Pensamento Crítico na Aprendizagem de Programação: Diagnóstico do Gap entre Autopercepção e Habilidade Demonstrada em Estudantes de Computação

Deise M. Arndt[1], Ramon M. Martins[1], Jean Carlo R. Hauck[2]

[1] Instituto Federal de Santa Catarina – Campus São José (IFSC)
[2]Programa de Pós-Graduação em Ciência da Computação –
Universidade Federal de Santa Catarina (UFSC).

{deise.arndt,ramon.martins}@ifsc.edu.br, jean.hauck@ufsc.br

Abstract. *Critical thinking is an essential competency in computing education, yet empirical diagnoses of how students perceive and demonstrate these skills in programming contexts remain scarce. This study applies the ProgCTQ to 384 programming students from three Brazilian institutions, analyzing nine critical thinking dimensions through Likert self-assessment items and rubric-scored open-ended responses. Results reveal statistically significant gaps between self-perception and demonstrated ability, with students overestimating skills such as Induction (d = 1.58) and Interpretation (d = 1.29), while underestimating Explanation (d = –0.29). No significant gender differences were found. These findings provide actionable evidence for educators designing metacognitive interventions in programming courses*

Resumo. *O pensamento crítico é uma competência essencial na educação em Computação, porém diagnósticos empíricos sobre como os estudantes percebem e demonstram essas habilidades no contexto da programação ainda são escassos. Este estudo aplica o ProgCTQ a 384 estudantes de programação de três instituições brasileiras, analisando nove dimensões do pensamento crítico por meio de itens Likert de autopercepção e questões abertas avaliadas por rubrica. Os resultados revelam discrepâncias estatisticamente significativas entre autopercepção e habilidade demonstrada, com estudantes superestimando habilidades como Indução (d = 1,58) e Interpretação (d = 1,29), e subestimando Explicação (d = –0,29). Não foram encontradas diferenças significativas por gênero. Estes achados fornecem evidências acionáveis para educadores que buscam intervenções metacognitivas em cursos de programação.*

1. Introdução

O pensamento crítico é reconhecido como uma competência essencial na formação de profissionais de Computação, sendo destacado em diferentes *frameworks* internacionais e diretrizes curriculares da área. Documentos como o *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, o *Computer Science Curricula da ACM/IEEE* e o relatório da Sociedade Brasileira de Computação sobre os Grandes Desafios da Educação em Computação enfatizam a importância de habilidades analíticas, avaliativas e reflexivas na prática profissional da Computação [ACM/IEEE-CS 2023; SBC 2021; Bourque e Fairley 2014]. No contexto do ensino de programação, essa competência assume relevância particular, pois programar exige análise sistemática, avaliação de alternativas, inferência lógica e autorregulação, processos que se alinham diretamente às habilidades do pensamento crítico [Arndt et al. 2025a].

Apesar dessa relevância reconhecida, estudos secundários recentes revelam que instrumentos validados e específicos para mensurar o pensamento crítico no contexto da programação são escassos [Arndt et al. 2025b]. A maioria dos estudos existentes utiliza instrumentos genéricos, como o *California Critical Thinking Skills Test* (*CCTST*) ou o *Computational Thinking Scale* (*CTS*), que não capturam as especificidades do raciocínio crítico mobilizado durante atividades de programação [Guggemos *et al*. 2023].

Para preencher essa lacuna, Arndt *et al*. (2025c, 2025d) desenvolveram e validaram o *ProgCTQ* (*Programming Critical Thinking Questionnaire*), um instrumento híbrido de avaliação do pensamento crítico composto por itens fechados em escala Likert e questões abertas avaliadas por rubrica analítica, cobrindo nove habilidades: análise, avaliação, interpretação, inferência, explicação, autorregulação, reconhecimento de premissas, indução e dedução. O instrumento está disponível em: https://gqs.ufsc.br/progctq/. O instrumento demonstrou excelente confiabilidade global ($\alpha = 0{,}900$; $\omega = 0{,}911$) e evidências de validade fatorial satisfatórias.

Entretanto, a validação psicométrica do instrumento, embora necessária, não responde à questão pedagógica central: o que o instrumento revela sobre o perfil real de pensamento crítico dos estudantes de programação? Para isso, a natureza híbrida do *ProgCTQ* oferece uma oportunidade de investigar a discrepância entre o que os estudantes percebem sobre suas habilidades (captado pelos itens Likert) e o que efetivamente demonstram quando confrontados com situações que exigem raciocínio crítico (captado pelas questões abertas).

Essa discrepância entre autopercepção e desempenho demonstrado é amplamente documentada na literatura sobre metacognição e autorregulação da aprendizagem. O efeito Dunning-Kruger [Kruger e Dunning 1999] sugere que indivíduos com menor proficiência tendem a superestimar suas habilidades, enquanto os mais proficientes podem subestimá-las. No contexto do ensino de programação, essa investigação pode fornecer evidências acionáveis para o planejamento de intervenções pedagógicas mais eficazes.

