UNIVERSIDADE FEDERAL DO CEARÁ

INSTITUTO UFC VIRTUAL

CURSO DE GRADUAÇÃO EM SISTEMAS E MÍDIAS DIGITAIS

ANA CAROLINA FREIRE FURTADO

ÁTOMOS DE CONFUSÃO: UMA REVISÃO SISTEMÁTICA DA LITERATURA

FORTALEZA

2026

ANA CAROLINA FREIRE FURTADO

ÁTOMOS DE CONFUSÃO: UMA REVISÃO SISTEMÁTICA DA LITERATURA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Orientador: Prof. Dr. Windson Viana de Carvalho

FORTALEZA

2026

ANA CAROLINA FREIRE FURTADO

ÁTOMOS DE CONFUSÃO: UMA REVISÃO SISTEMÁTICA DA LITERATURA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Aprovado em:

BANCA EXAMINADORA

Prof. Dr. Windson Viana de Carvalho (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Leonardo Oliveira Moreira
Univerisdade Federal do Ceará (UFC)

Profa. Dra. Amanda Drielly Pires Venceslau
Univerisdade Federal do Ceará (UFC)

Me. Daniel Mesquita Feijó Rabelo
Univerisdade Federal do Ceará (UFC)

Aos meus pais, por sempre estiveram comigo e deram uma privilegiada base. Mãe, seu cuidado e dedicação foi algo que sempre me deu a força necessária para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinha não importa o que aconteça.

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais, Leila e Raimundo, e meus irmãos, Beatriz e Pedro. Obrigada por todo o incentivo, pelos valores transmitidos e por compreenderem minhas ausências nos momentos em que precisei me dedicar aos estudos e ao trabalho. Tudo o que construí até aqui tem a marca do apoio incondicional de vocês.

Ao meu namorado, Davi, pelo companheirismo diário, pela paciência incansável durante as fases mais tensas deste trabalho e por sempre acreditar na minha capacidade, mesmo quando eu mesma duvidava. Obrigada por ser meu refúgio e por tornar essa caminhada mais leve.

Aos meus amigos de escola - especialmente Isabela, Vinício, Diego e Joana - que me acompanham há tanto tempo. A presença de vocês em minha vida me lembra de onde vim e de quem sou para além da vida acadêmica. Obrigada por continuarem ao meu lado, celebrando cada conquista.

Aos amigos que a faculdade me deu, parceiros de trabalhos e de pingue-pongue. Vocês transformaram a rotina da graduação na UFC em uma jornada compartilhada. Agradeço pelas trocas de conhecimento, pelo apoio mútuo nas disciplinas difíceis e pelas risadas que aliviaram o peso dos semestres.

Estendo meus agradecimentos, também, ao meu orientador Prof. Dr. Windson Viana de Carvalho, por sempre estar presente e me guiar da melhor forma neste trabalho. E, ainda, aos demais membros da banca examinadora, pela disponibilidade em ler e avaliar esta pesquisa. Agradeço pelas contribuições valiosas, pelas críticas construtivas e pelo olhar atento que certamente enriquecerão o resultado final.

Por fim, à Universidade Federal do Ceará e a todos os professores desta instituição, pela excelência do ensino público e pelo espaço de crescimento que me proporcionaram. Sou imensamente grata pelas oportunidades acadêmicas, como as bolsas concedidas ao longo da graduação. O suporte institucional foi vital para minha permanência e permitiu que eu me dedicasse com tranquilidade à minha formação e a este trabalho.

“Tudo o que temos de decidir é o que fazer com o tempo que nos é dado.”

(Gandalf, em *O Senhor dos Anéis* de J.R.R. Tolkien)

RESUMO

A compreensão de código é uma atividade cognitiva fundamental no desenvolvimento de software, impactando diretamente a manutenção e a qualidade dos sistemas. No entanto, certos padrões sintáticos, denominados "Átomos de Confusão" (ACs), tendem a induzir desenvolvedores a erros de interpretação, mesmo sendo tecnicamente corretos. Diante da dispersão do conhecimento sobre o tema, este trabalho apresenta uma Revisão Sistemática da Literatura (RSL) com o objetivo de consolidar o estado da arte sobre os ACs e seus impactos na engenharia de software. A pesquisa foi conduzida com base em um protocolo rigoroso (PICOC), resultando na seleção e análise de 32 estudos primários publicados entre 2017 e 2025. Os resultados indicam que a taxonomia dos átomos evoluiu de um foco inicial em C/C++ para incluir linguagens gerenciadas (Java) e dinâmicas (JavaScript), revelando que a confusão é um fenômeno universal, porém moldado pelo paradigma da linguagem. As evidências empíricas confirmam que a presença de ACs aumenta a carga cognitiva, o tempo de leitura e a propensão a erros, embora estudos recentes em sistemas Java maduros sugiram que são raros os casos em que um átomo resulta em defeitos funcionais. Identificou-se também uma lacuna tecnológica na ausência de ferramentas de detecção em tempo real (*plugins* de IDE) e a emergência da Inteligência Artificial como uma nova fronteira, atuando tanto como agente de refatoração quanto como potencial propagadora de confusão. Conclui-se que a mitigação eficaz dos ACs exige uma abordagem híbrida, combinando educação, barreiras ativas em *pipelines* de desenvolvimento e o uso cauteloso de assistentes de codificação baseados em IA.

Palavras-chave: Átomos de Confusão; Compreensão de Código; Revisão Sistemática; Qualidade de Software; Manutenção de Software.

ABSTRACT

Code comprehension is a fundamental cognitive activity in software development, directly impacting system maintenance and quality. However, certain syntactic patterns, called “Atoms of Confusion” (AoCs), tend to lead developers to misinterpretations, even though they are technically correct. Given the dispersion of knowledge on the subject, this work presents a Systematic Literature Review (SLR) with the aim of consolidating the state of the art on ACs and their impacts on software engineering. The research was conducted based on a rigorous protocol (PICOC), resulting in the selection and analysis of 32 primary studies published between 2017 and 2025. The results indicate that the taxonomy of atoms has evolved from an initial focus on C/C++ to include managed (Java) and dynamic (JavaScript) languages, revealing that confusion is a universal phenomenon, but one shaped by the language paradigm. Empirical evidence confirms that the presence of ACs increases cognitive load, reading time, and the propensity for errors, although recent studies on mature Java systems suggest that cases in which an atom results in functional defects are rare. A technological gap was also identified in the absence of real-time detection tools (IDE plugins) and the emergence of Artificial Intelligence as a new frontier, acting both as a refactoring agent and as a potential propagator of confusion. It is concluded that effective mitigation of AoC requires a hybrid approach, combining education, active barriers in development pipelines, and the cautious use of AI-based coding assistants.

Keywords: Atoms of Confusion; Code Comprehension; Systematic Review; Software Quality; Software Maintenance.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIAÇÕES E ACRÔNIMOS

ACs	Átomos de Confusão
CI/CD	Integração Contínua e Entrega/Implantação Contínua
EEG	Eletroencefalograma
IDE	*Integrated Development Environment*
IOCCC	*International Obfuscated C Code Contest*
LLMs	Large Language Models
PICOC	População, Intervenção, Comparação, Resultado/*Outcome*, Contexto
PRISMA	*Preferred Reporting Items for Systematic Reviews and Meta-Analyses*
QA	*Quality Assessment*
QPs	Questões de Pesquisa
RSL	Revisão Sistemática da Literatura
TCC	Trabalhos de Conclusão de Curso

SUMÁRIO

1 INTRODUÇÃO

A compreensão e manipulação de códigos fonte são habilidades cognitivas essenciais, além do domínio técnico, no processo de desenvolvimento de software (Singh *et al.*, 2024). Entretanto, apesar do avanço das linguagens de programação e ferramentas de suporte, desenvolvedores, especialmente iniciantes, enfrentam obstáculos na leitura e interpretação de partes específicas de código, apesar de estarem sintaticamente corretas (Costa; Gheyi, 2023).

Dessa forma, a habilidade de compreender programas é considerada um elemento importante para que um projeto de software seja bem sucedido, já que a compreensão é um conceito central de desenvolvimento de software. Assim, quando a compreensão de um código é confusa, ela pode afetar na qualidade e, principalmente, na manutenção e revisão de código (Gopstein *et al.*, 2017). Um exemplo de fenômeno que pode contribuir para dificultar a compreensão de código são os chamados Átomos de Confusão (ACs).

Nesse contexto, Gopstein *et al.* (2017) definiu Átomos de Confusão como pequenos trechos de código que possuem alto potencial de induzir mal entendidos no código. O termo recente foi originalmente proposto a partir de experimentos com desenvolvedores utilizando a linguagem C, sendo posteriormente adaptado e expandido para outras linguagens, como Java (Langhout; Aniche, 2021; Pinheiro *et al.*, 2023). Esses átomos consistem em construções que, embora corretas do ponto de vista sintático e funcional, tendem a gerar ambiguidade, servindo de fundamento empírico e quantitativo para o entendimento do que torna o código confuso.

Apesar disso, a literatura relacionada aos Átomos de Confusão, embora crescente, apresenta-se dispersa e heterogênea. Os estudos existentes diferem quanto às linguagens analisadas, aos métodos empíricos empregados, aos tipos de Átomos de Confusão identificados e aos impactos observados, considerando que determinados tipos exercem efeitos mais negativos do que outros (Oliveira *et al.*, 2020). Essa fragmentação dificulta a consolidação do conhecimento sobre o tema e limita tanto a comparação entre resultados quanto a aplicação prática das descobertas.

Diante desse cenário, observa-se a ausência de uma visão consolidada que sintetize de forma sistemática o conhecimento produzido sobre Átomos de Confusão. Em particular, ainda não está claramente estabelecido quais tipos de Átomos de Confusão são mais recorrentes, em quais linguagens de programação predominam, quais impactos exercem sobre a compreensão e manutenção de código e quais estratégias têm sido propostas para mitigar seus efeitos.

Nesse contexto, este trabalho apresenta uma Revisão Sistemática da Literatura (RSL)

sobre o tema que se justifica pela necessidade de consolidar, de forma crítica e estruturada, o conhecimento já produzido sobre os Átomos de Confusão (Kitchenham, 2004; Kitchenham; Charters, 2007). Este tipo de revisão permite identificar evidências empíricas existentes, avaliar a solidez teórica dos estudos, detectar lacunas na literatura e orientar o desenvolvimento de novas pesquisas com base em dados selecionados (Pollock; Berge, 2017).

1.1 Objetivo Central

Considerando a fragmentação do conhecimento existente e a relevância do tema para a Engenharia de Software, este trabalho tem como objetivo central sintetizar o que a literatura científica revela sobre os Átomos de Confusão, investigando seus tipos, contextos de ocorrência, impactos na compreensão e manutenção de código, assim como as abordagens propostas para sua detecção e mitigação.

Apesar da crescente atenção dedicada ao estudo do tema, ainda existem lacunas importantes a serem preenchidas na literatura. Nesse contexto, na literatura, existe a necessidade de respostas claras para perguntas como: quais são os tipos de ACs mais recorrentes nas diferentes linguagens? Como sua presença afeta o processo de desenvolvimento? Quais estratégias têm sido propostas para mitigar seus efeitos?

1.2 Questões de Pesquisa

Quadro 1 – Questões de Pesquisa (QPs) definidas para a Revisão Sistemática.

ID	**Descrição da Questão de Pesquisa**
QP1	Quais tipos de Átomos de Confusão são identificados na literatura e em quais contextos de linguagem de programação eles predominam?
QP2	Quais evidências empíricas reportam o impacto dos Átomos de Confusão na carga cognitiva e na acurácia da compreensão de código por desenvolvedores?
QP3	Qual é a relação observada entre a presença de Átomos de Confusão e a manutenção de software, especificamente em termos de propensão a erros e esforço de alteração?
QP4	Quais abordagens e ferramentas têm sido propostas para a detecção automática ou manual de Átomos de Confusão em bases de código?
QP5	Quais estratégias de refatoração e intervenções têm sido propostas para mitigar os efeitos negativos dos Átomos de Confusão?

Fonte: Elaborado pela autora (2025).

Com o intuito de alcançar o objetivo central, este trabalho se apoia em cinco Questões de Pesquisa (QPs), descritas no Quadro 1, que foram extraídas pelo protocolo da RSL, abordado no próximo capítulo, e que guiaram a extração dos dados.

1.3 Organização da Monografia

O restante deste trabalho está organizado da seguinte forma: o Capítulo 2 apresenta o referencial teórico sobre compreensão de código e a definição de Átomos de Confusão. O Capítulo 3 detalha o protocolo e a condução da revisão sistemática. O Capítulo 4 apresenta os resultados obtidos, incluindo a análise quantitativa e qualitativa dos estudos selecionados. O Capítulo 5 discute esses resultados à luz da literatura e apresenta as limitações do estudo. Por fim, o Capítulo 6 traz as considerações finais e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica que sustenta a pesquisa, organizando os conceitos chave e os trabalhos correlatos essenciais para o entendimento do estudo. O capítulo está dividido em quatro seções principais. Inicialmente, a Seção 2.1 discute a Compreensão de Código, abordando os processos cognitivos envolvidos na leitura de software e os desafios inerentes à interpretação humana de programas. Em seguida, a Seção 2.2 aprofunda-se no conceito de Átomos de Confusão, explorando suas definições, a tipologia dos padrões que dificultam o raciocínio e os impactos que causam na qualidade e manutenção do código. Posteriormente, a Seção 2.3 examina a metodologia de Revisão Sistemática da Literatura, detalhando sua importância para a consolidação científica de fenômenos dispersos na literatura. Por fim, a Seção 2.4 apresenta uma breve conclusão deste capítulo.

