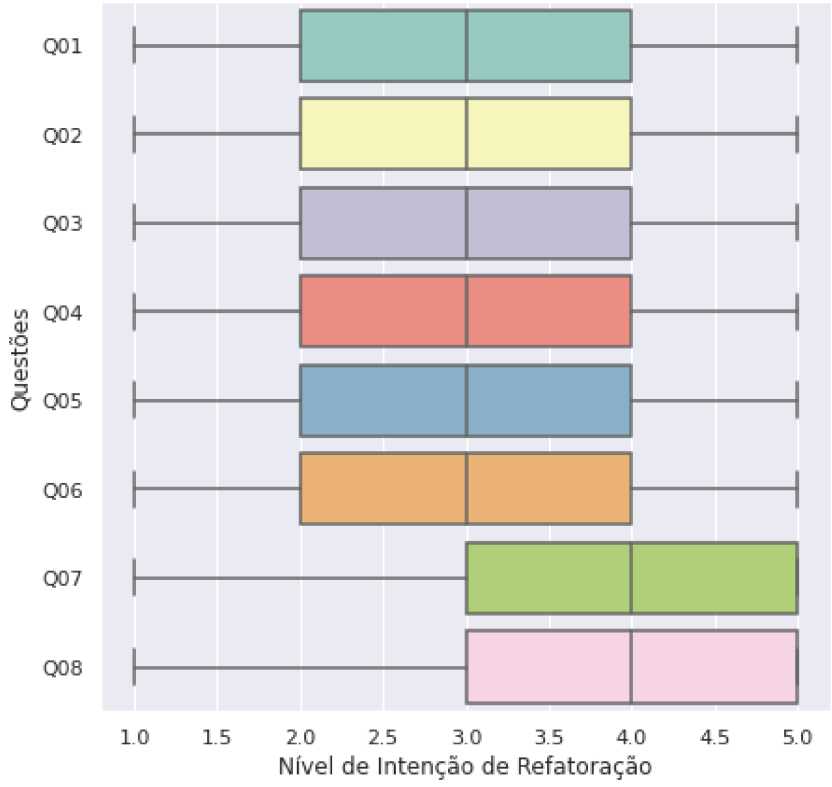
Fonte: Elaborado pelo autor.

das questões a que estes responderam. A composição deste questionário pode ser consultada na Tabela16 da Seção 5.2. Conforme discutido anteriormente, cada uma dessas questões visava avaliar o experimento em quatro dimensões qualitativas: *Clareza*, *Percepção*, *Efeitos* e *Percepção de Uso*. Os resultados obtidos mediante a aplicação do questionário estão dispostos na Tabela 31.

Os resultados apresentados indicam que os participantes consideraram que tiveram tempo suficiente para responder o questionário do experimento controlado (Q09) e o treinamento fornecido foi efetivo para que o experimento pudesse ser executado (Q10), possuindo clareza nas instruções (Q11), ainda que as questões não tenham sido totalmente claras(Q12), principalmente no que tange o código apresentado (Q13) e a métrica utilizada (Q14).

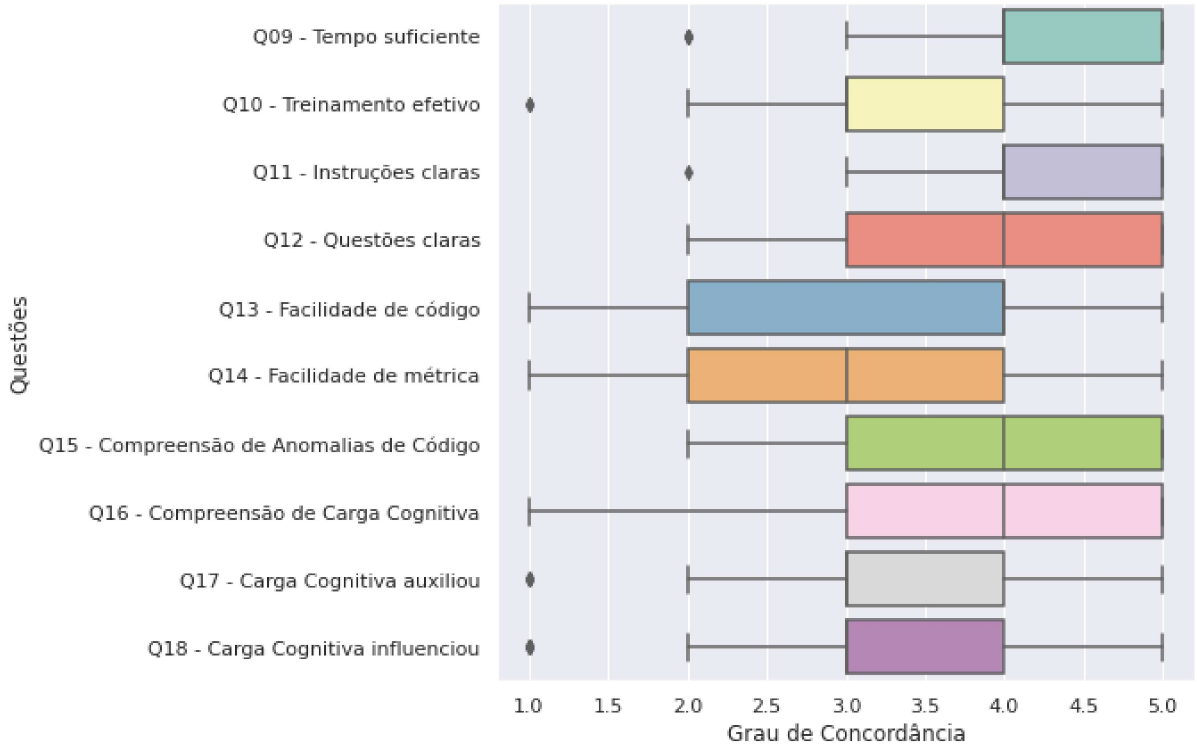
Os participantes ainda afirmaram que, segundo suas percepções, foi possível compreender o que eram as Anomalias de Código (Q15) e a Carga Cognitiva (Q16). Além disso, consideraram que a métrica de Carga Cognitiva os auxiliou, mesmo que não totalmente (Q17), assim como os influenciou nas decisões das respostas fornecidas (Q18).

Figura 30: Resultado dos Níveis de Intenção de Refatoração observados no experimento.



Fonte: Elaborado pelo autor.

Figura 31: Resultado da avaliação qualitativa do experimento.



Fonte: Elaborado pelo autor.

6 CONCLUSÃO

O desenvolvimento deste estudo buscou endereçar como os dados psicofisiológicos estão sendo utilizados no processo de desenvolvimento de software. Para isto, um mapeamento sistemático da literatura foi desenvolvido visando identificar a carência de estudos que trouxessem uma visão geral acerca da temática apresentada, além de prover a compreensão do que já é discutido na literatura corrente. A produção do mapeamento sistemático resultou em um montante inicial de 2084 artigos identificados, dos quais 27 foram selecionados como artigos primários após as etapas de filtragem e refinamento. Estes estudos serviram de base para a formulação das questões de pesquisa apresentadas na Seção 1.2. O principal objetivo deste estudo foi o de propor uma ferramenta para a integração entre dados psicofisiológicos coletados de desenvolvedores e IDEs reais, a fim apresentar tais métricas junto ao código-fonte e verificar o seu impacto no processo de desenvolvimento de software.

Após a obtenção de uma visão geral dos artigos publicados, o próximo passo no desenvolvimento do estudo foi aprofundar-se nas principais teorias relacionadas à aplicação de dados psicofisiológicos na engenharia de software. Para tanto, foram explorados tópicos como aquisição processamento e aplicação dados fisiológicos e os processos mentais ligados a eles, teoria da carga cognitiva e suas implicações na engenharia de software, anomalias de código-fonte e seus impactos no processo de desenvolvimento, além de IDEs e tecnologias envolvidas. Fundamentado pelos conhecimentos adquiridos, foi desenvolvida a ferramenta CognIDE a fim integrar dados oriundos de dispositivos biométricos, tal como EEGs, e exibi-los em IDEs, visando avaliar como estes poderiam impactar no processo de desenvolvimento de software.

Para a avaliação do impacto de se apresentar os dados psicofisiológicos coletados pelo CognIDE nas IDEs, foi confeccionado um experimento controlado envolvendo 61 indivíduos com experiência no mercado de tecnologia, em especial na subárea de desenvolvimento de software e suas correlatas. Neste experimento, os indivíduos selecionados foram expostos a 8 trechos de códigos e estes deveriam indicar se perceberam algumas anomalias de código e com qual intensidade desejariam que aquele trecho de código fosse refatorada. Para tanto, foram manipulas as quantidades de anomalias presentes no código, além de ter sido controlada a exibição de um indicador de nível de carga cognitiva junto a alguns trechos de código analisados. Como principal resultado obtido pelo experimento, observou-se que a métrica de *Carga Cognitiva*, quando presente em trechos de código com mais anomalias, pode influenciar tanto na *Percepção de Anomalias* como no *Nível de Intenção de Refatoração*, mediante a experiência do revisor.

Este capítulo está organizado em três seções conforme pode ser observado a seguir. A Seção 6.1 apresenta as principais contribuições com base nas questões de pesquisa definidas anteriormente (Seção 1.2). A Seção 6.2 realça as limitações e as precauções adotadas no desenvolvimento do estudo. Finalmente, a Seção 6.3 mostra as oportunidades para trabalhos futuros que poderão ser executados após a conclusão desta pesquisa.

6.1 Contribuições

Esta seção apresenta as contribuições adquiridas durante o desenvolvimento deste estudo em concordância com as questões de pesquisa especificadas na Seção 1.2. De um ponto de vista geral, este trabalho propiciou três contribuições principais durante o seu desenvolvimento. Primeiro, a criação de um mapeamento sistemático da literatura visando compreender como dados psicofisiológicos e IDEs estão sendo utilizados na produção de estudos. Segundo, a implementação do protótipo do CognIDE foi desenvolvida permitindo a integração de um dispositivo EEG com a IDE Visual Studio Code, possibilitando a análise de dados gerados durante a produção de artefatos de software. Por último, o experimento controlado permitiu que o CognIDE fosse validado dentro do seu propósito de apresentar dados psicofisiológicos em ambientes integrados de desenvolvimento, gerando assim resultados empíricos. Cada um dos pontos listados abaixo descreve como as contribuições provenientes deste estudo ajudaram a promover a expansão do conhecimento na temática pesquisada.

O estado-da-arte acerca da integração entre dados psicofisiológicos e IDEs (QP01): Um mapeamento sistemático da literatura foi produzido e forneceu um protocolo bem definido para a pesquisa e classificação de estudos relacionados à utilização de dados psicofisiológicos na engenharia de software e a sua aplicação em IDEs. O mapeamento propiciou um maior entendimento sobre as categorias de dados psicofisiológicos, quais dispositivos são utilizados para a sua coleta e como se dá o seu uso na engenharia de software. Ademais, mediante a organização e classificação dos estudos, oportunidades referentes à integração de dados psicofisiológicos e ambientes integrados de desenvolvimento foram identificadas, servindo de norteador para desafios futuros.

A integração entre dados psicofisiológicos e IDEs (QP02): O desenvolvimento da ferramenta CognIDE, assim como a sua abordagem de operação, permitiram contemplar a oportunidade de pesquisa que apontava para a lacuna na literatura acerca da integração de dados psicofisiológicos e ambientes integrados de desenvolvimento. A implementação da ferramenta permitiu comprovar a possibilidade de se coletar dados de desenvolvedores durante a execução de suas tarefas, e utilizá-los tanto para o desenvolvimento de novos estudos, como fator de melhoria no processo de desenvolvimento de software.

Não obstante, a própria abordagem de integração de dados adotada durante a idealização do CognIDE, serve de norteador para a formação de arquiteturas e fluxos de coleta de dados psicofisiológicos. Por fim, tal abordagem, quando combinada a estudos já produzidos, como os de Clark e Sharif (2017); Arnaoudova, Fakhoury e Roy (2020); Peitek et al. (2019), pode vir a expandir a aplicabilidade do uso de tal natureza de dados em experimentos na engenharia de software, além de possibilitar a criação de novas ferramentas visando a evolução da indústria e dos desenvolvedores.