Assim, este estudo contribui para a área de Educação em Computação ao fornecer um diagnóstico empírico do perfil de pensamento crítico de estudantes de programação utilizando o instrumento *ProgCTQ*. Especificamente, o trabalho contribui ao: (i) analisar simultaneamente autopercepção e habilidade demonstrada em nove dimensões do pensamento crítico; (ii) identificar discrepâncias sistemáticas entre essas medidas no contexto do ensino de programação; e (iii) oferecer evidências empíricas que podem orientar intervenções pedagógicas voltadas ao desenvolvimento metacognitivo em cursos de programação. Para responder a esse objetivo, este estudo investiga as seguintes questões de pesquisa:

QP1. Qual o perfil de autopercepção dos estudantes nas nove habilidades do pensamento crítico no contexto da programação?

QP2. Qual o nível de habilidade demonstrada nas questões abertas avaliadas por rubrica?

QP3. Existe discrepância significativa entre autopercepção e habilidade demonstrada, e em quais dimensões ela é mais acentuada?

QP4. Existem diferenças no perfil de pensamento crítico por variáveis demográficas (gênero)?

O artigo está organizado da seguinte forma: a Seção 2 apresenta o referencial teórico e os trabalhos relacionados; a Seção 3 apresenta o instrumento *ProgCTQ* e descreve a metodologia de pesquisa; a Seção 4 detalha a aplicação e os resultados; a Seção 5 discute os achados; e a Seção 6 conclui o trabalho.

2. Referencial Teórico e Trabalhos Relacionados

2.1. Pensamento Crítico e o Ensino de Programação

O Relatório Delphi, conduzido por Facione (1990) sob encomenda da *American Philosophical Association*, estabeleceu um consenso entre especialistas definindo o pensamento crítico como o "julgamento autorregulado e proposital que resulta em interpretação, análise, avaliação e inferência". Esse relatório identificou seis habilidades centrais: interpretação, análise, avaliação, inferência, explicação e autorregulação. Posteriormente, Yeh (2003) complementou esse modelo com habilidades adicionais, incluindo reconhecimento de premissas, indução e dedução.

No contexto do ensino de Computação, o pensamento crítico tem sido reconhecido como fundamental por diretrizes curriculares de referência. O SWEBOK e o framework ACM/IEEE-CS 2023 destacam a capacidade de análise reflexiva e avaliação como competências transversais essenciais para profissionais de Computação [ACM/IEEE-CS 2023]. A programação, em particular, constitui um ambiente privilegiado para o desenvolvimento dessas habilidades, pois envolve decomposição de problemas, avaliação de soluções, depuração baseada em evidências e comunicação de decisões técnicas [Huang e Qiao 2024, Voskogliou e Buckley 2012].

2.2. Autopercepção vs. Habilidade Demonstrada

A literatura sobre metacognição evidencia que a capacidade de autoavaliação é um componente central da aprendizagem autorregulada [Flavell 1979]. Estudantes que conseguem identificar com precisão suas forças e fraquezas estão em melhor posição para direcionar seus esforços de estudo. Contudo, Kruger e Dunning (1999) demonstraram que indivíduos com menor competência em um domínio tendem a superestimar seu desempenho, fenômeno conhecido como efeito Dunning-Kruger. No ensino de programação, essa discrepância pode ter implicações práticas significativas: estudantes que superestimam suas habilidades de análise ou depuração podem não investir adequadamente no desenvolvimento dessas competências.

2.3. Avaliação do Pensamento Crítico no Ensino de Computação

Estudos secundários recentes [Arndt et al. 2025a, 2025b] identificaram apenas 12 estudos primários que abordam a avaliação do pensamento crítico no ensino técnico e superior em Computação. A maioria utiliza instrumentos genéricos como o *Computational Thinking Scale (CTS)* [Korkmaz et al. 2017] ou o *CCTST* [Facione 1994], que não foram desenvolvidos especificamente para o contexto da programação, nem exclusivamente para o pensamento crítico. Além disso, predomina o uso de

autoavaliação como único método, sem contraste com medidas de desempenho demonstradas.

Essa lacuna motivou o desenvolvimento do *ProgCTQ*, que se diferencia por: (i) ser específico para o ensino de programação; (ii) utilizar um modelo híbrido que combina autopercepção (Likert) e demonstração (questões abertas com rubrica); e (iii) cobrir nove habilidades do pensamento crítico fundamentadas teoricamente [Arndt *et al*. 2025c, 2025d]. O presente estudo avança essa linha de pesquisa ao utilizar o instrumento, considerando sua validade e confiabilidade, para analisar o perfil de pensamento crítico dos estudantes de programação.