2.1 Compreensão de Código

O desenvolvimento de software envolve, além da codificação, um processo contínuo de compreensão do código-fonte, especialmente em sistemas de larga escala que requerem manutenção e evolução constante. A habilidade de compreender o código é essencial para que desenvolvedores possam implementar novas funcionalidades, realizar correções de defeitos e melhorar a legibilidade do sistema (Singh *et al.*, 2024). Essa atividade, conhecida como compreensão de código, exige raciocínio lógico, atenção a detalhes e familiaridade com a linguagem de programação utilizada.

A compreensão do código é um processo passível de influência por muitos fatores, como a experiência prévia do programador, a legibilidade do código e a clareza das estruturas lógicas. Estudos demonstram que o entendimento do código está relacionado à construção de modelos mentais daquilo que o programador está analisando, pois, como demonstrado por Gopstein *et al.* (2017), determinados padrões sintáticos e estruturais, embora corretos do ponto de vista da linguagem de programação, tendem a induzir a interpretações equivocadas, dificultando o entendimento e promovendo potenciais erros durante a leitura e manutenção do software.

Nesse sentido, no contexto brasileiro, Costa & Gheyi (2023) utilizaram o rastreamento ocular (*eye tracking*) para avaliar como programadores novatos compreendem diferentes trechos de código e observaram que padrões visuais podem indicar dificuldades específicas de entendimento em determinados trechos, sugerindo que elementos ambíguos ou com múltiplas

interpretações podem prejudicar a clareza.

A compreensão de código também foi estudada sob a ótica de aspectos cognitivos e emocionais. Singh *et al.* (2024) conduziram uma pesquisa empírica, publicada no *Journal of Computer Languages*[1], para entender como as emoções e estados mentais influenciam a leitura e interpretação de código. Os resultados apontam que fatores como estresse, ansiedade ou frustração diante de trechos confusos podem reduzir a eficiência e aumentar os erros. Tais efeitos são intensificados quando o código possui estruturas sintáticas pouco claras, como as associadas aos ACs, sugerindo uma interseção entre fatores técnicos e psicológicos.

2.2 Átomos de Confusão

O conceito de Átomos de Confusão foi introduzido por Gopstein *et al.* (2017) para descrever a menor unidade de código sintático que, embora tecnicamente válida e executável, induz desenvolvedores ao erro de interpretação. Diferentemente de códigos complexos devido a algoritmos difíceis, os ACs são caracterizados por sua sutileza: eles utilizam regras contra-intuitivas e sutis da linguagem para criar uma discrepância entre a intuição visual (o que o código *parece* fazer) e a execução semântica (e o que o código *realmente* faz). Essa ambiguidade provoca uma sobrecarga cognitiva imediata, tornando o código menos compreensível, difícil de manter e mais propenso a defeitos.

A origem desse conceito partiu de uma análise sistemática de códigos deliberadamente ofuscados. Conforme detalhado no estudo pioneiro de Gopstein *et al.* (2017), os autores examinaram as entradas vencedoras do *International Obfuscated C Code Contest* (IOCCC)[2], uma competição conhecida por premiar programas que violam princípios de clareza explorando as ambiguidades da linguagem C. O processo de identificação seguiu um rigoroso método de "destilação", a qual dois especialistas humanos fragmentaram esses programas complexos até isolarem padrões mínimos e indivisíveis que causassem confusão. Fragmentos que podiam ser reescritos de forma mais clara mantendo o mesmo comportamento (transformação equivalente) foram catalogados como ACs. O Quadro 2[3] exemplifica alguns desses padrões seminais identificados.

A validação de que esses padrões não são apenas uma preferência estética, mas

[1] <https://www.sciencedirect.com/journal/journal-of-computer-languages>

[2] <https://www.ioccc.org/>

[3] Nota: Os nomes dos átomos foram mantidos no original em inglês para preservar a terminologia padrão da taxonomia e evitar imprecisões de tradução.

Quadro 2 – Átomos de Confusão identificados em Gopstein *et al.* (2018)

Nome do Átomo	**Exemplo de Átomo**	**Transformado**
Change of Literal Encoding	`printf(“%d”, 013);`	`printf(“%d”, 11);`

Preprocessor in Statement	`int V1 = 1`
	`#define M1 1`
	`1;`
		`#define M1 1;`
		`int V1 = 1 + 1;`

Macro Operator Precedence	`#define M1 64-1` `2*M1`	`2*64-1`
Assignment as Value	`V1 = V2 = 3;`	`V2 = 3;` `V1 = V2;`
Logic as Control Flow	`V1 && F2();`	`if (V1) F2();`
Post-Increment	`V1 = V2++;`	`V1 = V2;` `V2 += 1;`
Type Conversion	`(double)(3/2)`	`trunc(3.0/2.0)`
Reversed Subscript	`1[“abc”]`	`“ab”[1]`
Conditional Operator	`V2 = (V1==3)?2:V2;`	`if (V1==3) V2=2;`
Operator Precedence	`0 && 1 || 2`	`(0 && 1) || 2`
Comma Operator	`V3 = (V1 += 1, V1)`	`V1 += 1;` `V3 = V1;`
Pre-Increment	`V1 = ++V2;`	`V2 += 1;` `V1 = V2;`
Implicit Predicate	`if (4 % 2)`	`if (4 % 2 != 0)`
Repurposed Variables	`argc = 7;`	`int V1 = 7;`
Omitted Curly Braces	`if(V) F(); G();`	`if(V){F();} G();`

Fonte: Elaborado pela autora (2026) baseado em Gopstein *et al.* (2018).

um fenômeno fisiológico, veio através da neurociência. Yeh *et al.* (2022) conduziram experimentos seminais utilizando Eletroencefalograma (EEG) para monitorar a atividade cerebral de programadores enquanto liam trechos com e sem átomos. Os resultados demonstraram que o processamento de um átomo ativa regiões neurais distintas, associadas à inibição de impulsos e à resolução de conflitos (córtex pré-frontal), indicando que o cérebro precisa “lutar” contra a intuição inicial para entender o código corretamente. Isso confirma que a confusão é um evento mensurável biologicamente, exigindo recursos cognitivos que poderiam estar sendo usados para a lógica do negócio.

Embora a definição tenha nascido no contexto da linguagem C, pesquisas subsequentes demonstraram que o fenômeno é universal, adaptando-se às características de cada tecnologia. Langhout (2020) expandiram o escopo para a linguagem Java, validando empiricamente quais padrões de confusão transcendem a barreira da linguagem e quais são específicos de cada ecossistema. Os autores constataram que, mesmo em um ambiente gerenciado e sem aritmética de ponteiros, certos átomos, como a precedência de operadores lógicos e o uso excessivo de condicionais ternários, elevam significativamente as taxas de erro e o tempo de compreensão. O estudo revelou ainda um dado crítico: os ACs frequentemente induzem a uma “confiança

ilusória", onde o desenvolvedor acredita ter compreendido a lógica corretamente, quando na verdade foi induzido ao erro pela ambiguidade sintática.

Expandindo a fronteira para linguagens dinâmicas, Torres *et al.* (2023) investigaram o ecossistema JavaScript e demonstraram que a confusão assume novas formas na Web. Além dos problemas aritméticos clássicos, eles identificaram que recursos modernos projetados para "facilitar" a codificação, como a Inserção Automática de Ponto e Vírgula (ASI), podem atuar como novos ACs, ocultando a lógica real do programa.

A relevância prática desses átomos vai além da dificuldade de leitura; ela impacta métricas fisiológicas e econômicas. Oliveira *et al.* (2020) utilizaram rastreamento ocular (*eye-tracking*) para provar que a leitura de um átomo exige maior esforço visual e maior número de regressões (releituras) do que um código limpo equivalente. No nível macroscópico de projetos, Pinheiro *et al.* (2023) analisaram a evolução de ACs em repositórios Java de código aberto e observaram uma correlação preocupante: trechos contendo átomos apresentam maior rotatividade de contribuidores e tendem a ser abandonados ou refatorados com mais frequência, sugerindo que a "dívida de legibilidade" pode afetar fortemente a mantenabilidade de longo prazo.

Em síntese, os Átomos de Confusão representam a antítese dos princípios fundamentais de legibilidade e manutenibilidade de software. Enquanto as boas práticas de engenharia priorizam que o código deve servir primariamente como um meio de comunicação claro entre seres humanos, os ACs exploram as eficiência da máquina em detrimento da cognição humana. O estudo sistemático dessas microestruturas torna-se, portanto, um requisito essencial para a garantia da qualidade, fundamentando a criação de ferramentas de análise e diretrizes de revisão capazes de detectar essas ambiguidades sutis, prevenindo que a complexidade acidental se transforme em falhas sistêmicas.

2.3 Revisão Sistemática da Literatura

A Revisão Sistemática da Literatura é uma metodologia estruturada, rigorosa e reprodutível utilizada para identificar, analisar e sintetizar os estudos sobre um determinado tema ou questão de pesquisa presentes na literatura. Diferentemente de uma revisão narrativa, que pode ser mais subjetiva e seletiva, a RSL adota critérios explícitos de busca, inclusão, exclusão e avaliação da qualidade dos estudos (Pollock; Berge, 2017).

Ainda conforme Pollock & Berge (2017), essa metodologia é amplamente aplicada

em áreas da saúde, mas tem ganhado destaque crescente na computação, engenharia de software e ciências sociais aplicadas, especialmente quando se deseja consolidar o estado de uma tecnologia, ferramenta ou conceito. A revisão sistemática também se diferencia por sua preocupação com a minimização de vieses na seleção e interpretação dos estudos, o que a torna mais robusta e confiável.

Na área de Engenharia de Software, a RSL pode ser empregada para identificar práticas de desenvolvimento, evidências empíricas sobre técnicas de programação, impactos de decisões de arquitetura e, como neste trabalho, compreender como um fenômeno específico, os Átomos de Confusão, tem sido abordado pela comunidade científica (Kitchenham, 2004).

Nesse tipo de investigação, uma etapa essencial é a formulação da pergunta de pesquisa, sendo comum o uso da estrutura População, Intervenção, Comparação, Resultado/*Outcome*, Contexto (PICOC) para organizar de forma lógica e precisa os elementos centrais de uma revisão sistemática (Kitchenham; Charters, 2007). Essa estratégia auxilia o pesquisador a delimitar o escopo do estudo, decompondo a questão central em cinco componentes fundamentais:

- **População (P):** refere-se ao grupo alvo da investigação, que pode ser composto por pessoas, artefatos de software ou tipos específicos de aplicações;
- **Intervenção (I):** define a tecnologia, ferramenta, metodologia ou fenômeno específico que será investigado ou avaliado;
- **Comparação (C):** estabelece o elemento de controle ou a tecnologia alternativa com a qual a intervenção será comparada, embora nem todas as revisões exijam uma comparação explícita;
- ***Outcome*/Resultado (O):** determina os fatores de importância que serão medidos para avaliar o sucesso ou impacto da intervenção;
- **Contexto (C):** descreve o ambiente em que o estudo se situa e as restrições operacionais da pesquisa.

A aplicação dos critérios PICOC permite reduzir a probabilidade de viés na seleção dos estudos primários e assegura que a busca bibliográfica recupere resultados alinhados aos objetivos do trabalho.

Entretanto, é importante destacar que existem modelos alternativos de formulação de perguntas, que podem ser mais adequados dependendo do foco e da natureza da pesquisa. Dentre os principais, destaca-se o modelo SPIDER (*Sample*, *Phenomenon of Interest*, *Design*, *Evaluation*, *Research type*), considerado mais apropriado para revisões qualitativas por focar

no fenômeno investigado e no tipo de estudo (Methley *et al.*, 2014). Outra opção relevante é o PEO (*Population*, *Exposure*, *Outcome*), útil em contextos onde o objetivo é analisar a exposição de um grupo a uma determinada condição e seus resultados (Cook; West, 2012). Por fim, há o modelo PICO (*Population*, *Intervention*, *Comparison*, *Outcome*), uma variação simplificada do PICOC que omite o elemento de contexto, sendo frequentemente utilizada quando o ambiente da intervenção não constitui um critério central de análise (Methley *et al.*, 2014).

Assim, a compreensão dos diferentes modelos de formulação de perguntas de pesquisa fornece a base conceitual necessária para a definição do protocolo da revisão sistemática adotada neste trabalho, o qual é detalhado na seção seguinte.

2.4 Síntese do Capítulo

Este capítulo estabeleceu as bases teóricas necessárias para a investigação dos Átomos de Confusão. Discutiu-se inicialmente um processo mais amplo de Compreensão de Código, demonstrando como a ambiguidade prejudica a formação de modelos mentais precisos e afeta fatores psicológicos do desenvolvedor. Em seguida, a natureza dos padrões sintáticos de Átomos de Confusão foram detalhados, diferenciando-os de meras falhas estéticas e destacando sua validação neurofisiológica como obstáculos cognitivos reais. Por fim, fundamentou-se a escolha da Revisão Sistemática da Literatura (RSL) como o método científico adequado para mapear este campo emergente. Com o referencial teórico consolidado, o capítulo seguinte detalhará o percurso metodológico adotado para conduzir esta revisão.

3 METODOLOGIA

Este capítulo descreve o percurso metodológico adotado para investigar o estado da arte dos Átomos de Confusão e seus impactos na engenharia de software. Inicialmente, classifica-se a natureza da pesquisa como qualitativa, exploratória e bibliográfica, caracterizando-a como um estudo secundário. A seguir, detalha-se a aplicação do método de Revisão Sistemática da Literatura, cuja execução fundamenta-se nas diretrizes propostas por (Kitchenham; Charters, 2007).

Para facilitar a compreensão do processo investigativo, o capítulo está estruturado da seguinte forma: a Seção 3.1 estabelece o alinhamento entre o objetivo do estudo e as questões de pesquisa formuladas; a Seção 3.2 mostra a visão geral das etapas da RSL; enquanto, nas Seções 3.3, 3.4 e 3.5, aprofunda-se no detalhamento de cada etapa; e, por fim, a Seção 3.6 apresenta uma síntese do percurso metodológico percorrido.