A influência do indicador de Carga Cognitiva no processo de desenvolvimento de software (QP03): Por fim, umas das maiores contribuições no desenvolvimento deste estudo foi a criação e execução do experimento controlado. Tal experimento foi capaz de produzir observações empíricas sobre o impacto de se apresentar dados psicofisiológicos em IDEs. Por intermédio dos resultados obtidos (Seção 5.3), constatou-se que, quando observado de forma geral, a métrica apresentada de alta Carga Cognitiva serve de fator auxiliar na identificação de Anomalias de Código, além de subsidiar a formação do Nível de Intenção de Refatoração de determinados trechos de código-fonte avaliados. Tal contribuição ainda reforça a aplicabilidade da ferramenta CognIDE no que tange a apresentação de dados nas IDEs, tanto na realização de novos experimentos que venham a avaliar os impactos da exibição, quanto na produção de funcionalidades personalizadas nas IDEs sensíveis aos dados psicofisiológicos dos desenvolvedores.

6.2 Limitações

Segundo o que foi discutido no decorrer deste trabalho, o seu desenvolvimento propiciou a produção da ferramenta CognIDE e sua abordagem de integração de dados, além da produção de conhecimento empírico sobre dados psicofisiológicos e o impacto de sua exibição em IDEs. No entanto, ressalta-se que se faz necessária a investigação de uma série de fatores limitantes ligados à produção do trabalho, os quais serão enumerados e descritos a seguir.

- **Volume de Dados:** Na fase de desenvolvimento da ferramenta CognIDE, foi utilizado o EEG NeuroSky MindWave Mobile 2. Este sensor é composto por um canal de medição e uma taxa de amostragem de apenas 512Hz, valor este inferior aos demais EEGs do mesmo segmento presentes no mercado. Apesar de tais especificações, o MindWave Mobile 2 é capaz de gerar um grande volume de dados durante a sua utilização, em especial quando operando de forma síncrona, ou seja, enviando o fluxo de dados para computador com o conector de dados instalado. Esta característica mostra-se como um fator limitante tanto na operação de manipulação de tal volume de dados para a devida análise, como na capacidade de armazenamento necessária.

- **Capacidade de Processamento:** Não obstante, os recursos necessários para o armazenamento dos dados coletados, o computador onde a ferramenta CognIDE está operando demanda uma alta capacidade de processamento e de memória, tendo em vista que a manipulação dos dados oriundos do EEG dá-se em tempo real. Este comportamento mostra-se como uma limitação dado que o CognIDE estaria operando de forma concorrente às demais ferramentas presentes no computador hospedeiro, tais como a própria IDE e o ambiente de desenvolvimento, competindo por recursos e reduzindo tanto o seu desempenho, como os das demais aplicações.

- **Eletroencefalografia:** Dispositivos EEG, em especial os modelos básicos de baixo custo,

podem sofrer de diversos modos às interferências provenientes do ambiente, desde ruídos mioelétricos proveniente da interação sinérgica muscular, a interferências eletromagnéticas oriundas de equipamentos eletrônicos, em especial àqueles dotados de cargas indutivas e capacitivas. Estas interferências podem levar à formação de artefatos durante a análise dos dados, podendo conduzir a resultados equivocados sobre a coleta realizada.

- **Experimento Controlado:** Diversas variáveis podem influenciar em quaisquer experimentos relacionados à leitura de código-fonte. O experimento controlado conduzido neste estudo buscou-se reproduzir uma etapa específica do fluxo de trabalho dos desenvolvedores. Aumentar a complexidade do cenário analisado, acaba por acarretar aumento das variáveis a serem controladas. Mesmo que estratégias de mitigação através de tratamento estatístico tenham sido aplicadas, tal controle de variáveis mostra-se como um fator limitante no experimento realizado, entretanto esta limitação era conhecida durante o desenho experimental e foi aceita visando manter o cenário do experimento fidedigno ao ambiente real.

- **Privacidade e Aspectos Éticos:** Ainda que o experimento controlado não tenha utilizado dados psicofisiológicos coletados de desenvolvedores reais, a popularização de dispositivos sensores, além da possibilidade trazida pelo CognIDE na utilização de tais dispositivos como insumo para a criação de métricas dentro da IDEs, de um ponto de vista ético e de privacidade, pode levar à limitação da aplicabilidade da ferramenta a apenas o meio acadêmico. Sendo assim, ressalta-se que tal utilização de dados em empresas pode ferir legislações vigentes acerca da privacidade de seus funcionários.

6.3 Trabalhos Futuros

Não obstante, os resultados obtidos por este trabalho, além da aplicabilidade dos mesmos tanto na indústria como na área acadêmica, espera-se que novos trabalhos sejam desenvolvidos visando a expansão do conhecimento obtido e a identificação de novos desafios na temática abordada. Portanto, esta seção foca na apresentação dos principais trabalhos futuros sugeridos.

- **Suporte a Dispositivos:** Em sua primeira versão, o CognIDE apenas suporta a integração de dados psicofisiológicos provenientes do EEG NeuroSky Mindwave Mobile 2. Entretanto, a arquitetura adotada para a implementação do CognIDE, permite a extensão do suporte a novos dispositivos. Em trabalhos futuros pode-se trabalha na criação de componentes capazes de coletar dados de EMGs, Eye-trackers, EDA, entre outros, e integrá-los no Visual Studio Code utilizando a abordagem sugerida pelo CognIDE, possibilitando assim a condução de novos categorias de experimentos controlados.

- **Suporte a IDEs:** Assim como no suporte a dispositivos, a versão inicial do CognIDE foi desenhada para funcionar com o Visual Studio Code. Apesar disso, a sua flexibilidade

arquitetural permite que em estudos futuros, sejam implementadas novas extensões de forma a suportar outras IDEs existentes no mercado.

- **Implementação do *Language Server*:** Como mencionado na Seção 4.3, tanto o CognIDE Extension como o CognIDE Server, utilizam os eventos de alteração de linhas de código como gatilho para disparar a coleta das medições provenientes do EEG. Tal granularidade de eventos pode ser trabalhada mediante o desenvolvimento de um servidor de linguagens (*Language Server*) capaz identificar blocos de código, como métodos, classes ou variáveis, de forma a permitir uma geração de eventos mais precisa. Além disso, por intermédio do servidor de linguagens, seria possível suportar as sintaxes de quaisquer linguagens de programação existentes para a geração de eventos.

REFERÊNCIAS

ALSHBATAT, A. I. N.; VIAL, P. J.; PREMARATNE, P.; TRAN, L. C. EEG-based Brain-computer Interface for Automating Home Appliances. **Journal of Computers**, [S.l.], v. 9, n. 9, p. 2159–2166, 2014.

ANDREASSI, J. L. **Psychophysiology: human behavior & physiological response, 5th ed.** Mahwah, NJ, US: Lawrence Erlbaum Associates Publishers, 2007. xxiii, 538–xxiii, 538 p.

ANTONENKO, P.; PAAS, F.; GRABNER, R.; GOG, T. van. Using Electroencephalography to Measure Cognitive Load. **Educational Psychology Review**, [S.l.], v. 22, n. 4, p. 425–438, 2010.

ARNAOUDOVA, V.; FAKHOURY, S.; ROY, D. VITALSE : visualizing eye tracking and biometric data. In: DEMONSTRATIONS IN 42 INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2020. **Anais...** [S.l.: s.n.], 2020. p. 1–4.

ASTROMSKIS, S.; BAVOTA, G.; JANES, A.; RUSSO, B.; Di Penta, M. Patterns of developers behaviour: a 1000-hour industrial study. **Journal of Systems and Software**, [S.l.], v. 132, p. 85–97, 2017.

BEDNARIK, R. Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. **International Journal of Human Computer Studies**, [S.l.], v. 70, n. 2, p. 143–155, 2012.

BEELDERS, T.; Du Plessis, J. P. The influence of syntax highlighting on scanning and reading behaviour for source code. In: ACM INTERNATIONAL CONFERENCE PROCEEDING SERIES, 2016. **Anais...** Association for Computing Machinery (ACM), 2016. v. 26-28-Sept, p. 1–10.

BISCHOFF, V.; FARIAS, K.; GONÇALES, L. J.; Victória Barbosa, J. L. Integration of feature models: a systematic mapping study. **Information and Software Technology**, [S.l.], v. 105, n. October 2016, p. 209–225, 2019.

BISCHOFF, V.; FARIAS, K.; MENZEN, J. P.; PESSIN, G. Technological support for detection and prediction of plant diseases: a systematic mapping study. **Computers and Electronics in Agriculture**, [S.l.], v. 181, n. November 2020, p. 105922, 2021.

BRITTON, B. K.; TESSER, A. Effects of prior knowledge on use of cognitive capacity in three complex cognitive tasks. **Journal of Verbal Learning and Verbal Behavior**, [S.l.], v. 21, n. 4, p. 421–436, 1982.

CARBONERA, C. E.; FARIAS, K.; BISCHOFF, V. Software development effort estimation: a systematic mapping study. **IET Software**, [S.l.], v. 14, n. 4, p. 328–344, 2020.

CLARK, B.; SHARIF, B. ITraceVis: visualizing eye movement data within eclipse. **Proceedings - 2017 IEEE Working Conference on Software Visualization, VISSOFT 2017**, [S.l.], v. 2017-Octob, p. 22–32, 2017.

COCHRAN, W. G. The Comparison of Percentages in Matched Samples. **Biometrika**, [S.l.], v. 37, n. 3/4, p. 256–266, 1950.

CONOVER, W. J. **Practical Nonparametric Statistics**. [S.l.]: Wiley, 1999. (Wiley Series in Probability and Statistics).

COUCEIRO, R.; De Carvalho, P.; BRANCO, M. C.; MADEIRA, H.; BARBOSA, R.; DURAES, J.; DUARTE, G.; CASTELHANO, J.; DUARTE, C.; TEIXEIRA, C.; LARANJEIRO, N.; MEDEIROS, J. Spotting Problematic Code Lines using Nonintrusive Programmers' Biofeedback. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING, ISSRE, 2019. **Proceedings...** [S.l.: s.n.], 2019. v. 2019-Octob, p. 93–103.

COUCEIRO, R.; DUARTE, G.; DURAES, J.; CASTELHANO, J.; DUARTE, C.; TEIXEIRA, C.; BRANCO, M. C.; CARVALHO, P.; MADEIRA, H. Biofeedback augmented software engineering: monitoring of programmers' mental effort. In: IEEE/ACM 41ST INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: NEW IDEAS AND EMERGING RESULTS, ICSE-NIER 2019, 2019., 2019. **Proceedings...** Institute of Electrical and Electronics Engineers Inc., 2019. p. 37–40.

CRK, I.; KLUTHE, T. Toward using alpha and theta brain waves to quantify programmer expertise. **2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2014**, [S.l.], p. 5373–5376, 2014.

DIANE, C. What is a "mapping study?". **Journal of the Medical Library Association**, [S.l.], v. 104, n. 1, p. 76–78, 2016.

Du Boulay, B. Some difficulties of learning to program. **Journal of Educational Computing Research**, [S.l.], v. 2, n. 1, p. 57–73, 1986.

DUNN, O. J. Multiple Comparisons Using Rank Sums. **Technometrics**, [S.l.], v. 6, n. 3, p. 241–252, 1964.

DURAES, J.; MADEIRA, H.; CASTELHANO, J.; DUARTE, C.; BRANCO, M. C. WAP: understanding the brain at software debugging. **Proceedings - International Symposium on Software Reliability Engineering, ISSRE**, [S.l.], p. 87–92, 2016.

FAKHOURY, S. Moving towards objective measures of program comprehension. **ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering**, [S.l.], p. 936–939, 2018.

FAKHOURY, S.; MA, Y.; ARNAOUDOVA, V.; ADESOPE, O. The effect of poor source code lexicon and readability on developers' cognitive load. **Proceedings - International Conference on Software Engineering**, [S.l.], p. 286–296, 2018.

FAKHOURY, S.; ROY, D.; MA, Y.; ARNAOUDOVA, V.; ADESOPE, O. Measuring the impact of lexical and structural inconsistencies on developers' cognitive load during bug localization. **Empirical Software Engineering**, [S.l.], v. 25, n. 3, p. 2140–2178, may 2020.

FISHBURN, F. A.; NORR, M. E.; MEDVEDEV, A. V.; VAIDYA, C. J. Sensitivity of fNIRS to cognitive state and load. **Frontiers in Human Neuroscience**, [S.l.], v. 8, n. 1 FEB, p. 1–11, 2014.