3.1 Justificativas da Pesquisa

A condução desta revisão é orientada pela seguinte questão norteadora: *"O que a literatura científica revela sobre os Átomos de Confusão e seus impactos na compreensão e manutenção de código?"*. Para responder a esta questão central, foram formuladas cinco questões de pesquisa específicas (apresentadas no Quadro 1), cujas justificativas se desdobram em três eixos de investigação: caracterização, impacto e intervenção.

O primeiro eixo (relacionado à QP1) busca caracterizar a diversidade dos Átomos de Confusão. Pretende-se mapear as definições adotadas na literatura, identificar em quais linguagens de programação esses padrões predominam e analisar a distribuição geográfica dos estudos. Esse mapeamento é fundamental para compreender a taxonomia dos ACs e permitir a comparação entre diferentes ecossistemas tecnológicos.

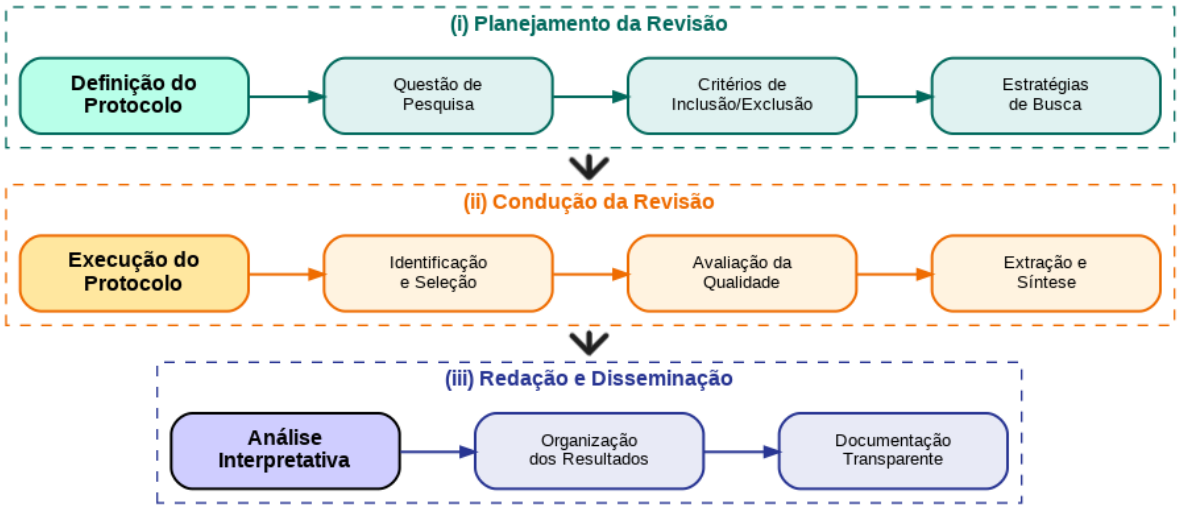
O segundo eixo foca na análise de consequências, desmembrando-se em aspectos cognitivos e técnicos. Sob a ótica da QP2, investiga-se como esses pequenos trechos de código influenciam a performance cognitiva dos desenvolvedores, ou seja, de que forma a ambiguidade afeta a precisão e o tempo de compreensão. Concomitantemente, a QP3 expande essa análise para o ciclo de vida do software, buscando evidências de como os ACs induzem a tomadas de decisão equivocadas e aumentam a propensão a erros, comprometendo a manutenibilidade e a qualidade dos sistemas.

Por fim, o terceiro eixo dedica-se aos métodos de detecção e mitigação de ACs. Isso envolve, por meio da QP4, identificar quais ferramentas e abordagens têm sido utilizadas para rastrear esses padrões em bases de código e, através da QP5, levantar quais soluções de refatoração ou intervenção têm sido propostas para neutralizar os efeitos negativos dos átomos, garantindo assim um panorama completo das contramedidas disponíveis na literatura.

3.2 Etapas da Revisão Sistemática

Esta pesquisa foi conduzida seguindo as diretrizes clássicas para Revisões Sistemáticas em Engenharia de Software, organizando-se em três etapas principais: (i) planejamento da revisão, (ii) condução da revisão e (iii) redação e disseminação dos resultados. A Figura 1 apresenta uma visão geral do fluxo metodológico adotado.

Figura 1 – Fluxo Metodológico das Etapas da Revisão Sistemática.



Fonte: Elaborado pela autora (2026).

A etapa de **planejamento da revisão** compreende a definição do protocolo de pesquisa, incluindo a formulação da pergunta de pesquisa, os critérios de inclusão e exclusão e as estratégias de busca.

A **condução da revisão** refere-se à execução do protocolo estabelecido, envolvendo a identificação, seleção e avaliação da qualidade dos estudos primários, bem como a extração e síntese dos dados.

Por fim, a etapa de **redação e disseminação dos resultados** contempla a análise interpretativa dos achados, a organização dos resultados e a documentação da revisão de forma

sistemática e transparente.

3.3 Planejamento da Revisão

A fase inicial de planejamento estabelece os alicerces metodológicos da Revisão Sistemática. Nesta etapa, define-se o protocolo de revisão, a questão norteadora e os critérios de seleção. O detalhamento destas definições é apresentado nas subseções a seguir.

3.3.1 Formulação da Pergunta de Pesquisa (PICOC)

Para estruturar a investigação de forma lógica e precisa, adotou-se o modelo PICOC. Esta escolha justifica-se pela ampla aceitação do modelo na Engenharia de Software e por sua robustez em capturar os múltiplos aspectos da pesquisa sobre ACs, abrangendo desde os sujeitos (desenvolvedores) até os contextos tecnológicos e seus efeitos práticos. A aplicação dos elementos do acrônimo ao contexto deste trabalho está detalhada no Quadro 3.

Quadro 3 – Definição da estratégia PICOC para a Revisão Sistemática

Elemento	**Descrição**
População (P)	Desenvolvedores de software (de todos os níveis, de novatos a experientes) e os artefatos de software que eles produzem ou mantêm.
Intervenção (I)	A presença de Átomos de Confusão no código-fonte.
Comparação (C)	A ausência de Átomos de Confusão, incluindo códigos que não os possuem originalmente ou códigos refatorados.
Resultado (O)	Impactos na compreensão (ex: tempo, taxa de acerto) e na manutenção (ex: tempo para corrigir bugs, propensão a erros).
Contexto (C)	O processo de desenvolvimento e manutenção de software em estudos empíricos no campo da Engenharia de Software.

Fonte: Elaborado pela autora (2025).

3.3.2 Elaboração do Protocolo de Revisão

O protocolo de revisão foi elaborado para servir como um guia documentado de todo o processo de pesquisa, assegurando a mitigação de vieses comuns em estudos secundários. A adesão a um protocolo pré-definido contribui para o controle de:

- **Viés de seleção**, relacionado à omissão de estudos relevantes, que é controlado por meio de critérios claros de inclusão e exclusão e da busca em múltiplas bases de dados;

- **Viés de aferição**, que ocorre durante a interpretação ou categorização dos dados extraídos, mitigado pelo uso de formulários padronizados e pela avaliação pareada dos estudos;
- **Viés de publicação**, minimizado pela consideração de literatura cinzenta e por estratégias específicas de busca ampliada.

Em consonância com as diretrizes de Kitchenham (2004), este documento norteador detalha os procedimentos de seleção, extração e síntese. Para garantir que apenas estudos pertinentes fossem incluídos no *corpus* final, estabeleceram-se critérios rigorosos de elegibilidade. Os Critérios de Inclusão (CI) e Exclusão (CE) aplicados durante a triagem estão detalhados no Quadro 4.

3.4 Condução da Revisão

Nesta fase, executou-se o protocolo planejado para identificar, selecionar e analisar os estudos primários. A condução foi estruturada em três momentos: (I) a identificação e seleção dos estudos; (II) a avaliação da qualidade metodológica; (III) e a extração e síntese dos dados.

3.4.1 Identificação e Seleção dos Estudos

A estratégia de busca centralizou-se no indexador acadêmico Google Scholar. Embora revisões sistemáticas tradicionalmente priorizem buscas individualizadas em bases proprietárias (como Scopus ou Web of Science), a escolha por tal indexador justifica-se pela natureza emergente e pelo volume ainda restrito do *corpus* literário associado ao termo técnico "Átomos de Confusão", cunhado recentemente por Gopstein *et al.* (2017).

A especificidade do termo “Átomos de Confusão” restringe o universo de pesquisa a uma quantidade gerenciável de publicações, diferentemente de temas clássicos que exigiriam amostragem. Isso viabilizou uma estratégia de busca exaustiva, na qual foi possível inspecionar a totalidade dos registros recuperados. O Google Scholar foi utilizado como ferramenta agregadora para acessar, em uma única interface, registros de bases proprietárias (como IEEE Xplore, ACM Digital Library e Springer Link) e publicações cinzentas. O objetivo foi obter a maior abrangência possível do estado da arte, superando a fragmentação das bases tradicionais para um termo ainda emergente. A condução seguiu duas etapas complementares, Etapa de Precisão e Etapa de Abrangência, detalhadas a seguir

Quadro 4 – Critérios de Inclusão e Exclusão com Justificativas

ID	Descrição do Critério	Justificativa
Critérios de Inclusão		
CI1	**Foco no tópico:** Estudos que abordam "Átomos de Confusão" como tema central da pesquisa.	Garantir que os estudos selecionados respondam diretamente às Questões de Pesquisa (QPs) e não apenas citem o termo tangencialmente.
CI2	**Período:** Estudos publicados entre os anos de 2017 e 2025.	O termo foi cunhado seminalmente por Gopstein et al. em 2017; o recorte cobre desde a origem do conceito até o estado da arte atual.
CI3	**Idioma:** Estudos escritos nos idiomas Inglês ou Português.	O Inglês é a língua franca da computação científica, enquanto o Português permite a inclusão de literatura relevante do contexto nacional.
CI4	**Disponibilidade:** Estudos que possuem acesso irrestrito ao texto completo para leitura e análise.	A leitura integral é um requisito mandatório para a extração de dados e para a avaliação da qualidade metodológica (QA).
Critérios de Exclusão		
CE1	**Não científico:** Artigos de opinião, editoriais, blogs ou materiais sem revisão por pares.	Assegurar o rigor científico e a confiabilidade das evidências primárias, excluindo fontes subjetivas ou informais.
CE2	**Duplicado:** Versões repetidas ou cópias do mesmo estudo em bases de dados diferentes.	Evitar viés estatístico na síntese dos dados e a contagem redundante de resultados.
CE3	**Incompleto:** Resumos simples, slides de apresentações ou pôsteres de eventos.	Documentos que não apresentam detalhes suficientes sobre metodologia e resultados, impossibilitando a avaliação de qualidade.
CE4	**Outro foco:** Estudos com foco exclusivo em temas correlatos sem conexão direta com Átomos de Confusão (ex: *code smells* genéricos).	Delimitar o escopo da revisão para microestruturas de confusão, diferenciando-as de problemas arquiteturais macro ou outros defeitos de código.

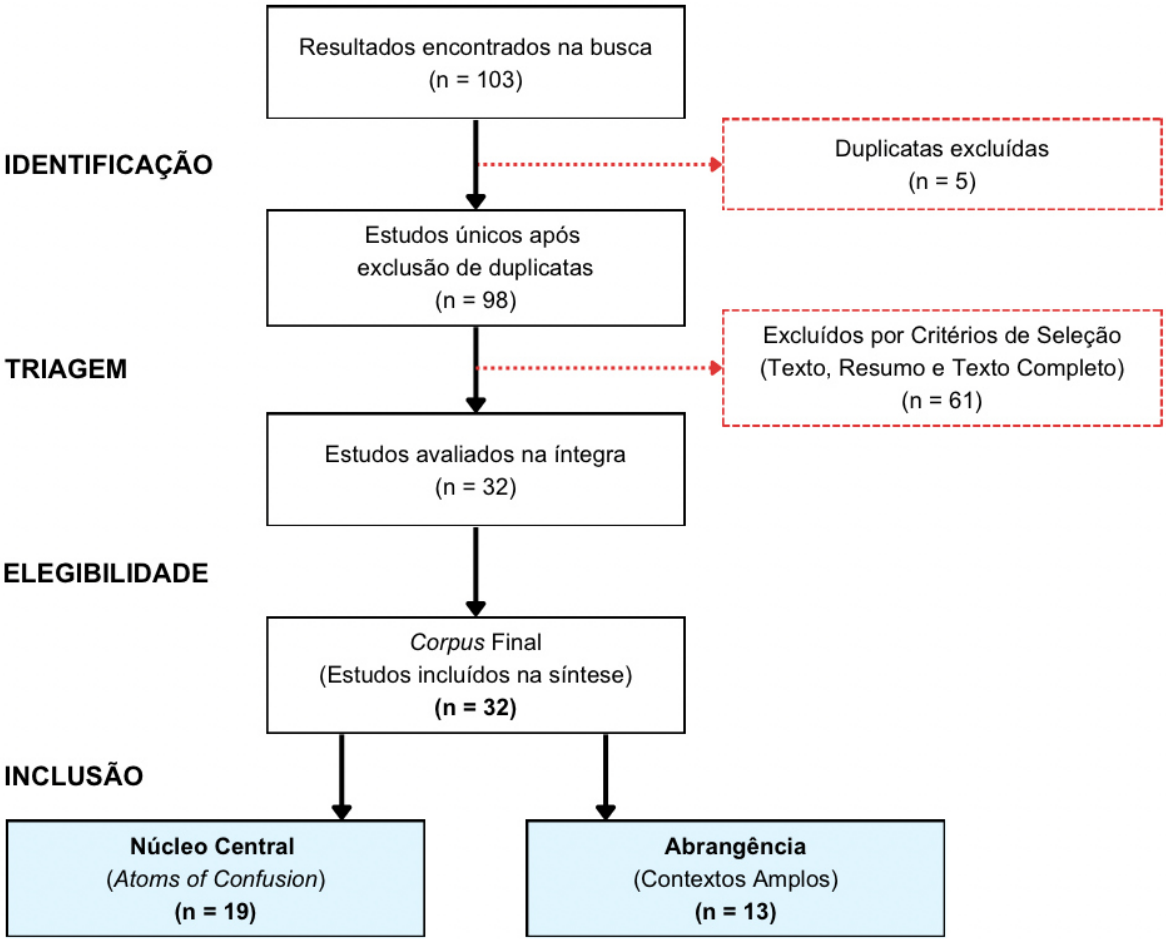
Fonte: Elaborado pela autora (2025).

A execução inicial retornou 103 registros. Após a verificação preliminar e remoção de duplicatas, obteve-se um conjunto bruto de 98 estudos únicos. Estes foram submetidos à triagem multinível (título, resumo e texto completo), aplicando-se os critérios de inclusão e exclusão com revisão independente e resolução de conflitos por um segundo avaliador (orientador), conforme

as boas práticas de Kitchenham (2004).

Para facilitar a visualização do processo de refinamento do *corpus*, a Figura 2 ilustra o fluxograma completo das fases de seleção. O detalhamento operacional de cada etapa, incluindo os protocolos específicos aplicados nas fases de Precisão e Abrangência, encontra-se descrito nas subseções subsequentes.

Figura 2 – Fluxograma PRISMA da Seleção de Estudos.



Fonte: Elaborado pela autora (2026).

3.4.1.1 Etapa de Precisão

Esta etapa priorizou a especificidade, aplicando filtros de busca exclusivamente aos títulos das publicações, em inglês e em português, para identificar o núcleo central de pesquisas sobre o tema. Dos 23 resultados potenciais iniciais, a seleção consolidou-se da seguinte forma:

- Busca em Português: 4 estudos identificados foram mantidos, representando as principais teses e dissertações nacionais sobre o tema (Manor, 2018; Mendes;

Viana, 2021; Campos, 2022). A string de busca foi: *allintitle: "Átomos de Confusão"*; e outras variações.

- Busca em Inglês: Dos 19 estudos identificados, 5 foram excluídos por serem duplicatas ou versões repetidas do mesmo estudo (ex: dissertações substituídas por artigos revisados por pares, como em Pinheiro *et al.* (2023)). Assim, 14 estudos foram mantidos neste grupo. A string de busca foi: *allintitle: "Atoms of Confusion"*; e outras variações.

Dessa forma, o Etapa de Precisão resultou em 18 estudos primários, acrescidos manualmente do trabalho seminal de Gopstein *et al.* (2017), totalizando 19 estudos que representam a espinha dorsal da área.

3.4.1.2 Etapa de Abrangência

Com foco na sensibilidade, esta etapa capturar estudos que investigaram o fenômeno em contextos mais amplos, como compreensão de código (Sugiyama *et al.*, 2025) e cognição (Yeh *et al.*, 2022), sem necessariamente utilizar o termo no título. A busca foi estendida ao **corpo do texto** dos 74 resultados remanescentes da triagem inicial (do total bruto inicial de 98, subtraindo-se os candidatos da etapa de precisão).

A aplicação rigorosa dos critérios de exclusão resultou na eliminação de 61 trabalhos. Os motivos de descarte, variando desde fugas ao tema até formatos de publicação incompletos, estão sintetizados na Tabela 1. Como resultado deste refinamento, 13 estudos complementares foram incorporados.

Tabela 1 – Estudos descartados na fase de seleção

Critério	Qtd.	Descrição
CE4 - Outro Foco	48	Estudos que apenas citam os ACs como referencial ou exemplo, sem análise própria. Inclui também temas distintos sem conexão direta .
CE2 - Duplicado	4	Trabalhos repetidos onde se priorizou a versão final (artigo).
CE3 - Incompleto	7	Publicações como *short papers,* propostas de pesquisa ou índices administrativos de conferências.
CI2 - Período	2	Estudos fora da janela temporal estabelecida ou *pre-prints* de publicações futuras não definitivas.

Fonte: Elaborado pela autora (2025).

3.4.1.3 Consolidação Final

A integração dos resultados das duas etapas consolidou o *corpus* final desta Revisão Sistemática em 32 estudos incluídos. Este conjunto abrange tanto as validações e aplicações contextuais provenientes da Etapa de Precisão e da Etapa de Abrangência.

3.4.2 Avaliação da Qualidade Metodológica

Para assegurar a robustez das evidências, todos os estudos selecionados foram submetidos a uma avaliação de qualidade, do inglês *Quality Assessment* (QA), baseada no *checklist* proposto por Kitchenham *et al.* (2008) para estudos empíricos. Esta ferramenta analítica verificou critérios como a clareza dos objetivos, a validade interna do delineamento experimental, a reprodutibilidade dos métodos e o rigor na análise dos dados, garantindo uma avaliação sistemática e objetiva do estado da arte dos ACs.

Quadro 5 – Checklist Adaptado de Avaliação da Qualidade (QA)

ID	**Critério de Avaliação**
QA1	Os objetivos da pesquisa foram claramente definidos?
QA2	O contexto do estudo (ex: indústria, laboratório, academia) foi descrito adequadamente?
QA3	O desenho da pesquisa foi apropriado para atingir os objetivos propostos?
QA4	A estratégia de coleta de dados foi descrita e justificada?
QA5	A análise dos dados foi rigorosa e apropriada para o tipo de dado coletado?
QA6	A validade e a confiabilidade dos resultados foram discutidas?
QA7	As conclusões são suportadas pelos resultados apresentados?
QA8	O valor e as implicações da pesquisa para a comunidade acadêmica ou industrial foram explicitados?

Fonte: Adaptado de Kitchenham *et al.* (2008).

3.4.2.1 Protocolo de Avaliação

O protocolo avaliou oito dimensões fundamentais da pesquisa empírica (QA1 a QA8), descritas na forma adaptada do *checklist* apresentada no Quadro 5, atribuindo pontuações parciais (0,0; 0,5 ou 1,0 ponto) a cada critério. Para permitir a comparabilidade entre diferentes tipos de estudo, a pontuação cumulativa foi normalizada para uma escala de 0 a 5 pontos, onde 5

representa a excelência metodológica.

3.4.2.2 Panorama Quantitativo dos Resultados

A análise revelou um nível de maturidade científica elevado na literatura sobre Átomos de Confusão, com uma média geral de 4,81 pontos. Devido ao alto rigor observado, 100% dos trabalhos foram classificados no estrato de Alta Qualidade (notas maiores ou iguais a 4,0), indicando que todas as pesquisas selecionadas fornecem evidências confiáveis para a revisão.

A distribuição das pontuações estratifica-se de modo que a excelência plena (Nota 5,0) foi observada em 20 estudos, correspondendo a 60,6% do *corpus*. Este grupo abrange trabalhos seminais, como Gopstein *et al.* (2017), e validações industriais robustas, como Medeiros *et al.* (2019) e Torres *et al.* (2023), que se destacaram pelo uso de desenhos experimentais controlados, estatística inferencial avançada e transparência total dos dados (*open science*).

Quanto aos demais estudos, 13 restantes (39,4% do *corpus*), são classificados de alta qualidade porém com ressalvas (notas entre 4,06 e 4,69). Embora metodologicamente sólidos, estes trabalhos apresentaram limitações pontuais em critérios específicos, frequentemente associadas à natureza exploratória de *short papers* ou Trabalhos de Conclusão de Curso (TCC), a exemplo de Manor (2018) e Campos (2022).

3.4.2.3 Análise dos Critérios

A avaliação individual dos critérios de qualidade (QA) permitiu identificar padrões metodológicos na área, destacando os aspectos já consolidados daqueles que ainda demandam maior rigor científico. As ressalvas metodológicas concentraram-se em três critérios específicos, impactando os 13 estudos que não atingiram a pontuação máxima.

O rigor na análise de dados (QA5), por exemplo, foi infringido parcialmente em 9 trabalhos (27,3%), incluindo Castor (2018) e Shi (2025); a limitação principal consistiu na ausência de testes de hipótese robustos, havendo predominância de conclusões baseadas em estatísticas descritivas ou correlações simples, sem o devido controle de variáveis de confusão, o que reduz a força da inferência causal. De modo similar, a validade e confiabilidade (QA6) apresentaram deficiências em 9 estudos (27,3%), a exemplo de Miranda (2024) e Zhuang *et al.* (2023), devido à discussão insuficiente das ameaças à validade (*threats to validity*), omitindo-se reflexões profundas sobre limitações críticas como tamanho amostral ou vieses de seleção. Por

fim, a estratégia de coleta de dados (QA4) foi apontada como falha em apenas um estudo (3,0%), de Tahsin *et al.* (2023), em razão da falta de justificativa detalhada para os critérios de seleção da amostra minerada.

Em contrapartida, os demais cinco critérios registraram 100% de conformidade em todo o *corpus*, evidenciando a solidez dos fundamentos da área. Todos os 32 estudos definiram com clareza seus objetivos (QA1), contextos (QA2) e desenhos de pesquisa (QA3), beneficiando-se da definição operacional de Átomo de Confusão estabelecida em 2017, o que garantiu coerência e reprodutibilidade mesmo em investigações de menor escopo. Observou-se também consistência integral entre os dados coletados e as conclusões reportadas (QA7), bem como na explicitação das implicações (QA8), demonstrando a preocupação dos autores em conectar os achados práticos ao ensino de programação e à manutenção de software industrial.

3.4.2.4 Síntese da Avaliação

Conclui-se, dessa forma, que o *corpus* desta Revisão Sistemática possui robustez metodológica considerável. A predominância de estudos de excelência (60,6%) e a ausência de trabalhos de baixa qualidade asseguram que as evidências sintetizadas são confiáveis. As limitações identificadas nos demais estudos são de natureza pontual e não invalidam seus resultados, servindo apenas como alerta para que seus achados sejam interpretados como indicativos e não conclusivos.

3.4.3 Extração e Síntese dos Dados

Para assegurar a uniformidade e a rastreabilidade das informações coletadas, a extração de dados foi realizada mediante o preenchimento de uma planilha. A estrutura de extração foi desenhada para atender diretamente aos objetivos da pesquisa, organizando os dados em cinco dimensões analíticas:

- Identificação e Rastreabilidade**:** Colunas destinados ao controle bibliográfico e mapeamento de grupos de pesquisa, incluindo “ID do Artigo”, “Autor, Ano, Título”, “Origem” (país/instituição), “Base de Dados” e “Tipo de Estudo”.
- Caracterização do Fenômeno (QP1): Extração de dados técnicos para mapear o contexto tecnológico e a taxonomia, abrangendo “Linguagem de Programação”, “Tipos de ACs Mapeados” e a “Definição dos ACs” adotada pelos autores.
- Análise de Impacto (QP2 e QP3): Coleta de evidências sobre os efeitos do fenô-

meno, registradas nas colunas “Impactos Relatados” (qualitativos) e “Evidências (Métricas)” (quantitativas).

- Intervenção Técnica (QP4 e QP5): Identificação das ferramentas e estratégias propostas na literatura, documentadas nas colunas “Métodos de Detecção” e “Métodos de Mitigação”.
- Síntese e Qualidade: Colunas para a consolidação dos achados (“Principais Conclusões”) e para o registro da pontuação obtida no *checklist* de rigor metodológico (“Avaliação de Qualidade”).

Após a extração, a síntese dos dados adotou uma abordagem predominantemente qualitativa e descritiva (síntese narrativa). Os achados foram tabulados e categorizados tematicamente para responder às QPs, correlacionando as evidências extraídas com a qualidade metodológica dos estudos, em conformidade com as diretrizes de análise de Kitchenham (2004).

3.5 Redação e Disseminação dos Resultados

A etapa final do processo de revisão sistemática consiste na consolidação e comunicação dos achados. Esta foi moldada para transformar os dados brutos extraídos em conhecimento estruturado, contendo as fases de (i) interpretação dos resultados, (ii) redação e (iii) atualização da revisão.

A análise dos dados ultrapassa a simples descrição estatística, adotando uma abordagem interpretativa crítica. O processo analítico focou na avaliação da força das evidências encontradas, identificando padrões de convergência entre os estudos, bem como contradições que apontam para lacunas ou divergências teóricas. O objetivo central desta interpretação foi traduzir os dados bibliográficos em relevância prática para a Engenharia de Software, utilizando fluxogramas e quadros-síntese para facilitar a visualização das conexões entre os Átomos de Confusão e seus impactos.

A estruturação do relatório final seguiu as diretrizes do *Preferred Reporting Items for Systematic Reviews and Meta-Analyses* (PRISMA), definidas por Moher *et al.* (2009). A adoção deste padrão internacional visa assegurar a transparência, a completude e a reprodutibilidade do relato. O documento resultante inclui o fluxograma detalhado do processo de seleção, apresentado na Figura **??**, e tabelas comparativas das características dos estudos primários, permitindo que outros pesquisadores possam replicar ou auditar o processo.

Para garantir a contemporaneidade dos resultados apresentados, realizou-se um ciclo

final de atualização das buscas imediatamente antes da conclusão da redação. Este procedimento assegurou a incorporação de estudos recentemente publicados ou indexados tardiamente, garantindo que o panorama traçado reflita o estado da arte mais atualizado disponível nas bases de dados consultadas.

3.6 Síntese do Capítulo

O percurso metodológico aqui delineado consolidou a estrutura necessária para investigar o estado da arte dos Átomos de Confusão com rigor científico. Fundamentada nas diretrizes de Kitchenham (2004), a Revisão Sistemática da Literatura foi planejada a partir de uma questão norteadora estruturada pelo modelo PICOC e executada mediante um protocolo estrito de mitigação de vieses.