FLOYD, B.; SANTANDER, T.; WEIMER, W. Decoding the Representation of Code in the Brain: an fmri study of code review and expertise. **Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017**, [S.l.], p. 175–186, 2017.

FOWLER, M. **Refactoring**: improving the design of existing code. [S.l.]: Addison-Wesley Professional, 2018.

FRIEDMAN, M. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. **Journal of the American Statistical Association**, [S.l.], v. 32, n. 200, p. 675–701, 1937.

FRIEDMAN, M. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. **The Annals of Mathematical Statistics**, [S.l.], v. 11, n. 1, p. 86–92, 1940.

FRITZ, T.; BEGEL, A.; MÜLLER, S. C.; YIGIT-ELLIOTT, S.; ZÜGER, M. Using psycho-physiological measures to assess task difficulty in software development. **Proceedings - International Conference on Software Engineering**, [S.l.], n. 1, p. 402–413, 2014.

FRITZ, T.; MULLER, S. C. Leveraging Biometric Data to Boost Software Developer Productivity. **2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering**, [S.l.], p. 66–77, 2016.

FUCCI, D.; GIRARDI, D.; NOVIELLI, N.; QUARANTA, L.; LANUBILE, F. A replication study on code comprehension and expertise using lightweight biometric sensors. **IEEE International Conference on Program Comprehension**, [S.l.], v. 2019-May, p. 311–322, 2019.

FUNKE, G.; KNOTT, B.; MANCUSO, V. F.; STRANG, A.; ESTEPP, J.; MENKE, L.; BROWN, R.; DUKES, A.; MILLER, B. Evaluation of subjective and EEG-based measures of mental workload. **Communications in Computer and Information Science**, [S.l.], v. 373, n. PART I, p. 412–416, 2013.

GONCALES, L.; FARIAS, K.; Da Silva, B.; FESSLER, J. Measuring the cognitive load of software developers: a systematic mapping study. **IEEE International Conference on Program Comprehension**, [S.l.], v. 2019-May, n. March, p. 42–52, 2019.

GONÇALES, L. J.; FARIAS, K.; De Oliveira, T. C.; SCHOLL, M. Comparison of software design models: an extended systematic mapping study. **ACM Computing Surveys**, [S.l.], v. 52, n. 3, p. 1–41, 2019.

GONÇALES, L. J.; FARIAS, K.; SCHOLL, M.; ROBERTO VERONEZ, M.; OLIVEIRA, T. C. de. Comparison of design models: a systematic mapping study. **International Journal of Software Engineering and Knowledge Engineering**, [S.l.], v. 25, n. 09n10, p. 1765–1769, 2015.

GUI, Q.; RUIZ-BLONDET, M. V.; LASZLO, S.; JIN, Z. A survey on brain biometrics. **ACM Computing Surveys**, [S.l.], v. 51, n. 6, p. 1–38, 2019.

HAAG, A.; GORONZY, S.; SCHAICH, P.; WILLIAMS, J. Emotion recognition using bio-sensors: first steps towards an automatic system. In: LECTURE NOTES IN ARTIFICIAL INTELLIGENCE (SUBSERIES OF LECTURE NOTES IN COMPUTER SCIENCE), 2004, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 2004. v. 3068, p. 36–48.

HEJMADY, P.; NARAYANAN, N. H. Visual attention switching patterns of programmers debugging with an IDE. **ETRA '12 Proceedings of the Symposium on Eye Tracking Research and Applications**, [S.l.], p. 197–200, 2012.

HOCHMAN, B.; NAHAS, F. X.; De Oliveira Filho, R. S.; FERREIRA, L. M. Desenhos de pesquisa. **Acta Cirurgica Brasileira**, [S.l.], v. 20, n. SUPPL. 2, p. 2–9, 2005.

HOFFMAN, B.; SCHRAW, G. Conceptions of efficiency: applications in learning and problem solving. **Educational Psychologist**, [S.l.], v. 45, n. 1, p. 1–14, 2010.

IKUTANI, Y.; UWANO, H. Brain activity measurement during program comprehension with NIRS. **2014 IEEE/ACIS 15th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2014 - Proceedings**, [S.l.], p. 1–6, 2014.

JACQUES, J. T.; KRISTENSSON, O. Understanding the effects of code presentation. **PLATEAU 2015 - Proceedings of the 6th Workshop on Evaluation and Usability of Programming Languages and Tools**, [S.l.], p. 27–30, 2015.

KLIMESCH, W. EEG alpha and theta oscillations reflect cognitive and memory performance: a rklimesch, w. (1999). eeg alpha and theta oscillations reflect cognitive and memory performance: a review and analysis. brain research reviews, 29(2-3), 169–195. doi:10.1016/s016. **Brain Research Reviews**, [S.l.], v. 29, n. 2-3, p. 169–195, 1999.

KLIMESCH, W.; SCHACK, B.; SAUSENG, P. The functional significance of theta and upper alpha oscillations. **Experimental Psychology**, [S.l.], v. 52, n. 2, p. 99–108, 2005.

KOSTI, M. V.; GEORGIADIS, K.; ADAMOS, D. A.; LASKARIS, N.; SPINELLIS, D.; ANGELIS, L. Towards an affordable brain computer interface for the assessment of programmers' mental workload. **International Journal of Human Computer Studies**, [S.l.], v. 115, n. March, p. 52–66, 2018.

LEE, S.; HOOSHYAR, D.; JI, H.; NAM, K.; LIM, H. Mining biometric data to predict programmer expertise and task difficulty. **Cluster Computing**, [S.l.], v. 21, n. 1, p. 1097–1107, mar 2018.

LEE, S.; MATTESON, A.; HOOSHYAR, D.; KIM, S.; JUNG, J.; NAM, G.; LIM, H. Comparing Programming Language Comprehension between Novice and Expert Programmers Using EEG Analysis. **Proceedings - 2016 IEEE 16th International Conference on Bioinformatics and Bioengineering, BIBE 2016**, [S.l.], p. 350–355, 2016.

LI, M.; LU, B. L. Emotion classification based on gamma-band EEG. In: ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY: ENGINEERING THE FUTURE OF BIOMEDICINE, EMBC 2009, 31., 2009. **Proceedings...** [S.l.: s.n.], 2009. p. 1323–1326.

LI, S. Z.; JAIN, A. K. (Ed.). **Encyclopedia of Biometrics, Second Edition**. [S.l.]: Springer US, 2015.

LIN, Y. T.; WU, C. C.; HOU, T. Y.; LIN, Y. C.; YANG, F. Y.; CHANG, C. H. Tracking Students' Cognitive Processes during Program Debugging-An Eye-Movement Approach. **IEEE Transactions on Education**, [S.l.], v. 59, n. 3, p. 175–186, 2016.

LOGOZZO, F.; BARNETT, M.; FANDRICH, M.; COUSOT, P.; COUSOT, R. A semantic integrated development environment. **SPLASH'12 - Proceedings of the 2012 ACM Conference on Systems, Programming, and Applications: Software for Humanity**, [S.l.], p. 15–16, 2012.

MASON, R.; SIMON; COOPER, G.; WILKS, B. Flipping the assessment of cognitive load: why and how. **ICER 2016 - Proceedings of the 2016 ACM Conference on International Computing Education Research**, [S.l.], p. 43–52, 2016.

MENS, T.; TOURWÉ, T. A survey of software refactoring. **IEEE Transactions on Software Engineering**, [S.l.], v. 30, n. 2, p. 126–139, 2004.

MORRISON, B. B.; DORN, B.; GUZDIAL, M. Measuring cognitive load in introductory CS: adaptation of an instrument. **ICER 2014 - Proceedings of the 10th Annual International Conference on International Computing Education Research**, [S.l.], p. 131–138, 2014.

MORSHAD, S.; MAZUMDER, M. R.; AHMED, F. Analysis of brain wave data using neurosky mindwave mobile II. In: OF THE 31ST , 2020. **Anais...** Association for Computing Machinery, 2020.

NIEDERMEYER, E.; OTHERS. The normal EEG of the waking adult. **Electroencephalography: basic principles, clinical applications and related fields**, [S.l.], v. 20, n. 4, p. 149–173, 1999.

NUNEZ, P. L.; CUTILLO, B. A. **Neocortical dynamics and human EEG rhythms**. New York: Oxford University Press, USA, 1995.

OBAIDELLAH, U.; Al Haek, M.; CHENG, P. C. A survey on the usage of eye-Tracking in computer programming. **ACM Computing Surveys**, [S.l.], v. 51, n. 1, p. 1–58, 2018.

OIZUMI, W.; SOUSA, L.; OLIVEIRA, A.; GARCIA, A.; AGBACHI, A. B.; OLIVEIRA, R.; LUCENA, C. On the identification of design problems in stinky code: experiences and tool support. **Journal of the Brazilian Computer Society**, [S.l.], v. 24, n. 1, p. 1–30, 2018.

PAAS, F. G. Training Strategies for Attaining Transfer of Problem-Solving Skill in Statistics: a cognitive-load approach. **Journal of Educational Psychology**, [S.l.], v. 84, n. 4, p. 429–434, 1992.

PAAS, F. G.; Van Merrienboer, J. J. The efficiency of instructional conditions: an approach to combine mental effort and performance measures. **Human Factors**, [S.l.], v. 35, n. 4, p. 737–743, 1993.

PEITEK, N. A neuro-cognitive perspective of program comprehension. **Proceedings - International Conference on Software Engineering**, [S.l.], p. 496–499, 2018.

PEITEK, N.; APEL, S.; BRECHMANN, A.; PARNIN, C.; SIEGMUND, J. CodersMUSE: multi-modal data exploration of program-comprehension experiments. **IEEE International Conference on Program Comprehension**, [S.l.], v. 2019-May, p. 126–129, 2019.

PEITEK, N.; SIEGMUND, J.; PARNIN, C.; APEL, S.; BRECHMANN, A. Toward conjoint analysis of simultaneous eye-tracking and fMRI data for program-comprehension studies. In: EMIP 2018: EYE MOVEMENTS IN PROGRAMMING, 2018. **Proceedings...** Association for Computing Machinery: Inc, 2018.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. **12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008**, [S.l.], p. 68–77, 2008.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: an update. **Information and Software Technology**, [S.l.], v. 64, p. 1–18, 2015.

PFURTSCHELLER, G.; Lopes Da Silva, F. H. Event-related EEG/MEG synchronization and desynchronization: basic principles. **Clinical Neurophysiology**, [S.l.], v. 110, n. 11, p. 1842–1857, 1999.

RADEVSKI, S.; HATA, H.; MATSUMOTO, K. Real-time monitoring of neural state in assessing and improving software developers' productivity. **Proceedings - 8th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2015**, [S.l.], p. 93–96, 2015.

RD, L.; GR, F.; PM, C.; RJ, D. Neural activation during selective attention to subjective emotional responses. **Neuroreport**, [S.l.], v. 8, n. 18, p. 3969, 1997.

RICCA, F.; Di Penta, M.; TORCHIANO, M.; TONELLA, P.; CECCATO, M. How developers' experience and ability influence web application comprehension tasks supported by UML stereotypes: a series of four experiments. **IEEE Transactions on Software Engineering**, [S.l.], v. 36, n. 1, p. 96–118, 2010.

ROSTAMI, S.; SHENFIELD, A.; SIGURNJAK, S.; FAKOREDE, O. Evaluation of Mental Workload and Familiarity in Human Computer Interaction with Integrated Development Environments using Single-Channel EEG. In: PSYCHOLOGY OF PROGRAMMING INTEREST GROUP 2015 - 26TH ANNUAL WORKSHOP, 2015., 2015. **Anais...** [S.l.: s.n.], 2015. n. July, p. 7–22.

SEGALOTTO, M. **ARNI: an eeg-based model to measure program comprehension**. São Leopoldo: Universidade do Vale do Rio dos Sinos, 2018.

SHAFFER, J. P. Multiple Hypothesis Testing. **Annual Review of Psychology**, [S.l.], v. 46, n. 1, p. 561–584, 1995.

SHARAFI, Z.; SOH, Z.; GUÉHÉNEUC, Y. G. A systematic literature review on the usage of eye-tracking in software engineering. **Information and Software Technology**, [S.l.], v. 67, p. 79–107, 2015.

SHARIF, B.; SHAFFER, T.; WISE, J.; MALETIC, J. I. Tracking Developers' Eyes in the IDE. **IEEE Software**, [S.l.], v. 33, n. 3, p. 105–108, 2016.

SIEGMUND, J.; KÄSTNER, C.; APEL, S.; PARNIN, C.; BETHMANN, A.; LEICH, T.; SAAKE, G.; BRECHMANN, A. Understanding understanding source code with functional magnetic resonance imaging. **Proceedings - International Conference on Software Engineering**, [S.l.], n. 1, p. 378–389, 2014.

SIEGMUND, J.; PEITEK, N.; BRECHMANN, A.; PARNIN, C.; APEL, S. Studying programming in the neuroage. **Communications of the ACM**, [S.l.], v. 63, n. 6, p. 30–34, 2020.

SIEGMUND, J.; PEITEK, N.; PARNIN, C.; APEL, S.; HOFMEISTER, J.; KÄSTNER, C.; BEGEL, A.; BETHMANN, A.; BRECHMANN, A. Measuring neural efficiency of program comprehension. **Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering**, [S.l.], p. 140–150, 2017.

SNIPES, W.; MURPHY-HILL, E.; FRITZ, T.; VAKILIAN, M.; DAMEVSKI, K.; NAIR, A. R.; SHEPHERD, D. A Practical Guide to Analyzing IDE Usage Data. **The Art and Science of Analyzing Software Data**, [S.l.], p. 85–138, 2015.

SOLOVEY, E. T.; AFERGAN, D.; PECK, E. M.; HINCKS, S. W.; JACOB, R. J. Designing implicit interfaces for physiological computing: guidelines and lessons learned using fnirs. **ACM Transactions on Computer-Human Interaction**, [S.l.], v. 21, n. 6, p. 35:1–35:27, 2015.

STIPACEK, A.; GRABNER, R. H.; NEUPER, C.; FINK, A.; NEUBAUER, A. C. Sensitivity of human EEG alpha band desynchronization to different working memory components and increasing levels of memory load. **Neuroscience Letters**, [S.l.], v. 353, n. 3, p. 193–196, 2003.

SWELLER, J. Cognitive load during problem solving: effects on learning. **Cognitive Science**, [S.l.], v. 12, n. 2, p. 257–285, 1988.

SWELLER, J.; AYRES, P.; KALYUGA, S. **Cognitive Load Theory**. 1st. ed. London: Springer, 2011. 273 p.

SZU, H.; HSU, C.; MOON, G.; YAMAKAWA, T.; TRAN, B. Q.; JUNG, T. P.; LANDA, J. Smartphone Household Wireless Electroencephalogram Hat. **Applied Computational Intelligence and Soft Computing**, [S.l.], v. 2013, p. 1–8, 2013.

TUKEY, J. W. Comparing Individual Means in the Analysis of Variance. **Biometrics**, [S.l.], v. 5, n. 2, p. 99–114, 1949.

UYSAL, M. P. Evaluation of learning environments for object-oriented programming: measuring cognitive load with a novel measurement technique. **Interactive Learning Environments**, [S.l.], v. 24, n. 7, p. 1590–1609, 2016.

VELTMAN, J. A.; GAILLARD, A. W. Physiological workload reactions to increasing levels of task difficulty. **Ergonomics**, [S.l.], v. 41, n. 5, p. 656–669, 1998.

VIDAL, S.; BERRA, I.; ZULLIANI, S.; MARCOS, C.; Andrés Díaz Pace, J. Assessing the refactoring of brain methods. **ACM Transactions on Software Engineering and Methodology**, [S.l.], v. 27, n. 1, p. 1–43, 2018.

VIEIRA, R. D.; FARIAS, K. Usage of psychophysiological data as an improvement in the context of software engineering: a systematic mapping study. **ACM International Conference Proceeding Series**, [S.l.], 2020.

VIEIRA, R. D.; FARIAS, K. CognIDE: a psychophysiological data integrator approach for visual studio code. **ACM International Conference Proceeding Series**, [S.l.], p. 393–398, 2020.

WIERINGA, R.; MAIDEN, N.; MEAD, N.; ROLLAND, C. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. **Requirements Engineering**, [S.l.], v. 11, n. 1, p. 102–107, 2006.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. **Experimentation in software engineering**. [S.l.]: Springer Science & Business Media, 2012.

XIE, B.; SALVENDY, G. Review and reappraisal of modelling and predicting mental workload in single and multi-task environments. **Work & Stress: An International Journal of Work , Health & Organisations**, [S.l.], n. January 2013, p. 37–41, 2010.

YAMASHITA, A.; COUNSELL, S. Code smells as system-level indicators of maintainability: an empirical study. **Journal of Systems and Software**, [S.l.], v. 86, n. 10, p. 2639–2653, 2013.

YEH, M. K.; GOPSTEIN, D.; YAN, Y.; ZHUANG, Y. Detecting and comparing brain activity in short program comprehension using EEG. **Proceedings - Frontiers in Education Conference, FIE**, [S.l.], v. 2017-Octob, n. 1444827, p. 1–5, 2017.

YIN, M.; LI, B.; TAO, C. Using cognitive easiness metric for program comprehension. **2nd International Conference on Software Engineering and Data Mining, SEDM 2010**, [S.l.], p. 134–139, 2010.

ZAYOUR, I.; HAJJDIAB, H. How much integrated development environments (IDEs) improve productivity? **Journal of Software**, [S.l.], v. 8, n. 10, p. 2425–2431, 2013.

ZAYOUR, I.; HAMDAR, A. A qualitative study on debugging under an enterprise IDE. **Information and Software Technology**, [S.l.], v. 70, p. 130–139, 2016.

ZIMMERMAN, D. W.; ZUMBO, B. D. Relative Power of the Wilcoxon Test, the Friedman Test, and Repeated-Measures ANOVA on Ranks. **The Journal of Experimental Education**, [S.l.], v. 62, n. 1, p. 75–86, 1993.

APÊNDICE A STRINGS DE BUSCA INDIVIDUAIS

Base de Dados	String de Busca
ACM Digital Library	acmdlTitle: (+(eeg fmri electroencephalography neural brain cognitive psychometric bioinformatic "functional magnetic resonance imaging") +(software developer programmer professional) +(ide code editor "integrated development environmentsoftware engineering")) OR recordAbstract: (+(eeg fmri electroencephalography neural brain cognitive psychometric bioinformatic "functional magnetic resonance imaging") +(software developer programmer professional) +(ide code editor "integrated development environmentsoftware engineering"))
IEEE Xplore	(sofware OR developer OR programmer OR professional OR "software engineering") AND (eeg OR fmri OR electroencephalography OR neural OR brain OR cognitive OR psychometric OR bioinformatic OR "functional magnetic resonance imaging") AND (ide OR code OR editor OR "integrated development environment")
Science Direct	({"software engineering"} OR developer OR programmer OR code) AND (IDE) AND (eeg OR Fmri OR cognitive OR bci)
Scopus	ALL (("software engineering"OR developer OR programmer OR code) AND (ide OR "integrated development environment") AND (eeg OR fmri OR cognitive OR bci)) AND PUBYEAR >2008 AND (LIMIT-TO (SUBJAREA , "COMP") OR LIMIT-TO (SUBJAREA , "ENGI")) AND (LIMIT-TO (LANGUAGE , "English"))
Springer Link	("integrated development environment"OR IDE) AND ("software engineering"OR "software development"OR programmer OR developer) AND (bci OR eeg OR fmri OR cognitive)
DBLP	IDE|"integrated development environment"
Semantic Scholar	(IDE OR "integrated development environment")

APÊNDICE B SELEÇÃO DE ESTUDOS PRIMÁRIOS

S01 ALSHBATAT, A. I. N.; VIAL, P. J.; PREMARATNE, P.; TRAN, L. C. EEG-based Brain-computer Interface for Automating Home Appliances. **Journal of Computers**, [S.l.], v. 9, n. 9, p. 2159–2166, 2014

S02 BEDNARIK, R. Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. **International Journal of Human Computer Studies**, [S.l.], v. 70, n. 2, p. 143–155, 2012

S03 BEELDERS, T.; Du Plessis, J. P. The influence of syntax highlighting on scanning and reading behaviour for source code. In: ACM INTERNATIONAL CONFERENCE PROCEEDING SERIES, 2016. **Anais...** Association for Computing Machinery (ACM), 2016. v. 26-28-Sept, p. 1–10

S04 CRK, I.; KLUTHE, T. Toward using alpha and theta brain waves to quantify programmer expertise. **2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2014**, [S.l.], p. 5373–5376, 2014

S05 DURAES, J.; MADEIRA, H.; CASTELHANO, J.; DUARTE, C.; BRANCO, M. C. WAP: understanding the brain at software debugging. **Proceedings - International Symposium on Software Reliability Engineering, ISSRE**, [S.l.], p. 87–92, 2016

S06 FAKHOURY, S.; MA, Y.; ARNAOUDOVA, V.; ADESOPE, O. The effect of poor source code lexicon and readability on developers' cognitive load. **Proceedings - International Conference on Software Engineering**, [S.l.], p. 286–296, 2018

S07 FAKHOURY, S. Moving towards objective measures of program comprehension. **ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering**, [S.l.], p. 936–939, 2018

S08 FLOYD, B.; SANTANDER, T.; WEIMER, W. Decoding the Representation of Code in the Brain: an fmri study of code review and expertise. **Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017**, [S.l.], p. 175–186, 2017

S09 FRITZ, T.; BEGEL, A.; MÜLLER, S. C.; YIGIT-ELLIOTT, S.; ZÜGER, M. Using psycho-physiological measures to assess task difficulty in software development. **Proceedings - International Conference on Software Engineering**, [S.l.], n. 1, p. 402–413, 2014

S10 FRITZ, T.; MULLER, S. C. Leveraging Biometric Data to Boost Software Developer Productivity. **2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering**, [S.l.], p. 66–77, 2016

S11 FUNKE, G.; KNOTT, B.; MANCUSO, V. F.; STRANG, A.; ESTEPP, J.; MENKE, L.; BROWN, R.; DUKES, A.; MILLER, B. Evaluation of subjective and EEG-based measures of mental workload. **Communications in Computer and Information Science**, [S.l.], v. 373, n. PART I, p. 412–416, 2013

S12 HEJMADY, P.; NARAYANAN, N. H. Visual attention switching patterns of programmers debugging with an IDE. **ETRA '12 Proceedings of the Symposium on Eye Tracking Research and Applications**, [S.l.], p. 197–200, 2012

S13 IKUTANI, Y.; UWANO, H. Brain activity measurement during program comprehension with NIRS. **2014 IEEE/ACIS 15th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2014 - Proceedings**, [S.l.], p. 1–6, 2014

S14 LEE, S.; MATTESON, A.; HOOSHYAR, D.; KIM, S.; JUNG, J.; NAM, G.; LIM, H. Comparing Programming Language Comprehension between Novice and Expert Programmers Using EEG Analysis. **Proceedings - 2016 IEEE 16th International Conference on Bioinformatics and Bioengineering, BIBE 2016**, [S.l.], p. 350–355, 2016

S15 LOGOZZO, F.; BARNETT, M.; FANDRICH, M.; COUSOT, P.; COUSOT, R. A semantic integrated development environment. **SPLASH'12 - Proceedings of the 2012 ACM Conference on Systems, Programming, and Applications: Software for Humanity**, [S.l.], p. 15–16, 2012

S16 OBAIDELLAH, U.; Al Haek, M.; CHENG, P. C. A survey on the usage of eye-Tracking in computer programming. **ACM Computing Surveys**, [S.l.], v. 51, n. 1, p. 1–58, 2018

S17 PEITEK, N. A neuro-cognitive perspective of program comprehension. **Proceedings - International Conference on Software Engineering**, [S.l.], p. 496–499, 2018

S18 RADEVSKI, S.; HATA, H.; MATSUMOTO, K. Real-time monitoring of neural state in assessing and improving software developers' productivity. **Proceedings - 8th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2015**, [S.l.], p. 93–96, 2015

S19 ROSTAMI, S.; SHENFIELD, A.; SIGURNJAK, S.; FAKOREDE, O. Evaluation of Mental Workload and Familiarity in Human Computer Interaction with Integrated Development Environments using Single-Channel EEG. In: PSYCHOLOGY OF PROGRAMMING INTEREST GROUP 2015 - 26TH ANNUAL WORKSHOP, 2015., 2015. **Anais...** [S.l.: s.n.], 2015. n. July, p. 7–22