A condução do estudo em duas etapas complementare, Precisão e Abrangência, permitiu o refinamento de um universo bruto de 98 registros iniciais para um *corpus* final de 32 estudos qualificados. A robustez desta seleção foi assegurada pela aplicação de critérios de inclusão e exclusão multiníveis e pela avaliação da qualidade metodológica dos trabalhos primários.

Por fim, a extração sistemática dos dados, organizada em dimensões analíticas, preparou o terreno para a síntese que será apresentada a seguir. Com a metodologia validada, o capítulo seguinte dedica-se a expor os resultados dessa investigação.

4 RESULTADOS

Este capítulo apresenta a síntese dos dados extraídos a partir da análise sistemática do núcleo de 32 estudos primários selecionados. As informações foram estruturadas para fornecer o embasamento empírico necessário à resolução das cinco Questões de Pesquisa (QPs) definidas no protocolo. A exposição inicia-se com uma análise bibliométrica e demográfica do *corpus* (Seção 4.1), oferecendo uma visão panorâmica do campo. Na sequência, detalham-se os achados específicos sobre a tipologia e evolução dos Átomos de Confusão (Seção 4.2), seus impactos quantitativos na cognição humana (Seção 4.3) e na manutenibilidade do software (Seção 4.4). Posteriormente, o capítulo consolida o levantamento das ferramentas tecnológicas de detecção (Seção 4.5) e das estratégias de mitigação (Seção 4.6) reportadas na literatura. Por fim, apresenta-se uma síntese dos resultados obtidos (Seção 4.7) .

4.1 Visão Geral dos Estudos (Bibliometria)

A análise quantitativa dos 32 estudos permitiu traçar o panorama descritivo da pesquisa sobre os ACs. Esta seção detalha nas subções a seguir: (i) evolução temporal das publicações; (ii) a distribuição geografia das instituições de pesquisa; (iii) e o contexto tecnológico predominante.

4.1.1 Evolução Temporal e Ondas de Pesquisa

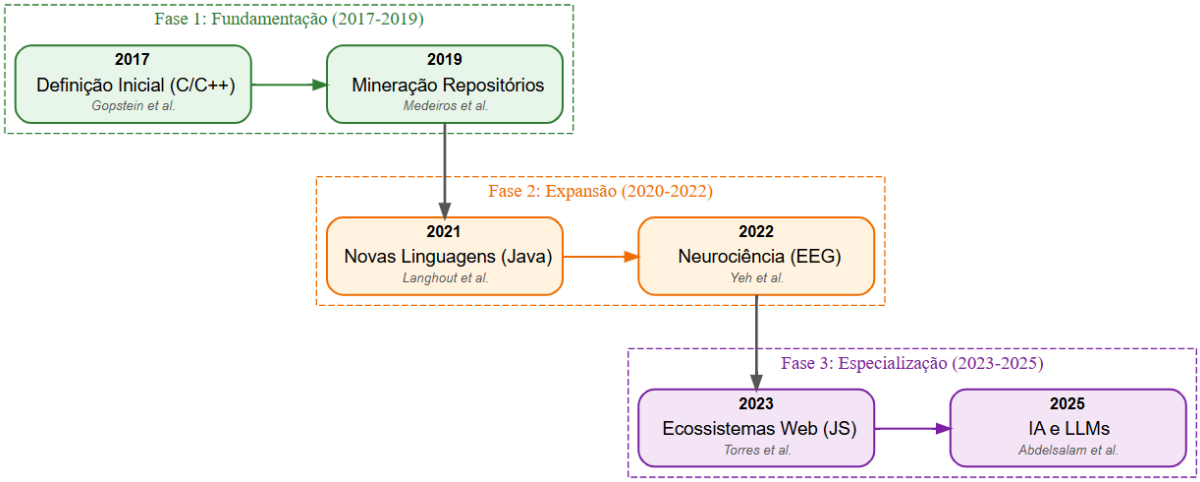
A distribuição cronológica das publicações revela um interesse crescente e sustentado pela comunidade de Engenharia de Software, assim como pode ser visualizado no fluxograma da Figura 3. Tomando como marco inicial o trabalho seminal de 2017, os estudos podem ser agrupados em três períodos distintos conforme o foco da publicação:

1. Período de Fundamentação (2017–2019): Concentra trabalhos focados na definição do catálogo inicial de átomos para C/C++ e nos primeiros experimentos de medição de erro humano (Gopstein *et al.*, 2017). O estudo de Medeiros *et al.* (2019) expandiu a análise para a mineração de grandes repositórios de software *open source*.
2. Período de Expansão (2020–2022): Marcado pela adaptação do conceito para novas linguagens, como Java (Langhout; Aniche, 2021). Estudos de neurociência, como o de Yeh *et al.* (2022), introduziram o uso de EEG para investigar a resposta

neural.

3. Período de Especialização e IA (2023–2025): Fase atual caracterizada por investigações em ecossistemas *web* (JavaScript) (Torres *et al.*, 2023), uso de rastreamento ocular (*eye-tracking*) (Bergum *et al.*, 2024) e a interseção com a Inteligência Artificial, onde trabalhos como o de Abdelsalam *et al.* (2025) investigam a relação entre Grandes Modelos de Linguagem, do inglês Large Language Models (LLMs), e a confusão humana.

Figura 3 – Fluxograma da Evolução Temporal e Ondas de Pesquisa.



Fonte: Elaborado pela autora (2026).

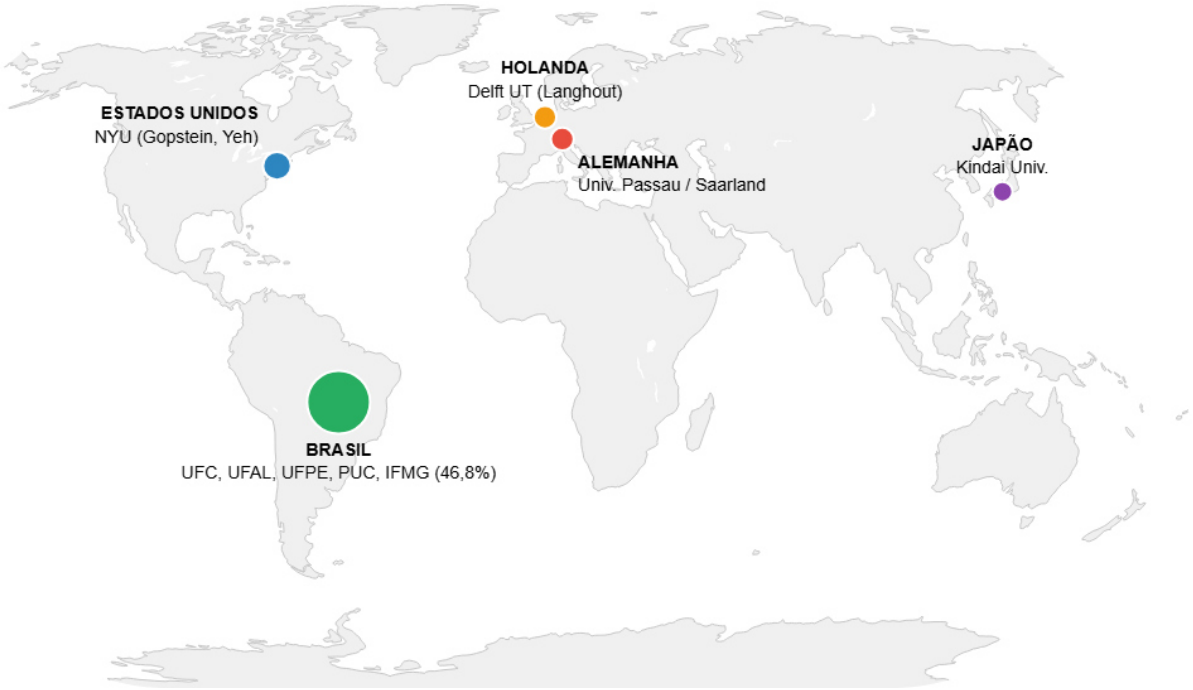
4.1.2 Distribuição Geográfica

A análise das afiliações institucionais permitiu mapear uma rede de colaboração global, caracterizada por polos de especialização distintos na América do Norte, Europa e América do Sul. O *corpus* final evidencia que, embora a liderança numérica de publicações seja dividida entre múltiplos centros, a origem e a evolução do tema Átomos de Confusão estão atreladas a grupos específicos, assim como apresentado na Figura 4.

Os Estados Unidos da América (EUA) figuram como o berço da pesquisa, liderados pela *New York University* (NYU). Foi neste centro que Gopstein *et al.* (2017) definiram o conceito original e onde Yeh *et al.* (2017) conduziram os experimentos seminais de neuroimagem (EEG).

Agora, no continente europeu, destaca-se o protagonismo da *Delft University of Technology* (*TU Delft*), na Holanda. Pesquisadores vinculados a esta instituição, como Langhout & Aniche (2021) e Shi (2025), foram responsáveis por transpor a teoria para o ecossistema Java

Figura 4 – Distribuição Geográfica dos principais grupos de pesquisa.



Fonte: Elaborado pela autora (2026).

e investigar a refatoração automática. Observa-se também a relevância da *University of Passau* (Alemanha), cujos pesquisadores colaboraram com grupos brasileiros na criação de *parsers* robustos para código C (Medeiros *et al.*, 2019), e da *Saarland University*, com contribuições inovadoras focadas na validação multimodal com *eye-tracking* (Bergum *et al.*, 2024).

Já o Brasil consolidou-se como um polo de aplicação prática e estudos de larga escala. Cerca de 46,8% do *corpus* selecionado (15 estudos) possui liderança ou coautoria de pesquisadores vinculados a instituições brasileiras, com destaque para a região Nordeste. Diferentemente do foco teórico inicial dos EUA, a pesquisa nacional diversificou-se em:

- Mineração de Repositórios (MSR): liderada pela Universidade Federal do Ceará (UFC), com foco inédito em ecossistemas móveis (Android) e impacto em CI/CD (Tabosa; Rocha, 2024; Feijó *et al.*, 2023).
- Cognição e Métricas: a Universidade Federal de Alagoas (UFAL) foi pioneira na replicação de estudos cognitivos e uso de *eye-tracking* no hemisfério sul (Oliveira *et al.*, 2020).
- Ensino e Novas Linguagens: a Universidade Federal de Pernambuco (UFPE) expandiram a investigação para a percepção de novatos e linguagens, como Swift

(Manor, 2018).

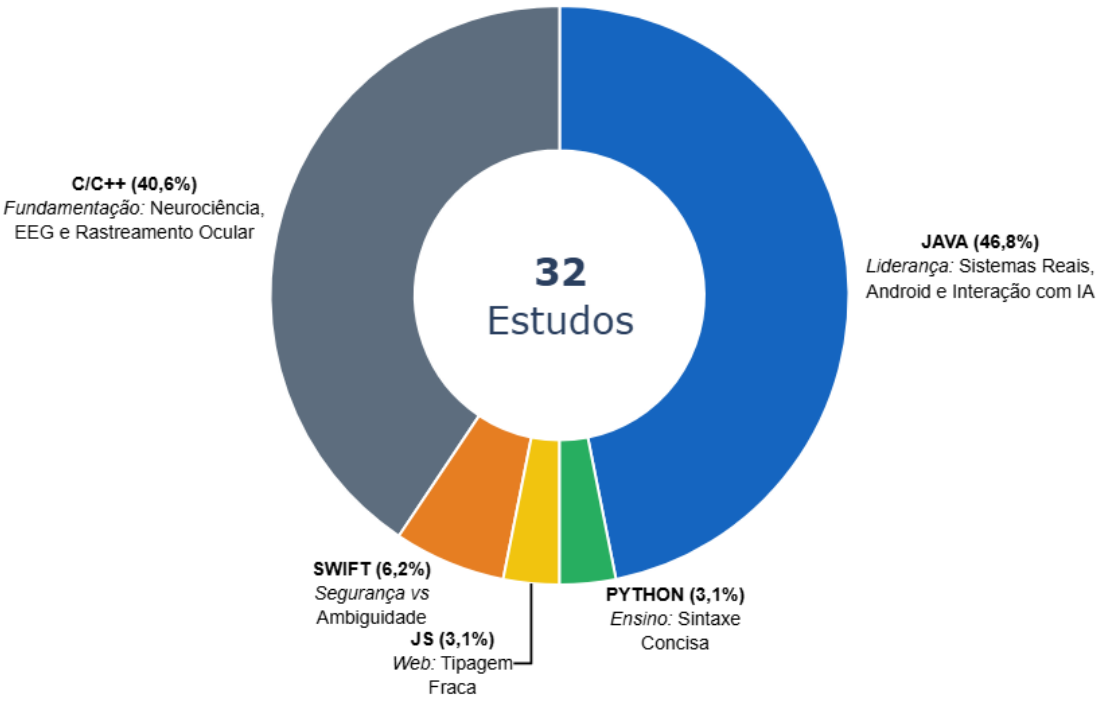
- Fator Humano e IA: instituições de Minas Gerais (PUC-Minas e IFMG) têm liderado as investigações recentes sobre a influência da experiência profissional e a interação com IA Generativa (Campos, 2022; Andrade; Silva, 2025).

Por fim, identificou-se uma participação emergente da Ásia, pontualmente representada pela *Kindai University* (Japão), com o trabalho de Sugiyama *et al.* (2025) sobre modelos de tomada de decisão, demonstrando que o interesse pelo tema alcançou escala global.

4.1.3 Contexto Tecnológico e Linguagens

A análise das linguagens de programação abordadas nos 32 estudos selecionados revela uma mudança de paradigma na literatura. Se inicialmente a pesquisa sobre ACs era exclusivamente associada à gestão manual de memória do C, o *corpus* atual demonstra que a investigação migrou massivamente para ecossistemas gerenciados e orientados a objetos. A Figura 5 ilustra essa nova distribuição.

Figura 5 – Linguagens de programação predominantes nos estudos selecionados.



Fonte: Elaborado pela autora (2026).

A linguagem C/C++, berço da definição original dos átomos, ocupa a segunda posição com cerca de 40,6% das publicações (13 estudos). Esta categoria concentra os estudos fundamentais e experimentais. Devido à sua flexibilidade sintática, o C continua sendo a escolha preferencial para investigações de neurociência e psicologia cognitiva, conforme observado nos trabalhos seminais (Gopstein *et al.*, 2017; Gopstein; Apel, 2020) e nas validações fisiológicas recentes utilizando EEG e rastreamento ocular (Yeh *et al.*, 2022; Bergum *et al.*, 2024).

Além disso, a linguagem Java assume a liderança quantitativa, representando cerca de 46,8% do corpus analisado (15 estudos). Esse volume expressivo reflete o papel da comunidade de pesquisa brasileira em investigar sistemas reais e de longa duração. Enquanto Langhout & Aniche (2021) estabeleceram a base taxonômica para a linguagem, uma série de trabalhos brasileiros expandiu a análise para contextos industriais críticos, como o desenvolvimento Android (Tabosa; Rocha, 2024) e a integração contínua (Feijó *et al.*, 2023; Feijó *et al.*, 2024). Adicionalmente, estudos recentes começam a utilizar Java como base para testar a interação com Inteligência Artificial (Miranda, 2024; Andrade; Silva, 2025).

As demais linguagens (JavaScript, Python e Swift) representam nichos específicos que, embora quantitativamente menores no *corpus* (3,1% cada, com exceção da linguagem Swift, que possui 6,2%), são qualitativamente vitais para comprovar a universalidade do fenômeno. No contexto do desenvolvimento Web, Torres *et al.* (2023) demonstraram que a tipagem fraca e o dinamismo do JavaScript reiteram padrões de confusão. Paralelamente, Costa *et al.* (2023) investigaram o ecossistema Python, revelando como a sintaxe concisa, frequentemente elogiada no ensino, pode impactar negativamente a compreensão de programadores novatos.

Por fim, a análise da linguagem Swift, também, recebe destaque (Manor, 2018; Castor, 2018). Castor (2018) propôs uma definição estruturada para o conceito de ACs, estabelecendo que um átomo deve ser precisamente identificável, propenso a causar confusão, indivisível e substituível por uma alternativa mais clara. O estudo apresentou um catálogo preliminar para a linguagem, identificando que construções consideradas pouco conhecidas ou menos comuns, como o Operador Condicional e a Atribuição como Valor, persistem como fontes de ambiguidade mesmo em um ecossistema moderno projetado com foco em segurança.

4.2 Dados sobre a Caracterização dos Átomos (QP1)

Para atender à primeira Questão de Pesquisa (QP1), foram extraídos os dados referentes à tipologia e taxonomia dos Átomos de Confusão. Os estudos indicam que a taxonomia

original de C (Gopstein *et al.*, 2017) foi utilizada como base e adaptada para linguagens gerenciadas (Langhout; Aniche, 2021) e dinâmicas (Torres *et al.*, 2023).

Os dados mostram a persistência de certos padrões através das tecnologias. Átomos como Pós-Incremento e Atribuição como Valor foram identificados e validados experimentalmente em C (Gopstein *et al.*, 2017), Java (Langhout; Aniche, 2021) e Swift (Castor, 2018).

Em contrapartida, foram catalogados átomos específicos por ecossistema. Para Java, Langhout & Aniche (2021) testaram o Operador Condicional (Ternário) e a Precedência de Operadores, reportando taxas de erro distintas das observadas em C. Para JavaScript, Torres *et al.* (2023) catalogaram a Inserção Automática de Ponto e Vírgula (ASI) e a Desestruturação de Objetos como novas fontes de confusão derivadas do dinamismo da linguagem.

4.3 Métricas de Impacto na Compreensão (QP2)

Em relação à segunda Questão de Pesquisa (QP2), foram extraídas métricas quantitativas e observações qualitativas de experimentos controlados sobre o desempenho dos desenvolvedores. Os dados reportados categorizam-se em acurácia, esforço cognitivo e confiança.

4.3.1 Métricas de Acurácia

Os estudos reportam taxas de erro comparativas entre códigos com e sem átomos:

- Em C: Gopstein *et al.* (2017) mediram que a remoção de átomos aumentou a probabilidade de respostas corretas em até 60%.
- Em Java: Langhout & Aniche (2021) identificaram que a taxa de erro na previsão da saída de programas com átomos foi até 3,8 vezes maior do que nas versões refatoradas.
- Em JavaScript: Torres *et al.* (2023) calcularam *odds ratios* que indicaram uma probabilidade estatisticamente baixa de interpretação correta em trechos contendo o Operador Vírgula.

4.3.2 Métricas de Esforço e Carga Cognitiva

Estudos fundamentados em instrumentação psicofisiológica forneceram dados objetivos sobre o custo biológico do processamento de código confuso:

- Rastreamento Ocular (*eye-tracking*): Oliveira *et al.* (2020) mediram um aumento

de 43% no tempo necessário para compreensão e de 36% no esforço visual total (fixações) em códigos com átomos, além de frequentes movimentos de regressão.

- Eletroencefalograma (EEG): Yeh *et al.* (2022) e Bergum *et al.* (2024) registraram ativação diferenciada em regiões cerebrais associadas à resolução de conflitos e memória de trabalho (córtex pré-frontal) durante a leitura de átomos.

4.3.3 Dados sobre Confiança

Gopstein & Apel (2020), utilizando protocolos verbais (*Thinking Aloud*), documentaram que os participantes frequentemente expressavam altos níveis de confiança em suas respostas, mesmo quando estas estavam incorretas, indicando uma discrepância entre a percepção subjetiva de entendimento e a acurácia real.

4.4 Dados sobre Impacto na Manutenção (QP3)

Para a terceira Questão de Pesquisa (QP3), foram compilados dados de estudos de Mineração de Repositórios de Software (MSR) que correlacionaram a presença de átomos com defeitos e atividades de manutenção.

Estudos em linguagens de sistema reportaram correlações positivas.Gopstein *et al.* (2018) e Medeiros *et al.* (2019) analisaram históricos de versionamento (incluindo o *kernel* Linux) e identificaram que arquivos contendo átomos apareceram com maior frequência em *commits* corretivos (*bug fixes*) do que arquivos livres de átomos.

Em contrapartida, estudos em ecossistemas gerenciados apresentaram dados divergentes. Shi *et al.* (2024), analisando 76 mil *Pull Requests* em Java, reportaram que a presença de ACs não demonstrou correlação estatisticamente significativa com a propensão a defeitos no conjunto de dados analisado. Especificamente para a Precedência de Operadores e o Operador Condicional, a remoção desses padrões não esteve associada à resolução de *bugs*. Pinheiro *et al.* (2023) observaram que, embora alguns átomos sejam removidos ao longo do tempo (rotatividade), a causalidade direta com falhas foi menos evidente.

4.5 Ferramentas de Detecção Identificadas (QP4)

A extração de dados para a quarta Questão de Pesquisa (QP4) resultou no inventário das ferramentas e abordagens tecnológicas propostas para a identificação automática de ACs. As

soluções reportadas dividem-se em:

- Análise Estática (AST): Gopstein *et al.* (2018) e Medeiros *et al.* (2019) desenvolveram analisadores baseados em *srcML* e na infraestrutura do GCC para C. Para Java, Mendes & Viana (2021) apresentou a ferramenta *BOHR*, e Langhout & Aniche (2021) desenvolveram *scripts* dedicados baseados em *JavaParser*.
- Linguagens de Consulta: Torres et al. (2023) utilizaram o *CodeQL* (GitHub) para modelar átomos em JavaScript como consultas de banco de dados, permitindo a varredura em larga escala.
- Modelos de Linguagem (LLMs): Andrade & Silva (2025) e Miranda (2024) realizaram experimentos utilizando modelos como Gemini, ChatGPT e Bard para detecção de átomos, reportando a capacidade dessas ferramentas de identificar padrões, mas também a ocorrência de alucinações.

A revisão dos estudos não retornou registros de *plugins* de Ambiente de Desenvolvimento Integrado, do inglês *Integrated Development Environment* (IDE), disponíveis publicamente para detecção em tempo real nos trabalhos analisados.

4.6 Estratégias de Mitigação (QP5)

Para a quinta Questão de Pesquisa (QP5), foram catalogadas as estratégias de intervenção descritas nos estudos primários, as quais se dividem entre abordagens manuais, processuais e automatizadas. A técnica predominante na literatura é a "Transformação de Remoção de Átomo" (Gopstein *et al.*, 2017), uma refatoração manual que visa explicitar a lógica do código, por exemplo, através da adição de parênteses ou expansão de operadores ternários, sem alterar seu comportamento funcional. A eficácia dessa abordagem foi quantificada por Oliveira *et al.* (2020), que mediram uma redução de 36% na carga cognitiva visual dos desenvolvedores após a aplicação dessas transformações.

No que tange à mitigação via processos e ferramentas, os dados sugerem que a modernização da infraestrutura, isoladamente, é insuficiente. Feijó *et al.* (2023) e Feijó *et al.* (2024) demonstraram que a adoção de esteiras de Integração Contínua e Entrega/Implantação Contínua (CI/CD) e práticas de DevOps não resultou em redução significativa na densidade de átomos. Adicionalmente, Bogachenkova *et al.* (2022) analisaram *Code Reviews* e constataram a escassez de discussões explícitas sobre átomos, sugerindo uma "cegueira intencional" dos revisores humanos, que tendem a focar na funcionalidade em detrimento da legibilidade sintática.

Por fim, a fronteira mais recente da pesquisa envolve a refatoração assistida por Inteligência Artificial. Shi (2025) propôs o uso de LLMs como agentes de refatoração adaptativos, capazes de ajustar a limpeza do código ao nível de experiência do desenvolvedor. Contudo, dados complementares de Shi *et al.* (2024) alertam contra abordagens dogmáticas, indicando que a remoção indiscriminada de certos tipos de átomos em Java não gerou ganhos mensuráveis na redução de defeitos, sugerindo a necessidade de intervenções mais seletivas.

4.7 Síntese do Capítulo

Este capítulo apresentou os dados extraídos dos 32 estudos primários selecionados, organizados para responder às cinco Questões de Pesquisa. A análise bibliométrica evidenciou a evolução temporal do tema e a expansão do interesse para ecossistemas modernos como Java e JavaScript. Os resultados da QP1 e QP2 confirmaram a universalidade do impacto cognitivo dos ACs, enquanto a QP3 revelou nuances importantes sobre a correlação com defeitos, que varia conforme a linguagem. As QPs 4 e 5 mapearam as ferramentas e estratégias disponíveis, destacando a lacuna de soluções em tempo real e a emergência da IA como agente de mitigação. Estes achados formam a base empírica para a discussão aprofundada que será realizada no capítulo a seguir.

5 DISCUSSÃO

Este capítulo dedica-se à interpretação crítica e contextualizada dos achados apresentados na seção de resultados. A análise integrada dos 32 estudos selecionados busca transcender a descrição bibliométrica para compreender a natureza intrínseca e os efeitos dos Átomos de Confusão na Engenharia de Software. Para estruturar essa discussão, o capítulo organiza-se em quatro eixos fundamentais. Inicialmente, as evidências são confrontadas e discutidas individualmente para cada uma das cinco Questões de Pesquisa (Seções 5.1 a 5.5). Na sequência, expande-se o debate para as implicações teóricas transversais emergentes da investigação (Seção 5.6) e avaliam-se as limitações e ameaças à validade que permeiam a condução deste estudo (Seção 5.7). Por fim, apresenta-se uma síntese consolidadora do capítulo (Seção 5.8).

5.1 QP1 - Quais tipos de Átomos de Confusão são identificados na literatura e em quais contextos de linguagem de programação eles predominam?

A primeira questão buscou identificar a tipologia dos Átomos de Confusão e seus contextos de predominância. A discussão dos achados revela que a taxonomia dos ACs não é estática, mas evolui em paralelo à complexidade das linguagens de programação.

Observa-se uma dicotomia clara na literatura. De um lado, identificam-se padrões sintáticos persistentes, oriundos da herança direta da linguagem C, compostos por operadores aritméticos e de ponteiros (Gopstein *et al.*, 2017). Esses padrões provaram ser universalmente nocivos, manifestando-se até mesmo em linguagens modernas, como Swift (Castor, 2018). Contudo, a transição para linguagens gerenciadas (Java) e dinâmicas (JavaScript) introduziu uma nova classe de átomos: os "átomos de comportamento implícito". Em contraste com a confusão visual explícita do C, o estudo de (Torres *et al.*, 2023) sugere que, na Web, a confusão nasce daquilo que o desenvolvedor *não* vê, como a Inserção Automática de Ponto e Vírgula (ASI).

Essa mudança de paradigma impõe uma revisão crítica sobre os catálogos tradicionais de *Code Smells*. Embora os Átomos de Confusão possam ser classificados como uma subcategoria destes, especificamente focados em microestruturas que prejudicam a compreensão imediata, a literatura indica que sua definição não pode ser dogmática. Diferentemente de *smells* arquiteturais clássicos que são agnósticos à linguagem (como acoplamento excessivo), a nocividade de um átomo é dependente do ecossistema. O que é classificado como um átomo em C (e.g., Operador Ternário) pode ser considerado uma prática idiomática aceitável em Java (Shi

et al., 2024). Portanto, listas estáticas de más práticas tornam-se obsoletas se não considerarem as armadilhas cognitivas específicas de cada linguagem.

5.2 QP2 - Quais evidências empíricas reportam o impacto dos Átomos de Confusão na carga cognitiva e na acurácia da compreensão de código por desenvolvedores?