S20 SHARAFI, Z.; SOH, Z.; GUÉHÉNEUC, Y. G. A systematic literature review on the usage of eye-tracking in software engineering. **Information and Software Technology**, [S.l.], v. 67, p. 79–107, 2015

S21 SIEGMUND, J.; KÄSTNER, C.; APEL, S.; PARNIN, C.; BETHMANN, A.; LEICH, T.; SAAKE, G.; BRECHMANN, A. Understanding understanding source code with functional magnetic resonance imaging. **Proceedings - International Conference on Software Engineering**, [S.l.], n. 1, p. 378–389, 2014

S22 SIEGMUND, J.; PEITEK, N.; PARNIN, C.; APEL, S.; HOFMEISTER, J.; KÄSTNER, C.; BEGEL, A.; BETHMANN, A.; BRECHMANN, A. Measuring neural efficiency of program comprehension. **Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering**, [S.l.], p. 140–150, 2017

S23 SZU, H.; HSU, C.; MOON, G.; YAMAKAWA, T.; TRAN, B. Q.; JUNG, T. P.; LANDA, J. Smartphone Household Wireless Electroencephalogram Hat. **Applied Computational Intelligence and Soft Computing**, [S.l.], v. 2013, p. 1–8, 2013

S24 UYSAL, M. P. Evaluation of learning environments for object-oriented programming: measuring cognitive load with a novel measurement technique. **Interactive Learning Environments**, [S.l.], v. 24, n. 7, p. 1590–1609, 2016

S25 YEH, M. K.; GOPSTEIN, D.; YAN, Y.; ZHUANG, Y. Detecting and comparing brain activity in short program comprehension using EEG. **Proceedings - Frontiers in Education Conference, FIE**, [S.l.], v. 2017-Octob, n. 1444827, p. 1–5, 2017

S26 YIN, M.; LI, B.; TAO, C. Using cognitive easiness metric for program comprehension. **2nd International Conference on Software Engineering and Data Mining, SEDM 2010**, [S.l.], p. 134–139, 2010

S27 ZAYOUR, I.; HAMDAR, A. A qualitative study on debugging under an enterprise IDE. **Information and Software Technology**, [S.l.], v. 70, p. 130–139, 2016

APÊNDICE C QUESTIONÁRIO DO EXPERIMENTO CONTROLADO

CognIDE

Esta seção tem por finalidade apresentar os conceitos necessários para devida compreensão e execução das etapas subsequentes.

Anomalias de Código

Uma Anomalia de Código, ou *Bad Smell*, pode ser definido como quaisquer características no código-fonte de um programa que *possivelmente* indiquem um problema maior (FOWLER, 2018). Ainda, as anomalias podem ser conceituados como sendo "*certas estruturas no código que indiquem a violação de princípios básicos de design e impactam negativamente na sua qualidade*" (SURYANARAYANA, 2014). Sendo assim, uma anomalia não necessariamente seria considerada como um defeito no código, entretanto pode servir como um indicador de problemas futuros os quais, por sua vez, poderiam desencadear a confecção de artefatos de software defeituosos.

Carga Cognitiva

Esforço cognitivo, ou carga cognitiva se refere ao nível de utilização de recursos psicológicos como **memórias**, **atenção**, **percepção**, **representação de conhecimento**, **raciocínio** e **criatividade** na resolução de problemas. Os principais fatores ligados ao aumento da carga cognitiva são: **experiência com atividades similares**, **grau de abstração**, **número de atividades simultâneas**, **horas de descanso**, **nível de fadiga,** entre outros. Enquanto a resolução de um problema sem nenhuma experiência prévia é considerado como muito desgastante, a resolução de um problema ao qual já foi desenvolvido um padrão de respostas usuais (esquema) envolve apenas um mínimo de esforço cognitivo que facilmente passa despercebido pelo sujeito (SWELLER, 1988).

Referências

FOWLER, Martin. **Refactoring: improving the design of existing code**. Addison-Wesley Professional, 2018.

SURYANARAYANA, Girish; SAMARTHYAM, Ganesh; SHARMA, Tushar. **Refactoring for software design smells: managing technical debt**. Morgan Kaufmann, 2014.

SWELLER, John. Cognitive load during problem solving: Effects on learning. **Cognitive science**, v. 12, n. 2, p. 257-285, 1988.

Próxima

CognIDE

Introdução | **Perfil** | Questionário | Avaliação

Perfil

Estas perguntas têm por objetivo trazer uma visão geral sobre o perfil dos participantes.

Qual a sua idade? *

- ◯ menos de 20 anos
- ◯ de 20 a 29 anos
- ◯ de 30 a 39 anos
- ◯ de 40 a 49 anos
- ◯ mais de 49 anos

Qual a sua escolaridade? *

- ◯ Ensino Médio
- ◯ Ensino Técnico
- ◯ Ensino Superior Incompleto
- ◯ Ensino Superior Completo
- ◯ Especialização/MBA
- ◯ Mestrado/Doutorado
- ◯ Outros

Há quanto tempo trabalha na área de Tecnologia da Informação? *

- ◯ até 2 anos
- ◯ de 3 a 4 anos
- ◯ de 5 a 6 anos
- ◯ de 7 a 8 anos
- ◯ acima de 8 anos

Qual opção abaixo descreve melhor o seu cargo atual? *

- ◯ Desenvolvedor de Software/Programador
- ◯ Engenheiro de Software
- ◯ Arquiteto de Software
- ◯ Tech Lead
- ◯ Analista de Sistemas
- ◯ Outros

Email *

Este e-mail será utilizado apenas se for necessária entrevista e não será divulgado de forma alguma.

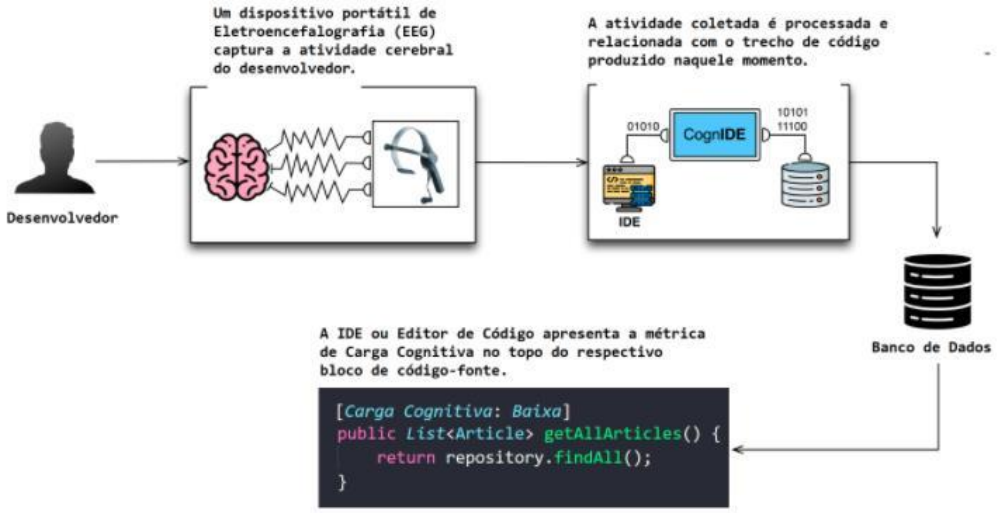
Voltar | Próxima

CognIDE

Introdução | Perfil | **Questionário** | Avaliação

Percepção e Recomendação de Refactoring

Você é o **líder técnico** de um time de desenvolvimento, onde sua principal função é a de **garantir a qualidade do código**. Você foi incumbido de revisar novos artefatos utilizando uma ferramenta experimental, a qual indica o **nível de Carga Cognitiva** (ou esforço mental) apresentada pelo **desenvolvedor** durante a produção do código, conforme ilustrado a seguir:



Fazendo uso da funcionalidade apresentada, você deverá revisar os trechos de código apresentados a seguir, respondendo se você **percebeu** alguma anomalia de código (*Bad Smell*) e/ou o quanto você **recomendaria** o seu refactoring.

Questão 01

Analise o trecho de código a seguir e responda às perguntas:

```java
@Service
public class CSVService {
    @Autowired
    ArticleRepository repository;

    public void save(MultipartFile file) {
        try {
            List<Article> articles = CSVHelper.csvToArticles(file.getInputStream());
            repository.saveAll(articles);
        } catch (IOException e) {
            throw new RuntimeException("Failed save the CSV data: " + e.getMessage());
        }
    }

    public ByteArrayInputStream load() {
        List<Article> articles = repository.findAll();

        ByteArrayInputStream in = CSVHelper.articlesToCSV(articles);
        return in;
    }

    public List<Article> getAllArticles() {
        return repository.findAll();
    }
}
```

Você percebeu alguma anomalia de código (Bad Smell) no trecho apresentado?

◯ Sim ◉ Não

O quanto você recomendaria o refactoring do trecho de código apresentado? *

◯ Certamente não recomendaria ◯ Não recomendaria ◯ Talvez recomendasse ◯ Recomendaria

◯ Certamente recomendaria

Questão 02

Analise o trecho de código a seguir e responda às perguntas:

```java
public interface Responsible {

}
```

```java
public abstract class BaseResponse implements Responsible {

    private String correlationId;

    public BaseResponse(String correlationId) {
        this.correlationId = correlationId;
    }

    public String getCorrelationId() {
        return correlationId;
    }
}
```

```java
public abstract class BaseResponseMessage extends BaseResponse {

    private String messageType;

    public BaseResponseMessage(String messageType, String correlationId) {
        super(correlationId);
        this.messageType = messageType;
    }

    public String getMessageType() {
        return messageType;
    }
}
```

```java
public class ResponseMessage extends BaseResponseMessage {

    private String message;

    public ResponseMessage(String message, String messageType, String correlationId) {
        super(messageType, correlationId);
        this.message = message;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

Você percebeu alguma anomalia de código (Bad Smell) no trecho apresentado?

◯ Sim ◉ Não

O quanto você recomendaria o refactoring do trecho de código apresentado? *

◯ Certamente não recomendaria ◯ Não recomendaria ◯ Talvez recomendasse ◯ Recomendaria

◯ Certamente recomendaria

Questão 03

Analise o trecho de código a seguir e responda às perguntas:

```
[Carga Cognitiva: Alta]
@Service
public class CSVService {
    @Autowired
    ArticleRepository repository;

    public void save(MultipartFile file) {
        try {
            List<Article> articles = CSVHelper.csvToArticles(file.getInputStream());
            repository.saveAll(articles);
        } catch (IOException e) {
            throw new RuntimeException("Failed save the CSV data: " + e.getMessage());
        }
    }

    public ByteArrayInputStream load() {
        List<Article> articles = repository.findAll();

        ByteArrayInputStream in = CSVHelper.articlesToCSV(articles);
        return in;
    }

    public List<Article> getAllArticles() {
        return repository.findAll();
    }
}
```

Você percebeu alguma anomalia de código (Bad Smell) no trecho apresentado?

◯ Sim ◉ Não

O quanto você recomendaria o refactoring do trecho de código apresentado? *

◯ Certamente não recomendaria ◯ Não recomendaria ◯ Talvez recomendasse ◯ Recomendaria

◯ Certamente recomendaria

Questão 04

Analise o trecho de código a seguir e responda às perguntas:

```
[Carga Cognitiva: Baixa]
@Service
public class CSVService {
    @Autowired
    ArticleRepository repository;

    public void save(MultipartFile file) {
        try {
            List<Article> articles = CSVHelper.csvToArticles(file.getInputStream());
            repository.saveAll(articles);
        } catch (IOException e) {
            throw new RuntimeException("Failed save the CSV data: " + e.getMessage());
        }
    }

    public ByteArrayInputStream load() {
        List<Article> articles = repository.findAll();

        ByteArrayInputStream in = CSVHelper.articlesToCSV(articles);
        return in;
    }

    public List<Article> getAllArticles() {
        return repository.findAll();
    }
}
```

Você percebeu alguma anomalia de código (Bad Smell) no trecho apresentado?

◯ Sim ◉ Não

O quanto você recomendaria o refactoring do trecho de código apresentado? *

◯ Certamente não recomendaria ◯ Não recomendaria ◯ Talvez recomendasse ◯ Recomendaria

◯ Certamente recomendaria

Questão 05

Analise o trecho de código a seguir e responda às perguntas:

```
[Carga Cognitiva: Alta]
public interface Responsible {

}
```

```
[Carga Cognitiva: Alta]
public abstract class BaseResponse implements Responsible {

    private String correlationId;

    public BaseResponse(String correlationId) {
        this.correlationId = correlationId;
    }

    public String getCorrelationId() {
        return correlationId;
    }
}
```

```
[Carga Cognitiva: Alta]
public abstract class BaseResponseMessage extends BaseResponse {

    private String messageType;

    public BaseResponseMessage(String messageType, String correlationId) {
        super(correlationId);
        this.messageType = messageType;
    }

    public String getMessageType() {
        return messageType;
    }
}
```

```
[Carga Cognitiva: Alta]
public class ResponseMessage extends BaseResponseMessage {

    private String message;

    public ResponseMessage(String message, String messageType, String correlationId) {
        super(messageType, correlationId);
        this.message = message;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

Você percebeu alguma anomalia de código (Bad Smell) no trecho apresentado?

◯ Sim ◉ Não

O quanto você recomendaria o refactoring do trecho de código apresentado? *

◯ Certamente não recomendaria ◯ Não recomendaria ◯ Talvez recomendasse ◯ Recomendaria

◯ Certamente recomendaria

Questão 06

Analise o trecho de código a seguir e responda às perguntas:

```
[Carga Cognitiva: Baixa]
public interface Responsible {

}
```

```
[Carga Cognitiva: Baixa]
public abstract class BaseResponse implements Responsible {

    private String correlationId;

    public BaseResponse(String correlationId) {
        this.correlationId = correlationId;
    }

    public String getCorrelationId() {
        return correlationId;
    }
}
```

```
[Carga Cognitiva: Baixa]
public abstract class BaseResponseMessage extends BaseResponse {

    private String messageType;

    public BaseResponseMessage(String messageType, String correlationId) {
        super(correlationId);
        this.messageType = messageType;
    }

    public String getMessageType() {
        return messageType;
    }
}
```

```
[Carga Cognitiva: Baixa]
public class ResponseMessage extends BaseResponseMessage {

    private String message;

    public ResponseMessage(String message, String messageType, String correlationId) {
        super(messageType, correlationId);
        this.message = message;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

Você percebeu alguma anomalia de código (Bad Smell) no trecho apresentado?

◯ Sim ◉ Não

O quanto você recomendaria o refactoring do trecho de código apresentado? *

◯ Certamente não recomendaria ◯ Não recomendaria ◯ Talvez recomendasse ◯ Recomendaria

◯ Certamente recomendaria

Questão 07

Analise o trecho de código a seguir e responda às perguntas:

```
[Carga Cognitiva: Alta]
@PostMapping("/upload")
public ResponseEntity<ResponseMessage> uploadFile(@RequestParam("file") MultipartFile file) {
  String message = "";
  String TYPE = "text/csv";
  SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
  Date date = new Date();

  if (!TYPE.equals(file.getContentType())) {
    try (BufferedReader fileReader = new BufferedReader(
          new InputStreamReader(file.getInputStream(), "UTF-8"));
          CSVParser csvParser = new CSVParser(
            fileReader,
            CSVFormat.DEFAULT.withFirstRecordAsHeader()
              .withIgnoreHeaderCase()
              .withTrim());
        ) {

      System.out.println("Datetime: " + formatter.format(date) + "Message: Parsing the CSV file.");
      List<Article> articles = new ArrayList<Article>();
      Iterable<CSVRecord> csvRecords = csvParser.getRecords();

      for (CSVRecord csvRecord : csvRecords) {
        Article article = new Article(Long.parseLong(csvRecord.get("Id")), csvRecord.get("Title"),
            csvRecord.get("Description"), Boolean.parseBoolean(csvRecord.get("Published")));

        articles.add(article);
      }

      System.out.println("Datetime: " + formatter.format(date) + "Message: Saving the parsed articles.");
      repository.saveAll(articles);

      message = "Uploaded the file successfully: " + file.getOriginalFilename();
      return ResponseEntity.status(HttpStatus.OK).body(new ResponseMessage(message));
    } catch (Exception e) {

      System.out.println("Datetime: " + formatter.format(date) +
        "Message: An error has been occurred while uploading the file.");
      message = "Could not upload the file: " + file.getOriginalFilename() + "!";
      return ResponseEntity.status(HttpStatus.EXPECTATION_FAILED).body(new ResponseMessage(message));
    }
  }

  System.out.println("Datetime: " + formatter.format(date) + "Message: File uploaded in wrong format.");
  message = "Please upload a file in CSV format.";
  return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(new ResponseMessage(message));
}
```

Você percebeu alguma anomalia de código (Bad Smell) no trecho apresentado?

◯ Sim ◉ Não

O quanto você recomendaria o refactoring do trecho de código apresentado? *

◯ Certamente não recomendaria ◯ Não recomendaria ◯ Talvez recomendasse ◯ Recomendaria

◯ Certamente recomendaria

Questão 08

Analise o trecho de código a seguir e responda às perguntas:

```
[Carga Cognitiva: Baixa]
@PostMapping("/upload")
public ResponseEntity<ResponseMessage> uploadFile(@RequestParam("file") MultipartFile file) {
  String message = "";
  String TYPE = "text/csv";
  SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
  Date date = new Date();

  if (!TYPE.equals(file.getContentType())) {
    try (BufferedReader fileReader = new BufferedReader(
          new InputStreamReader(file.getInputStream(), "UTF-8"));
          CSVParser csvParser = new CSVParser(
            fileReader,
            CSVFormat.DEFAULT.withFirstRecordAsHeader()
              .withIgnoreHeaderCase()
              .withTrim());
        ) {

      System.out.println("Datetime: " + formatter.format(date) + "Message: Parsing the CSV file.");
      List<Article> articles = new ArrayList<Article>();
      Iterable<CSVRecord> csvRecords = csvParser.getRecords();

      for (CSVRecord csvRecord : csvRecords) {
        Article article = new Article(Long.parseLong(csvRecord.get("Id")), csvRecord.get("Title"),
            csvRecord.get("Description"), Boolean.parseBoolean(csvRecord.get("Published")));

        articles.add(article);
      }

      System.out.println("Datetime: " + formatter.format(date) + "Message: Saving the parsed articles.");
      repository.saveAll(articles);

      message = "Uploaded the file successfully: " + file.getOriginalFilename();
      return ResponseEntity.status(HttpStatus.OK).body(new ResponseMessage(message));
    } catch (Exception e) {

      System.out.println("Datetime: " + formatter.format(date) +
        "Message: An error has been occurred while uploading the file.");
      message = "Could not upload the file: " + file.getOriginalFilename() + "!";
      return ResponseEntity.status(HttpStatus.EXPECTATION_FAILED).body(new ResponseMessage(message));
    }
  }

  System.out.println("Datetime: " + formatter.format(date) + "Message: File uploaded in wrong format.");
  message = "Please upload a file in CSV format.";
  return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(new ResponseMessage(message));
}
```

Você percebeu alguma anomalia de código (Bad Smell) no trecho apresentado?

◯ Sim ◉ Não

O quanto você recomendaria o refactoring do trecho de código apresentado? *

◯ Certamente não recomendaria ◯ Não recomendaria ◯ Talvez recomendasse ◯ Recomendaria

◯ Certamente recomendaria

Voltar Próxima

CognIDE

Introdução | Perfil | Questionário | **Avaliação**

Avaliação

As questões a seguir visam avaliar a sua experiência ao responder este questionário. Utilize as opções para dizer o quanto vocês concorda com cada uma das afirmações abaixo.

Clareza *

	Discordo totalmente	Discordo	Neutral	Concordo	Concordo totalmente
Tive tempo o suficiente para responder o questionário*	○	○	○	○	○
O treinamento me capacitou para executar o experimento.*	○	○	○	○	○
As instruções do treinamento foram claras.*	○	○	○	○	○
As questões apresentadas foram claras.*	○	○	○	○	○

Percepção *

	Discordo totalmente	Discordo	Neutral	Concordo	Concordo totalmente
Não tive dificuldade em compreender o código apresentado.*	○	○	○	○	○
Não tive dificuldade em compreender a métrica de Carga Cognitiva apresentada.*	○	○	○	○	○

Efeitos *

	Discordo totalmente	Discordo	Neutral	Concordo	Concordo totalmente
Compreendi o que são anomalias (Bad Smells) de código.*	○	○	○	○	○
Compreendi o que é Carga Cognitiva.*	○	○	○	○	○

Percepção de Uso *

	Discordo totalmente	Discordo	Neutral	Concordo	Concordo totalmente
A métrica de Carga Cognitiva auxiliou na percepção de anomalias de código.*	○	○	○	○	○
A métrica de Carga Cognitiva influenciou na recomendação da refatoração do código apresentado.*	○	○	○	○	○

Voltar | Submeter

APÊNDICE D ANÁLISE DOS DADOS COLETADOS

Importação de Bibliotecas e Dados para Análise

```
!pip install scikit-posthocs


import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import graphviz
import seaborn as sb
import itertools
import io
from google.colab import files
from google.colab import drive
import scipy.stats as stats
from scipy.stats import friedmanchisquare
from scipy.stats import chisquare
import scikit_posthocs as sp
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.stats.contingency_tables import cochrans_q
from statsmodels.stats.anova import AnovaRM
from mlxtend.evaluate import mcnemar_table
```

```
Requirement already satisfied: scikit-posthocs in /usr/local/lib/python3.7/dist-packages (0.6.6)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from scikit-posthocs) (1.19.5)
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (from scikit-posthocs) (0.11.1)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages (from scikit-posthocs) (0.10.2)
Requirement already satisfied: pandas>=0.20.0 in /usr/local/lib/python3.7/dist-packages (from scikit-posthocs) (1.1.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from scikit-posthocs) (1.4.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from scikit-posthocs) (3.2.2)
Requirement already satisfied: patsy>=0.4.0 in /usr/local/lib/python3.7/dist-packages (from statsmodels->scikit-posthocs) (0.5.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.20.0->scikit-posthocs) (2018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.20.0->scikit-posthocs) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->scikit-posthocs) (1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->scikit-posthocs) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->scikit-posthocs) (0.10.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from patsy>=0.4.0->statsmodels->scikit-posthocs) (1.15.0)
```

```
# mounting GDrive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
raw_data =  pd.read_csv('/content/drive/MyDrive/Mestrado/DISS2020/DISS2020 - Dados - Tabulação CSV.csv', decimal=",", enc
```

Extração de Dados

```
df1 = raw_data[['ID','Q00_PERF_IDA', 'Q00_PERF_EXP', 'Q00_PERF_PRO','Q01_DESC_GRP','Q01_SMLL_QTD','Q01_CARG_CAT', 'Q01_IN
df2 = raw_data[['ID','Q00_PERF_IDA', 'Q00_PERF_EXP', 'Q00_PERF_PRO','Q02_DESC_GRP','Q02_SMLL_QTD','Q02_CARG_CAT', 'Q02_IN
df3 = raw_data[['ID','Q00_PERF_IDA', 'Q00_PERF_EXP', 'Q00_PERF_PRO','Q03_DESC_GRP','Q03_SMLL_QTD','Q03_CARG_CAT', 'Q03_IN
df4 = raw_data[['ID','Q00_PERF_IDA', 'Q00_PERF_EXP', 'Q00_PERF_PRO','Q04_DESC_GRP','Q04_SMLL_QTD','Q04_CARG_CAT', 'Q04_IN
df5 = raw_data[['ID','Q00_PERF_IDA', 'Q00_PERF_EXP', 'Q00_PERF_PRO','Q05_DESC_GRP','Q05_SMLL_QTD','Q05_CARG_CAT', 'Q05_IN
df6 = raw_data[['ID','Q00_PERF_IDA', 'Q00_PERF_EXP', 'Q00_PERF_PRO','Q06_DESC_GRP','Q06_SMLL_QTD','Q06_CARG_CAT', 'Q06_IN
df7 = raw_data[['ID','Q00_PERF_IDA', 'Q00_PERF_EXP', 'Q00_PERF_PRO','Q07_DESC_GRP','Q07_SMLL_QTD','Q07_CARG_CAT', 'Q07_IN
df8 = raw_data[['ID','Q00_PERF_IDA', 'Q00_PERF_EXP', 'Q00_PERF_PRO','Q08_DESC_GRP','Q08_SMLL_QTD','Q08_CARG_CAT', 'Q08_IN

df1.columns = ['ID','PERF_IDA','PERF_EXP','PERF_PRO','DESC_GRP','SMLL_QTD','CARG_CAT', 'INTE_NUM','PERC_BIN']
df2.columns = ['ID','PERF_IDA','PERF_EXP','PERF_PRO','DESC_GRP','SMLL_QTD','CARG_CAT', 'INTE_NUM','PERC_BIN']
df3.columns = ['ID','PERF_IDA','PERF_EXP','PERF_PRO','DESC_GRP','SMLL_QTD','CARG_CAT', 'INTE_NUM','PERC_BIN']
df4.columns = ['ID','PERF_IDA','PERF_EXP','PERF_PRO','DESC_GRP','SMLL_QTD','CARG_CAT', 'INTE_NUM','PERC_BIN']
df5.columns = ['ID','PERF_IDA','PERF_EXP','PERF_PRO','DESC_GRP','SMLL_QTD','CARG_CAT', 'INTE_NUM','PERC_BIN']
df6.columns = ['ID','PERF_IDA','PERF_EXP','PERF_PRO','DESC_GRP','SMLL_QTD','CARG_CAT', 'INTE_NUM','PERC_BIN']
df7.columns = ['ID','PERF_IDA','PERF_EXP','PERF_PRO','DESC_GRP','SMLL_QTD','CARG_CAT', 'INTE_NUM','PERC_BIN']
df8.columns = ['ID','PERF_IDA','PERF_EXP','PERF_PRO','DESC_GRP','SMLL_QTD','CARG_CAT', 'INTE_NUM','PERC_BIN']
```

Teste de Hipótese

Variável resposta é qualitativa ordinal, aplicada em mais de dois grupos com medidas repetidas

Cenário 01

Teste ANOVA de Friedman para Nível Intenção de Refatoração