A segunda questão investigou o impacto na carga cognitiva e na acurácia. O debate central aqui reside na validação fisiológica do erro. Os estudos analisados superaram a noção de que a legibilidade é subjetiva. As evidências de neuroimagem (Yeh *et al.*, 2022) e rastreamento ocular (Bergum *et al.*, 2024) confirmam que o cérebro humano é biologicamente ineficiente ao processar átomos, ativando mecanismos de inibição de resposta que consomem recursos mentais escassos.

Um ponto crítico de discussão emerge do trabalho de Gopstein & Apel (2020): a "confiança ilusória". A literatura sugere que o maior perigo dos ACs não é a dificuldade óbvia, mas a facilidade aparente. Diferente de um algoritmo complexo que incita cautela, o átomo engana a intuição, levando o desenvolvedor a cometer erros com alta convicção. Isso altera a forma como o ensino de programação deve ser abordado: não basta ensinar a sintaxe correta; é necessário treinar os desenvolvedores a desconfiarem de suas primeiras impressões visuais.

5.3 QP3 - Qual é a relação observada entre a presença de Átomos de Confusão e a manutenção de software, especificamente em termos de propensão a erros e esforço de alteração?

A terceira questão, focada na relação entre ACs e defeitos, suscita uma das discussões mais ricas e controversas desta revisão. Identificou-se uma divergência significativa entre a literatura inicial (2017–2019), focada em ambientes C/C++, na qual a ambiguidade visual frequentemente resulta em falhas críticas de gerenciamento de memória, e os estudos recentes em ecossistemas gerenciados (Java), que pintam um cenário mais nuançado.

Os achados de Shi *et al.* (2024), fundamentados na mineração massiva de 76 mil *Pull Requests*, desafiam a diretriz da "limpeza total". Ao constatarem que a remoção de átomos nem sempre se correlaciona com a redução de defeitos, sugere-se que o impacto do átomo na manutenibilidade é fortemente mediado pelo *tooling* (ferramental).

Em ambientes de baixo nível, a confusão sintática tende a traduzir-se diretamente

em falhas de execução. Já em ambientes gerenciados e fortemente tipados, como em Java ou Swift (Castor, 2018), compiladores rígidos e IDEs robustas atuam como uma rede de segurança, impedindo que a confusão cognitiva se materialize em erro lógico. Portanto, a discussão aponta que, em projetos modernos, os Átomos de Confusão atuam majoritariamente como uma "dívida técnica de legibilidade", comprometendo a compreensão e o *onboarding* de novos membros, mas não necessariamente como um passivo imediato de confiabilidade funcional.

5.4 QP4 - Quais abordagens e ferramentas têm sido propostas para a detecção automática ou manual de Átomos de Confusão em bases de código?

A quarta questão buscou mapear as ferramentas de detecção e a análise crítica revela um descompasso entre a maturidade acadêmica e a aplicação industrial. Embora a academia tenha produzido soluções robustas para análise científica, como o BOHR (Mendes; Viana, 2021) e consultas CodeQL (Torres *et al.*, 2023), estas operam predominantemente de forma *post-mortem* (análise de dados históricos).

A discussão evidencia uma lacuna tecnológica crítica: a ausência de ferramentas de intervenção em tempo real. Houveram estudos que foram determinantes para demonstrar que a adoção passiva de esteiras de CI/CD não mitiga o problema (Feijó *et al.*, 2023; Feijó *et al.*, 2024). Somado a isso, a "confusão silenciosa" em revisões de código, relatada por Bogachenkova *et al.* (2022), reforça que o olho humano — mesmo em processos de revisão por pares — falha em detectar esses padrões sutis devido à cegueira inatencional.

Conclui-se que, para a detecção ser efetiva, ela precisa migrar dos servidores de análise para o IDE. A literatura de psicologia cognitiva sugere que o momento ideal para intervir na "falha de intuição" é durante a escrita (via plugins bloqueantes). A estratégia atual de detecção tardia atua apenas quando o custo cognitivo de refatoração já é substancialmente mais alto.

5.5 QP5 - Quais estratégias de refatoração e intervenções têm sido propostas para mitigar os efeitos negativos dos Átomos de Confusão?

A quinta questão abordou as estratégias de mitigação. O debate contrapõe a eficácia da refatoração manual à escalabilidade da automação. Embora a remoção manual seja eficaz para a compreensão (Oliveira *et al.*, 2020), ela é custosa em bases de código legadas.

A fronteira mais recente da pesquisa aponta para a Inteligência Artificial como a

solução emergente, indicando uma mudança de paradigma onde a IA passa de objeto de estudo para agente de mitigação. A proposta de Shi (2025) de usar LLMs como "sanitizadores" é promissora, mas os alertas de Andrade & Silva (2025) e Miranda (2024) trazem ressalvas críticas. Se modelos como o Gemini ou ChatGPT reproduzem os padrões de treinamento, e se esses padrões contêm átomos, existe o risco de um ciclo de feedback positivo, onde a IA perpetua e amplifica a densidade de átomos em novos projetos.

A síntese dessa discussão sugere um caminho híbrido: a mitigação seletiva. Isso envolve utilizar ferramentas automáticas para barrar átomos comprovadamente nocivos, ao mesmo tempo em que se investe na curadoria rigorosa de datasets de treinamento e na engenharia de prompts, garantindo que os assistentes de código não aprendam a confundir seus usuários humanos.

5.6 Principais Implicações Temáticas

A análise conjunta das cinco questões de pesquisa revela um ciclo interdependente entre a tipologia de Átomos de Confusão, seu impacto cognitivo, a manutenção do software e as estratégias de mitigação.

Os átomos identificados (QP1) influenciam diretamente a acurácia e a carga cognitiva dos desenvolvedores (QP2), criando efeitos que, dependendo do ecossistema e do tooling disponível (QP3), podem ou não se refletir em defeitos de software. As ferramentas de detecção (QP4) atuam como mediadores desse impacto, permitindo intervenções que reduzem o esforço cognitivo e potencialmente previnem problemas futuros. Finalmente, as estratégias de mitigação (QP5), sejam manuais, automatizadas ou assistidas por IA, fecham o ciclo, transformando o conhecimento sobre a tipologia e os efeitos dos ACs em práticas concretas que aumentam a legibilidade, reduzem a dívida técnica e promovem a saúde cognitiva das equipes.

Essa integração evidencia que os ACs devem ser abordados de maneira contextual, seletiva e estratégica, considerando simultaneamente a natureza do átomo, o ambiente tecnológico e os mecanismos de intervenção disponíveis.

Para além das respostas pontuais às Questões de Pesquisa, a revisão permitiu identificar padrões transversais que caracterizam o fenômeno na atualidade. Inicialmente, observa-se uma tensão evidente entre a biologia do programador e a cultura da linguagem. Embora o substrato neurocognitivo seja universal, visto que o cérebro humano invariavelmente se opõe a ambiguidade, a tolerância ao erro varia conforme a linguagem empregada. O caso do Operador

Condicional (Ternário) exemplifica essa dicotomia: frequentemente vilanizado em C, ele é aceito e considerado seguro em Java. Isso implica que a definição de "Código Limpo" não é um conceito absoluto, mas sim dependente do contexto tecnológico e cultural da equipe de desenvolvimento.

Adicionalmente, a revisão desafia a noção clássica de que "código confuso sempre gera bug", revelando um verdadeiro paradoxo da qualidade. Em ecossistemas modernos, as ferramentas de proteção, como compiladores e *linters*, conseguem mascarar os efeitos negativos dos átomos na estabilidade do sistema. Cria-se, assim, um cenário onde o software é tecnicamente confiável (não apresenta falhas de execução), mas permanece cognitivamente inacessível (difícil de ser compreendido). O risco, portanto, desloca-se da falha do sistema para a exaustão da equipe responsável por sua manutenção.

Por fim, nota-se que a pesquisa sobre Átomos de Confusão está redirecionando seu foco: de estudar humanos lendo código para investigar IAs lendo e gerando código. Considerando que modelos, como o ChatGPT, aprendem a partir de código humano, que naturalmente contém átomos, existe o risco real de perpetuação desses padrões. Nesse cenário, a detecção de ACs torna-se um requisito vital para a curadoria de *datasets* de treinamento, garantindo que a próxima geração de assistentes de codificação não herde as falhas cognitivas de seus criadores.

5.7 Ameaças à Validade

Apesar do rigor metodológico aplicado nesta Revisão Sistemática, a interpretação dos resultados deve considerar limitações intrínsecas ao processo de pesquisa secundária.

Inicialmente, reconhece-se a limitação referente à replicabilidade da estratégia de busca. A opção pelo uso centralizado do Google Scholar como agregador, embora tenha maximizado a abrangência de um termo emergente, impõe desafios à reprodução exata do estudo. Diferentemente de buscas booleanas rígidas em bases estruturadas, os algoritmos de relevância do indexador são dinâmicos e podem variar os resultados conforme o perfil de navegação e a localização, dificultando que outros pesquisadores recuperem a mesma ordenação de registros no futuro.

Uma ameaça adicional reside no viés de publicação, fenômeno sistêmico onde estudos com resultados negativos (ex: átomos que não causaram bugs) tendem a não ser publicados. Isso pode gerar uma percepção de que os ACs são mais perigosos do que a realidade industrial sugere.

Outra limitação significativa concerne à cobertura linguística. A predominância de estudos em C e Java pode ocultar particularidades de linguagens de *script* menos tipadas (PHP, Ruby) ou funcionais. A generalização dos resultados deve ser feita com cautela, evitando assumir que o "núcleo universal"de confusão se aplica indistintamente a todos os paradigmas.

Por fim, destaca-se a volatilidade das ferramentas. As conclusões sobre a capacidade de LLMs (como Gemini e ChatGPT) em detectar átomos representam um recorte momentâneo de 2024/2025. Dada a velocidade de evolução desses modelos, as limitações de alucinação reportadas podem ser superadas rapidamente, exigindo reavaliações constantes da literatura.

5.8 Síntese do Capítulo

A discussão empreendida neste capítulo permitiu consolidar uma visão holística sobre os Átomos de Confusão, demonstrando que o fenômeno transcende a classificação simplista de “código ruim”. Ao confrontar as cinco Questões de Pesquisa, desenhou-se uma trajetória evolutiva clara: a taxonomia dos ACs migrou de um núcleo legado de aritmética visual (em C) para sutilezas comportamentais implícitas (em Java e Web), exigindo que as definições de qualidade sejam dinâmicas e contextuais.

O debate evidenciou que o impacto dos átomos não é apenas técnico, mas fisiológico. A validação neurocognitiva confirmou que a ambiguidade impõe um custo biológico ao desenvolvedor, gerando o que se identificou aqui como uma “dívida técnica de legibilidade”. Contudo, observou-se o paradoxo da qualidade: em ecossistemas modernos, o ferramental robusto dissocia a confusão cognitiva da falha de *software*, criando sistemas que funcionam corretamente, mas que exaurem as equipes de manutenção.

Por fim, a análise das ferramentas e estratégias de mitigação expôs um momento de transição tecnológica. Identificou-se uma lacuna crítica na ausência de detecção em tempo real nos IDEs e uma mudança de paradigma com a entrada da Inteligência Artificial. Conclui-se que o combate aos ACs no futuro próximo dependerá de um modelo híbrido, que combine a “mitigação seletiva” automatizada com a curadoria humana rigorosa para evitar que a IA perpetue os vieses cognitivos que herdou. Com essas interpretações estabelecidas, o trabalho segue para as suas considerações finais.

6 CONCLUSÃO

A presente Revisão Sistemática da Literatura cumpriu o objetivo de mapear e analisar o estado da arte sobre os Átomos de Confusão, consolidando um *corpus* de 32 estudos primários que abrangem desde a definição seminal do conceito até as fronteiras mais recentes da Inteligência Artificial. A análise integrada dos dados permitiu responder às cinco Questões de Pesquisa, oferecendo uma visão holística sobre como microestruturas de código impactam a cognição humana e a qualidade do software. Em consonância com as práticas de Ciência Aberta e visando garantir a replicabilidade deste estudo, as planilhas de organização, extração e apuração dos artigos selecionados encontram-se disponíveis publicamente no repositório da autora deste trabalho[1].

6.1 Principais Contribuições

A investigação proporcionou contribuições teóricas e práticas significativas para a Engenharia de Software. Primeiramente, conclui-se que a taxonomia dos Átomos de Confusão evoluiu de um inventário estático de "más práticas" em C/C++ para um conceito dinâmico e dependente de paradigma. A revisão evidenciou que, embora exista um substrato neurocognitivo universal, a qual o cérebro invariavelmente luta contra a ambiguidade, a materialização do erro é moldada pela cultura da linguagem. O que é um vetor crítico de falhas em linguagens de sistema pode ser apenas um ruído de legibilidade tolerável em ecossistemas gerenciados e modernos.

Uma contribuição central deste trabalho foi a identificação e a formalização do que chamamos de paradoxo da qualidade. O estudo desmistificou a premissa de que a remoção de átomos resulta automaticamente na redução de defeitos. Demonstrou-se que, graças à robustez dos compiladores e ferramentas modernas, os átomos atuam hoje majoritariamente como uma "dívida técnica de legibilidade", onerando a compreensão e o aprendizado da equipe, e menos como uma causa direta de falhas catastróficas. Essa constatação reorienta a estratégia de manutenção: recomenda-se abandonar a refatoração dogmática e massiva em favor de uma mitigação seletiva, focada nos padrões que comprovadamente geram erro humano.

Por fim, o estudo posiciona a Inteligência Artificial como um elemento de dupla face. Enquanto os LLMs prometem atuar como "sanitizadores" eficientes, reduzindo o custo da refatoração manual, eles também introduzem o risco de um ciclo de feedback negativo,

[1] <https://github.com/cacaffurtado/rsl-atoms-of-confusion-data.git>

perpetuando a confusão se treinados sobre bases de código “sujas”. Assim, a pesquisa sobre ACs reafirma-se não como um tema superado, mas como um requisito fundamental para a curadoria de dados e para a garantia da inteligibilidade na era da programação assistida por IA.