```
print('Cenário 01: \n[Variável Dependente]: Nível de Intenção de Refatoração e Percepção de Anomalias \n[Variáveis Indepe

#Friedman's ANOVA for non parametric data https://machinelearningmastery.com/nonparametric-statistical-significance-tests
stat, p = friedmanchisquare(raw_data['Q01_INTE_NUM'], raw_data['Q03_INTE_NUM'], raw_data['Q04_INTE_NUM'])

print('Teste ANOVA de Friedman (Nível de Intenção de Refatoração): \nChi²: {:.10f}. p-Value: {:.10f} \n'.format(stat, p))
```

```
Cenário 01:
[Variável Dependente]: Nível de Intenção de Refatoração e Percepção de Anomalias
[Variáveis Independentes] Quantidade de Anomalias: 0 - Métricas de Carga Cognitiva: ND, Alta e Baixa.

Teste ANOVA de Friedman (Nível de Intenção de Refatoração):
Chi²: 4.7605633803. p-Value: 0.0925245106
```

Teste Q de Cochran para a Percepção de Anomalias no Código

```
df_perc_bin = raw_data[['Q01_PERC_BIN', 'Q03_PERC_BIN', 'Q04_PERC_BIN' ]]
cochran_result = cochrans_q(df_perc_bin)
print('Teste Q de Cochran (Percepção de Anomalias): \nQ: {:.10f}. p-Value: {:.10f} \n'.format(cochran_result.statistic, c
```

```
Teste Q de Cochran (Percepção de Anomalias):
Q: 1.4000000000. p-Value: 0.4965853038
```

Cenário 02

Teste ANOVA de Friedman para Nível de Intenção de Refatoração

```
print('Cenário 02: \n[Variável Dependente]: Nível de Intenção de Refatoração  \n[Variáveis Independentes] Quantidade de A

#Friedman's ANOVA for non parametric data https://machinelearningmastery.com/nonparametric-statistical-significance-tests
stat, p = friedmanchisquare(raw_data['Q02_INTE_NUM'], raw_data['Q05_INTE_NUM'], raw_data['Q06_INTE_NUM'])

print('Teste ANOVA de Friedman (Nível de Intenção de Refatoração): \nChi²: {:.10f}. p-Value: {:.10f} \n'.format(stat, p))
```

```
Cenário 02:
[Variável Dependente]: Nível de Intenção de Refatoração
[Variáveis Independentes] Quantidade de Anomalias: 1 - Métricas de Carga Cognitiva: ND, Alta e Baixa.

Teste ANOVA de Friedman (Nível de Intenção de Refatoração):
Chi²: 2.7311827957. p-Value: 0.2552296891
```

Teste Q de Cochran para a Percepção de Anomalias no Código

```
df_perc_bin = raw_data[['Q02_PERC_BIN', 'Q05_PERC_BIN', 'Q06_PERC_BIN' ]]

cochran_result = cochrans_q(df_perc_bin)
print('Teste Q de Cochran (Percepção de Anomalias): \nQ: {:.10f}. p-Value: {:.10f} \n'.format(cochran_result.statistic, c
```

```
Teste Q de Cochran (Percepção de Anomalias):
Q: 1.2857142857. p-Value: 0.5257880244
```

Cenário 03

Teste ANOVA de Friedman para Nível Intenção de Refatoração

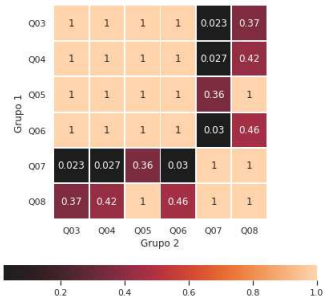
```
print('Cenário 03: \n[Variável Dependente]: Nível de Intenção de Refatoração e Percepção de Anomalias  \n[Variáveis Indep

merged_dfs = pd.concat([df3, df4, df5, df6, df7, df8])

#Friedman's ANOVA for non parametric data https://machinelearningmastery.com/nonparametric-statistical-significance-tests
stat, p = friedmanchisquare(raw_data['Q03_INTE_NUM'], raw_data['Q04_INTE_NUM'], raw_data['Q05_INTE_NUM'], raw_data['Q06_

print('Teste ANOVA de Friedman (Nível de Intenção de Refatoração): \nChi²: {:.10f}. p-Value: {:.10f} \n'.format(stat, p))

print(pairwise_tukeyhsd(merged_dfs['INTE_NUM'], merged_dfs['DESC_GRP'], 0.05)._results_table)
```

```
Cenário 03:
[Variável Dependente]: Nível de Intenção de Refatoração e Percepção de Anomalias
[Variáveis Independentes] Quantidade de Anomalias: 1, 2 e 3 - Métricas de Carga Cognitiva: ND, Alta e Baixa.

Teste ANOVA de Friedman (Nível de Intenção de Refatoração):
Chi²: 30.9403669725. p-Value: 0.0000096247

Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower  upper  reject
-----------------------------------------------------
   Q03    Q04      0.0    0.9 -0.6159 0.6159  False
   Q03    Q05   0.1803    0.9 -0.4356 0.7962  False
   Q03    Q06  -0.0164    0.9 -0.6323 0.5995  False
   Q03    Q07   0.6721 0.0233  0.0562  1.288   True
   Q03    Q08   0.4754 0.2349 -0.1405 1.0913  False
   Q04    Q05   0.1803    0.9 -0.4356 0.7962  False
   Q04    Q06  -0.0164    0.9 -0.6323 0.5995  False
   Q04    Q07   0.6721 0.0233  0.0562  1.288   True
   Q04    Q08   0.4754 0.2349 -0.1405 1.0913  False
   Q05    Q06  -0.1967    0.9 -0.8126 0.4192  False
   Q05    Q07   0.4918 0.2016 -0.1241 1.1077  False
   Q05    Q08   0.2951 0.7162 -0.3208  0.911  False
   Q06    Q07   0.6885 0.0184  0.0726 1.3044   True
   Q06    Q08   0.4918 0.2016 -0.1241 1.1077  False
   Q07    Q08  -0.1967    0.9 -0.8126 0.4192  False
-----------------------------------------------------
```

Teste Q de Cochran para a Percepção de Anomalias no Código

```
df_perc_bin = raw_data[['Q03_PERC_BIN', 'Q04_PERC_BIN', 'Q05_PERC_BIN','Q06_PERC_BIN', 'Q07_PERC_BIN','Q08_PERC_BIN' ]]

cochran_result = cochrans_q(df_perc_bin)
print('Teste Q de Cochran (Percepção de Anomalias): \nQ: {:.10f}. p-Value: {:.10f} \n'.format(cochran_result.statistic, c

merged_dfs = pd.concat([df3, df4, df5, df6, df7, df8])
result = sp.posthoc_dunn(merged_dfs, val_col='INTE_NUM', group_col='DESC_GRP', p_adjust='bonferroni')


plt.figure(figsize=(7,7))
sb.set(font_scale=1)

heatmap = sb.heatmap(result, annot=True, linewidths=1, cbar_kws={"orientation": "horizontal"}, square=True, center=result
heatmap.set_yticklabels(heatmap.get_yticklabels(), rotation=0)
plt.ylabel("Grupo 1")
plt.xlabel("Grupo 2")
```

```
Teste Q de Cochran (Percepção de Anomalias):
Q: 21.2372881356. p-Value: 0.0007305549

Text(0.5, 153.65600000000006, 'Grupo 2')
```



Resultados

Sociodemográfico

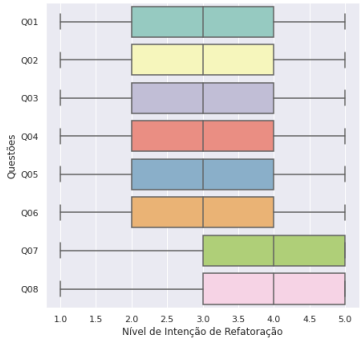
```
s1 = raw_data[["Q00_PERF_IDA", "Q00_PERF_ESC", "Q00_PERF_EXP", "Q00_PERF_PRO"]].groupby(["Q00_PERF_PRO"]).size()
s2 = raw_data[["Q00_PERF_IDA", "Q00_PERF_ESC", "Q00_PERF_EXP", "Q00_PERF_PRO"]].groupby(["Q00_PERF_EXP"]).size()
s3 = raw_data[["Q00_PERF_IDA", "Q00_PERF_ESC", "Q00_PERF_EXP", "Q00_PERF_PRO"]].groupby(["Q00_PERF_ESC"]).size()
s4 = raw_data[["Q00_PERF_IDA", "Q00_PERF_ESC", "Q00_PERF_EXP", "Q00_PERF_PRO"]].groupby(["Q00_PERF_IDA"]).size()

con = pd.concat([s1, s2, s3, s4])
con
```

```
Analista de Sistemas                     6
Analista de Suporte                      3
Analista de qualidade de sistemas        1
Arquiteto de Software                    6
Coordenador de Suporte                   1
Desempregado                             1
Desenvolvedor de Software/Programador    34
Engenheiro de Software                   3
Iniciação Cientifica                     1
Product Owner                            1
QE                                       1
Supote                                   1
Web designer                             1
estudante de ADS                         1
acima de 8 anos                          15
até 2 anos                               18
de 3 a 4 anos                            19
de 5 a 6 anos                            7
de 7 a 8 anos                            2
Cursando ensino superior                 1
Ensino Superior Completo                 9
Ensino Superior Incompleto               42
Ensino Técnico                           2
Especialização/MBA                       4
Mestrado/Doutorado                       3
de 20 a 29 anos                          42
de 30 a 39 anos                          16
de 40 a 49 anos                          2
menos de 20 anos                         1
dtype: int64
```

Estatística Descritiva

```
inte = raw_data[["Q01_INTE_NUM","Q02_INTE_NUM","Q03_INTE_NUM","Q04_INTE_NUM",
                 "Q05_INTE_NUM","Q06_INTE_NUM","Q07_INTE_NUM","Q08_INTE_NUM"]]

inte.columns = ["Q01","Q02","Q03","Q04","Q05","Q06","Q07","Q08"]
plt.figure(figsize=(7,7))
sb.set(font_scale=1)
plt.ylabel("Questões")
plt.xlabel("Nível de Intenção de Refatoração")
sb.boxplot(data=inte, orient="h", palette="Set3")
```

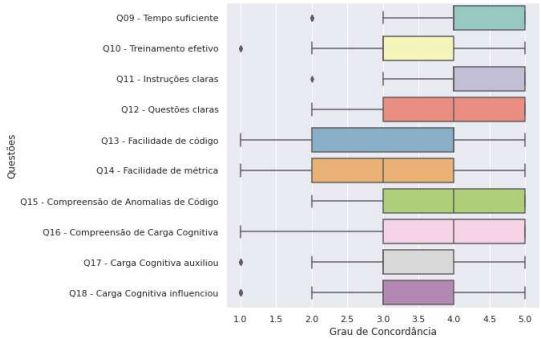
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5dd3159610>
```



Avaliação Qualitativa

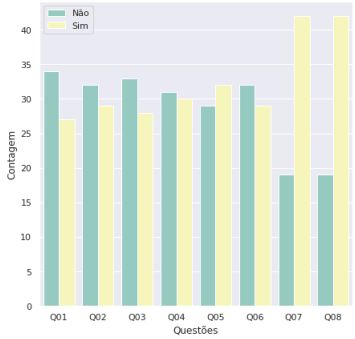
```
quali = raw_data[["CLAREZA_Q01_LVL","CLAREZA_Q02_LVL", "CLAREZA_Q03_LVL", "CLAREZA_Q04_LVL",
                  "PERCEPCAO_Q01_LVL","PERCEPCAO_Q02_LVL",
                  "EFEITOS_Q01_LVL","EFEITOS_Q02_LVL",
                  "USO_Q01_LVL","USO_Q02_LVL"]]

quali.columns = ["Q09 - Tempo suficiente","Q10 - Treinamento efetivo","Q11 - Instruções claras","Q12 - Questões claras",'
                 "Q15 - Compreensão de Anomalias de Código","Q16 - Compreensão de Carga Cognitiva","Q17 - Carga Cognitiva
plt.figure(figsize=(7,7))
sb.set(font_scale=1)

plt.ylabel("Questões")
plt.xlabel("Grau de Concordância")
sb.boxplot(data=quali, orient="h", palette="Set3")

```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5dd2acf7d0>
```




```
merged_dfs = pd.concat([df1, df2, df3, df4, df5, df6, df7, df8])

sb.set(font_scale=1)
plt.figure(figsize=(7,7))

plt.legend(title="Percepção de Anomalias", labels=["Não","Sim"])
plot = sb.countplot(data=merged_dfs, orient="h", palette="Set3", x="DESC_GRP", hue="PERC_BIN")
plot.legend(labels=["Não","Sim"])
plot.set(xlabel="Questões", ylabel="Contagem")
```

```
[Text(0, 0.5, 'Contagem'), Text(0.5, 0, 'Questões')]
```



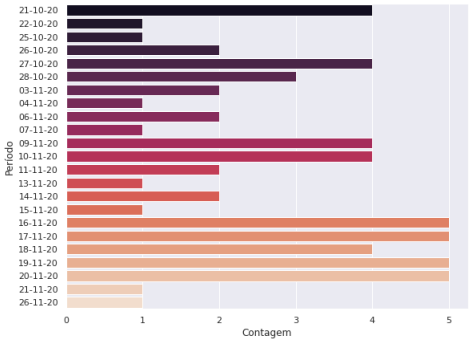

```
s1 = raw_data[["Q00_PERF_EXP", "Q00_PERF_PRO"]].groupby(["Q00_PERF_PRO","Q00_PERF_EXP"]).size()
s1

```

```
Q00_PERF_PRO                          Q00_PERF_EXP
Analista de Sistemas                  até 2 anos          3
                                      de 3 a 4 anos       1
                                      de 5 a 6 anos       2
Analista de Suporte                   de 3 a 4 anos       3
Analista de qualidade de sistemas     acima de 8 anos     1
Arquiteto de Software                 acima de 8 anos     4
                                      de 3 a 4 anos       1
                                      de 7 a 8 anos       1
Coordenador de Suporte                de 3 a 4 anos       1
Desempregado                          de 5 a 6 anos       1
Desenvolvedor de Software/Programador acima de 8 anos     5
                                      até 2 anos          13
                                      de 3 a 4 anos       12
                                      de 5 a 6 anos       3
                                      de 7 a 8 anos       1
Engenheiro de Software                acima de 8 anos     3
```

```
Iniciação Científica                  até 2 anos          1
Product Owner                         acima de 8 anos     1
QE                                    acima de 8 anos     1
Supote                                de 3 a 4 anos       1
Web designer                          de 5 a 6 anos       1
estudante de ADS                      até 2 anos          1
dtype: int64
```

```
d = pd.to_datetime(raw_data["META_DATA"] , format='%d/%m/%Y', dayfirst=True)


sb.set(font_scale=1)
fig, ax = plt.subplots(figsize = (9,7))
fig = sb.countplot(y=d, palette="rocket")
y_dates = d.dt.strftime('%d-%m-%y').unique()
ax.set_yticklabels(labels=y_dates)
fig.set(xlabel="Contagem", ylabel="Período")


```

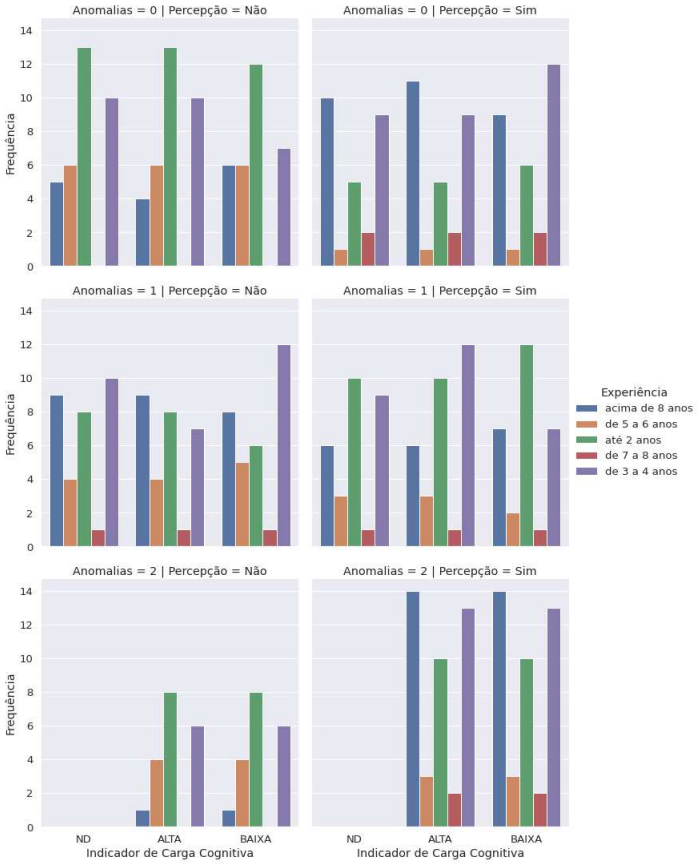
```
[Text(0, 0.5, 'Período'), Text(0.5, 0, 'Contagem')]
```



Sandbox

```
merged_dfs = pd.concat([df1, df2, df3, df4, df5, df6, df7, df8])
sb.set(font_scale=1.2)


fig = sb.catplot(data=merged_dfs, x="CARG_CAT", hue="PERF_EXP" ,col="PERC_BIN", row="SMLL_QTD", kind="count", aspect=1)
fig._legend.set_title("Experiência")

fig.axes[0,0].set_ylabel('Frequência')
fig.axes[1,0].set_ylabel('Frequência')
fig.axes[2,0].set_ylabel('Frequência')

fig.axes[2,0].set_xlabel('Indicador de Carga Cognitiva')
fig.axes[2,1].set_xlabel('Indicador de Carga Cognitiva')


fig.axes[0,0].set_title('Anomalias = 0 | Percepção = Não')
fig.axes[0,1].set_title('Anomalias = 0 | Percepção = Sim')
fig.axes[1,0].set_title('Anomalias = 1 | Percepção = Não')
fig.axes[1,1].set_title('Anomalias = 1 | Percepção = Sim')
fig.axes[2,0].set_title('Anomalias = 2 | Percepção = Não')
fig.axes[2,1].set_title('Anomalias = 2 | Percepção = Sim')

```

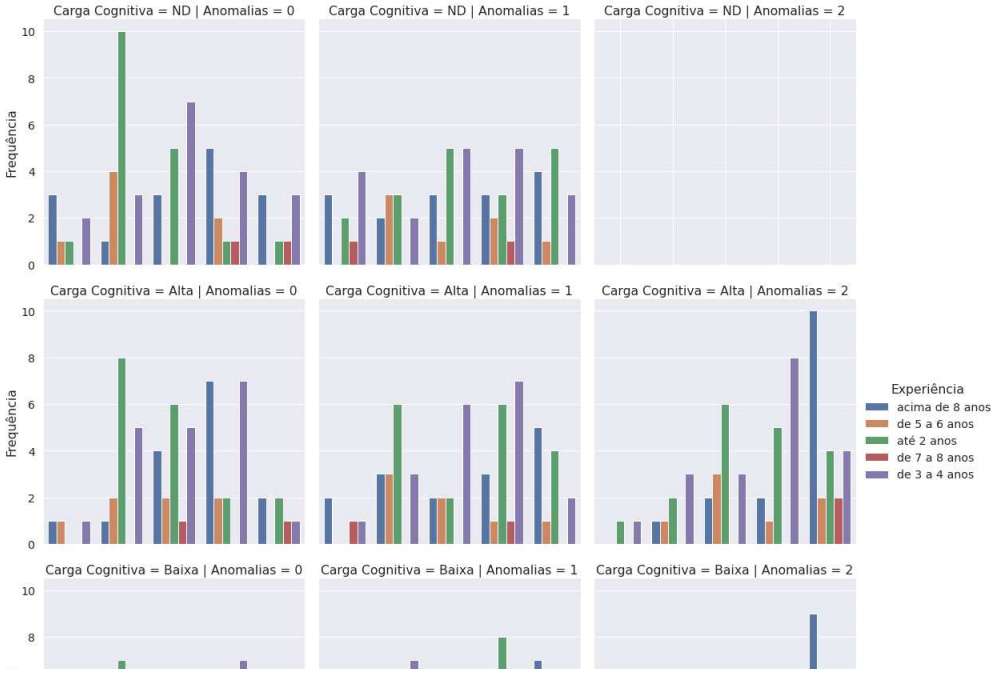
```
Text(0.5, 1.0, 'Anomalias = 2 | Percepção = Sim')
```




```
merged_dfs = pd.concat([df1, df2, df3, df4, df5, df6, df7, df8])
sb.set(font_scale=1.3)
```

```
fig = sb.catplot(data=merged_dfs, x="INTE_NUM", hue="PERF_EXP", col="SMLL_QTD", row="CARG_CAT", kind="count", aspect=1)
fig._legend.set_title("Experiência")


fig.axes[0,0].set_ylabel('Frequência')
fig.axes[1,0].set_ylabel('Frequência')
fig.axes[2,0].set_ylabel('Frequência')

fig.axes[2,0].set_xlabel('Nível de Intenção de Refatoração')
fig.axes[2,1].set_xlabel('Nível de Intenção de Refatoração')
fig.axes[2,2].set_xlabel('Nível de Intenção de Refatoração')



fig.axes[0,0].set_title('Carga Cognitiva = ND | Anomalias = 0')
fig.axes[0,1].set_title('Carga Cognitiva = ND | Anomalias = 1')
fig.axes[0,2].set_title('Carga Cognitiva = ND | Anomalias = 2')

fig.axes[1,0].set_title('Carga Cognitiva = Alta | Anomalias = 0')
fig.axes[1,1].set_title('Carga Cognitiva = Alta | Anomalias = 1')
fig.axes[1,2].set_title('Carga Cognitiva = Alta | Anomalias = 2')

fig.axes[2,0].set_title('Carga Cognitiva = Baixa | Anomalias = 0')
fig.axes[2,1].set_title('Carga Cognitiva = Baixa | Anomalias = 1')
fig.axes[2,2].set_title('Carga Cognitiva = Baixa | Anomalias = 2')

```

```
Text(0.5, 1.0, 'Carga Cognitiva = Baixa | Anomalias = 2')
```



ANEXO A ARTIGOS PUBLICADOS

VIEIRA, R. D.; FARIAS, K. Usage of psychophysiological data as an improvement in the context of software engineering: a systematic mapping study. **ACM International Conference Proceeding Series**, [S.l.], 2020

VIEIRA, R. D.; FARIAS, K. CognIDE: a psychophysiological data integrator approach for visual studio code. **ACM International Conference Proceeding Series**, [S.l.], p. 393–398, 2020