6.2 Trabalhos Futuros

Considerando as lacunas tecnológicas e teóricas identificadas, propõe-se uma agenda de pesquisa para avançar o estado da arte:

- Desenvolvimento de Plugins de IDE em Tempo Real: Criação e validação experimental de ferramentas que atuem no momento da digitação ("shift-left"), verificando se o alerta imediato é mais eficaz do que a detecção tardia em *Pull Requests* para prevenir a inserção de átomos.
- Código Gerado por IA: Estudos comparativos para medir a densidade de átomos em códigos gerados por ferramentas como ChatGPT, Copilot e Gemini versus código humano, visando identificar se as IAs estão amplificando ou reduzindo a entropia do software.
- Investigação em Linguagens Modernas: Mapeamento taxonômico de átomos em linguagens focadas em segurança e concisão, como Rust, Go e Kotlin, para verificar se suas novas sintaxes eliminam a confusão cognitiva ou apenas a transmutam para novas formas.
- Impacto no *Onboarding*: Estudos longitudinais focados em mensurar como a alta densidade de átomos afeta o tempo de *onboarding* de novos desenvolvedores em projetos legados, quantificando o custo financeiro da “dívida de legibilidade”.

Espera-se que este mapeamento sirva de base para o desenvolvimento de ferramentas mais sensíveis ao Fator Humano e inspire novas investigações sobre a interação cognitiva entre desenvolvedores e a complexidade crescente das linguagens de programação.

REFERÊNCIAS

ABDELSALAM, Y.; PEITEK, N. *et al.* How do humans and llms process confusing code? **Journal of Software: Evolution and Process**, 2025. Acessado em: 6 May 2025. Disponível em: <https://arxiv.org/abs/2508.18547>.

ANDRADE, M. V. R.; SILVA, J. R. Detectando átomos de confusão utilizando o gemini: percepções iniciais. In: **Anais do V Workshop de Sistemas de Informação**. [s.n.], 2025. Acessado em: 7 May 2025. Disponível em: <https://www.ifmg.edu.br/ourobranco/nossos-cursos/graduacao-6/anais_do_v_wsi_2025_compressed.pdf>.

BERGUM, O. H. *et al.* Unexpected but informative: What fixation-related potentials tell us about the processing of confusing program code. In: **Anais da ICPC 2024**. [s.n.], 2024. Acessado em: 8 May 2025. Disponível em: <https://arxiv.org/abs/2412.10099>.

BOGACHENKOVA, V.; NGUYEN, L.; EBERT, F.; SEREBRENIK, A.; CASTOR, F. Evaluating atoms of confusion in the context of code reviews. In: **2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)**. [s.n.], 2022. Acessado em: 11 Junho 2025. Disponível em: <https://ieeexplore.ieee.org/document/9978262>.

CAMPOS, G. H. **A Relação entre a Experiência de Engenheiros de Software e a Compreensão e Refatoração de Átomos de Confusão**. Dissertação (Mestrado) — PUC Minas, 2022. Acessado em: 7 May 2025. Disponível em: <https://bib.pucminas.br/pergamumweb/vinculos/000013/00001341.pdf>.

CASTOR, F. Identifying confusing code in swift programs. In: **Proceedings of the VI Workshop on Software Visualization, Evolution and Maintenance (VEM 2018)**. Porto Alegre, RS, Brasil: SBC, 2018. p. 41–48. Acessado em: 12 Dez. 2026. Disponível em: <https://doi.org/10.5753/vem.2018.14442>.

COOK, D. A.; WEST, C. P. Conducting systematic reviews in medical education: a stepwise approach. **Medical Education**, v. 46, n. 10, p. 943–952, 2012. Acessado em: 18 June 2025. Disponível em: <https://doi.org/10.1111/j.1365-2923.2012.04328.x>.

COSTA, J. A. S.; GHEYI, R.; CASTOR, F.; OLIVEIRA, P. R. F.; RIBEIRO, M.; FONSECA, B. Seeing confusion through a new lens: on the impact of atoms of confusion on novices' code comprehension. **Empirical Software Engineering**, v. 28, n. 4, 2023. Acessado em: 7 May 2025. Disponível em: <https://link.springer.com/article/10.1007/s10664-023-10311-0>.

COSTA, J. A. S. D.; GHEYI, R. Evaluating the code comprehension of novices with eye tracking. In: **Proceedings of the XXII Brazilian Symposium on Software Quality (SBQS 2023)**. [s.n.], 2023. p. 332–334. Acessado em: 6 May 2025. Disponível em: <https://doi.org/10.1145/3629479.3629490>.

FEIJÓ, D. N.; ALMEIDA, C. D. A. de; ROCHA, L. S. Studying the impact of continuous delivery adoption on atoms of confusion rate in open-source projects. In: **Proceedings of the Brazilian Symposium on Software Engineering (SBES/VEM)**. [s.n.], 2023. Acessado em: 7 May 2025. Disponível em: <https://www.authorea.com/doi/full/10.22541/au.167570695.54470176>.

FEIJÓ, D. N.; ALMEIDA, C. D. A. de; ROCHA, L. S. Studying the impact of ci/cd adoption on atoms of confusion distribution and prevalence in open-source projects. **Journal of Software Engineering Research and Development (JSERD)**, v. 12, n. 1, 2024. Acessado em: 7 May 2025. Disponível em: <https://journals-sol.sbc.org.br/index.php/jserd/article/view/4118>.

GOPSTEIN, D.; APEL, S. Thinking aloud about confusing code: A qualitative investigation of program comprehension and atoms of confusion. **arXiv preprint**, 2020. Acessado em: 5 May 2025. Disponível em: <https://dl.acm.org/doi/10.1145/3368089.3409714>.

GOPSTEIN, D.; HONG, H.; YI, Y.; DELONG, L. A.; ZHUANG, Y.; YEH, M. K.-C.; CAPPOS, J. Prevalence of confusing code in software projects: Atoms of confusion in the wild. In: **Proceedings of the 15th International Conference on Mining Software Repositories (MSR '18)**. [s.n.], 2018. Acessado em: 5 May 2025. Disponível em: <https://dl.acm.org/doi/abs/10.1145/3196398.3196432>.

GOPSTEIN, D.; IANNACONE, J.; YAN, Y.; DELONG, L. A.; ZHUANG, Y.; YEH, M. K.-C.; CAPPOS, J. Understanding misunderstandings in source code. In: **Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017)**. [s.n.], 2017. p. 129–139. Acessado em: 5 May 2025. Disponível em: <https://doi.org/10.1145/3106237.3106264>.

KITCHENHAM, B. **Procedures for performing systematic reviews**. [S.l.], 2004. Acessado em: 1 June 2025. Disponível em: <https://www.researchgate.net/publication/228756057>.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing systematic literature reviews in software engineering**. [S.l.], 2007. Acessado em: 2 June 2025. Disponível em: <https://www.researchgate.net/publication/302924724>.

KITCHENHAM, B. A.; BURN, A. J.; LI, Z. **A Quality Checklist for Technology-Centred Testing Studies**. [S.l.], 2008. Acessado em: 1 June 2025. Disponível em: <https://www.researchgate.net/publication/228938479>.

LANGHOUT, C. **Investigating the perception and effects of misunderstandings in Java code**. Dissertação (Mestrado) — Delft University of Technology, 2020. Acessado em: 7 May 2025. Disponível em: <https://resolver.tudelft.nl/uuid:230e4b8e-8c6c-4188-aa8b-9d4e8a0e0b2c>.

LANGHOUT, C.; ANICHE, M. Atoms of confusion in java. In: **Proceedings of the 29th IEEE/ACM International Conference on Program Comprehension (ICPC '21)**. [s.n.], 2021. Acessado em: 11 Junho 2025. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9462963/>.

MANOR, I. M. V. **Átomos de Confusão em Swift**. Trabalho de Graduação — Universidade Federal de Pernambuco (UFPE), 2018. Acessado em: 7 May 2025. Disponível em: <https://www.cin.ufpe.br/~tg/2018-2/TG_CC/tg_imvm.pdf>.

MEDEIROS, F.; LIMA, G.; AMARAL, G.; APEL, S.; KÄSTNER, C.; RIBEIRO, M.; GHEYI, R. An investigation of misunderstanding code patterns in c open-source software projects. **Empirical Software Engineering**, v. 24, p. 1693–1726, 2019. Acessado em: 6 May 2025. Disponível em: <https://doi.org/10.1007/s10664-018-9666-x>.

MENDES, W.; VIANA, W. **BOHR - Uma Ferramenta para a Identificação de Átomos de Confusão em Códigos Java**. 2021. Acessado em: 7 May 2025. Disponível em: <https://sol.sbc.org.br/index.php/vem/article/view/17216>.

METHLEY, A. M.; CAMPBELL, S.; CHEW-GRAHAM, C. *et al.* Pico, picos and spider: a comparison study of specificity and sensitivity in three search tools for qualitative systematic reviews. **BMC Health Serv Res**, v. 14, n. 579, 2014. Acessado em: 18 June 2025. Disponível em: <https://doi.org/10.1186/s12913-014-0579-0>.

MIRANDA, T. J. P. **Discernimento entre Códigos-fonte Gerados por Pessoas e por Inteligência Artificial: Estudo com Programação em Java**. Trabalho de Graduação — Universidade Federal da Paraíba (UFPB), 2024. Acessado em: 7 May 2025. Disponível em: <https://repositorio.ufpb.br/jspui/handle/123456789/34736>.

MOHER, D.; LIBERATI, A.; TETZLAFF, J.; ALTMAN, D. G. Preferred reporting items for systematic reviews and meta-analyses: the prisma statement. **PLoS Medicine**, v. 6, n. 7, 2009. Acessado em: 5 June 2025. Disponível em: <https://www.researchgate.net/publication/51156625>.

OLIVEIRA, B.; RIBEIRO, M.; COSTA, J. A. S. D. *et al.* Atoms of confusion: The eyes do not lie. In: **Proceedings of the XXXIV Brazilian Symposium on Software Engineering (SBES '20)**. New York, NY, USA: ACM, 2020. p. 243–252. Acessado em: 7 May 2025. Disponível em: <https://doi.org/10.1145/3422392.3422437>.

PINHEIRO, O.; ROCHA, L.; VIANA, W. How they relate and leave: understanding atoms of confusion in open-source java projects. In: **2023 IEEE 23rd International Working Conference on Source Code Analysis and Manipulation (SCAM)**. [s.n.], 2023. p. 119–130. Acessado em: 7 May 2025. Disponível em: <https://doi.org/10.1109/SCAM59687.2023.00022>.

POLLOCK, A.; BERGE, E. How to do a systematic review. **International Journal of Stroke**, v. 13, n. 2, p. 138–156, 2017. Acessado em: 5 Agost 2025. Disponível em: <https://doi.org/10.1177/1747493017743796>.

SHI, G. Studying and improving code understandability through atoms of confusion. In: **ICSE 2025 Doctoral Symposium**. [s.n.], 2025. Acessado em: 7 May 2025. Disponível em: <https://ieeexplore.ieee.org/document/11024327>.

SHI, G.; KAZEMI, F.; GODFREY, M. W.; MCINTOSH, S. Reevaluating the defect proneness of atoms of confusion in java systems. In: **Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '24)**. [s.n.], 2024. Acessado em: 7 May 2025. Disponível em: <https://dl.acm.org/doi/abs/10.1145/3674805.3686677>.

SINGH, D.; MISHRA, A.; AGGARWAL, A. An empirical approach to understand the role of emotions in code comprehension. **Journal of Computer Languages**, v. 79, 2024. ISSN 2590-1184. Acessado em: 5 May 2025. Disponível em: <https://doi.org/10.1016/j.cola.2024.101269>.

SUGIYAMA, Y.; MORISAKI, S.; TOYAMA, A.; KATAHIRA, K. Relationship between model-based decision-making and the comprehension performance of source code with confusing patterns. **Empirical Software Engineering**, v. 51, n. 6, p. 1783–1800, 2025. Acessado em: 6 May 2025. Disponível em: <https://ieeexplore.ieee.org/document/10985862>.

TABOSA, D.; ROCHA, L. A dataset of atoms of confusion in the android open source project. **Data in Brief**, 2024. Acessado em: 7 May 2025. Disponível em: <https://dl.acm.org/doi/10.1145/3643991.3644874>.

TAHSIN, N.; FUAD, N.; SATTER, A. Prevalence of 'atoms of confusion' in open source java systems: An empirical study. **IEEE Access**, 2023. Acessado em: 7 May 2025. Disponível em: <https://scite.ai/reports/prevalence-of-atoms-of-confusion-LeOd5bZA>.

TORRES, A. *et al.* An investigation of confusing code patterns in javascript. **Journal of Systems and Software**, 2023. Acessado em: 6 October 2025. Disponível em: <https://doi.org/10.1016/j.jss.2023.111731>.

YEH, M. K.-C. *et al.* Detecting and comparing brain activity in short program comprehension using eeg. In: **Anais da FIE 2017**. [s.n.], 2017. Acessado em: 6 May 2025. Disponível em: <https://doi.org/10.1109/FIE.2017.8190486>.

YEH, M. K.-C.; YAN, Y.; ZHUANG, Y.; DELONG, L. A. Identifying program confusion using electroencephalogram measurements. **Biomedical Signal Processing and Control**, 2022. Acessado em: 6 May 2025. Disponível em: <https://doi.org/10.1080/0144929X.2021.1933182>.

ZHUANG, Y. *et al.* Do developer perceptions have borders? comparing c code responses across continents. **Software Quality Journal**, v. 32, n. 2, p. 431–457, 2023. Acessado em: 6 May 2025. Disponível em: <https://dl.acm.org/doi/abs/10.1007/s11219-023-09654-0>.