3. Metodologia de Pesquisa

Este estudo caracteriza-se como uma pesquisa quantitativa, transversal e diagnóstica, do tipo *survey*, com o objetivo de analisar o perfil de pensamento crítico de estudantes de programação a partir da aplicação do instrumento *ProgCTQ* [Arndt *et al*. 2025c; Lakatos e Marconi 2021, Gil 2008]. A pesquisa foi aprovada pelos Comitês de Ética em Pesquisa em Seres Humanos (CEPSH) da UFSC (CAAE n° 84019224.4.3001.0185) e do IFSC (CAAE nº 84019224.4.0000.0121), em conformidade com os princípios éticos da Resolução CNS nº 466/2012, garantindo participação voluntária e anônima.

3.1. Participantes

Participaram do estudo 384 estudantes matriculados em disciplinas de programação ou correlatas em três instituições públicas brasileiras: Instituto Federal de Santa Catarina (IFSC), Universidade Federal de Santa Catarina (UFSC) e Universidade Federal de Santa Maria (UFSM). A amostra inclui estudantes de Ensino Superior (n = 372; 96,9%) e Ensino Técnico Integrado ao Médio (n = 12; 3,1%). Quanto ao gênero, 311 participantes (81,0%) se identificaram como masculino, 69 (18,0%) como feminino e 4 (1,0%) preferiram não informar. A idade média foi de 22 anos (DP = 4,1), com mediana de 21 anos.

3.2. Instrumento

O *Programming Critical Thinking Questionnaire* (*ProgCTQ*) foi desenvolvido com base no *Evidence-Centered Design* (ECD) [Mislevy et al. 2003] e fundamentado nas habilidades de pensamento crítico definidas por Facione (1990) e Yeh (2003).

O *ProgCTQ* é composto por 27 itens fechados em escala Likert de cinco pontos (1 = Nunca a 5 = Sempre) e 9 questões abertas, distribuídos em nove habilidades do pensamento crítico: Análise, Avaliação, Interpretação, Inferência, Explicação, Autorregulação, Reconhecimento de Premissas, Indução e Dedução. Cada habilidade é avaliada por três itens Likert (autopercepção) e uma questão aberta (habilidade demonstrada), avaliada por rubrica analítica de cinco níveis (1 = Inadequado a 5 = Excelente). O instrumento foi previamente validado com α de Cronbach = 0,900 e ω de McDonald = 0,911[Arndt et al. 2025d].

A Tabela 1 apresenta a estrutura do instrumento, com exemplos de itens Likert e questões abertas para cada habilidade.

Tabela 1. Estrutura do ProgCTQ: habilidades, itens Likert e questões abertas

Habilidade	Exemplo de Item com escala Likert	Questão Aberta (resumida)
Análise	Identifico partes do código que podem ser melhoradas	Como você analisaria um código com instruções repetidas para melhorá-lo?
Avaliação	Verifico se minha solução é a mais adequada para o problema	Como você decide qual é a melhor opção para resolver um problema?
Interpretação	Entendo o que cada parte do código faz	Como você descobriria o que um código sem documentação faz?
Inferência	Tiro conclusões lógicas a partir dos resultados do código	Como você descobre a causa de um comportamento inesperado?
Explicação	Explico meu código de forma clara para qualquer pessoa	Como você explicaria seu programa para alguém sem conhecimento técnico?
Autorregulação	Reviso meu próprio código antes de entregar	Como você avalia se seu código está realmente bom?
Reconhecimento de Premissas	Identifico as condições necessárias antes de começar	Que suposições você faz sobre os dados ao criar um programa?
Indução	Aprendo com experiências anteriores	Descreva uma situação em que usou experiências anteriores para resolver um problema
Dedução	Prevejo os resultados de mudanças no código	Como você prevê o impacto de alterar uma condição no código?

3.3. Coleta de dados

A aplicação ocorreu durante aulas regulares de disciplinas de programação, durante o período letivo, de forma presencial e digital, com tempo médio de preenchimento de 30 minutos. O instrumento foi aplicado ao final de cada disciplina, captando o nível de desenvolvimento do pensamento crítico após a experiência formativa. As questões abertas foram pontuadas pela rubrica analítica manualmente por dois dos autores, com revisão item a item.

3.4. Procedimentos de Análise

Os escores Likert foram agregados por habilidade (média dos três itens por dimensão), sendo tratados como dados aproximadamente intervalares, conforme sustentado na literatura para escalas agregadas com cinco ou mais categorias de resposta [Huh e Gim 2025; Harpe 2015]. Para a análise inferencial, adotou-se uma abordagem conservadora com testes não paramétricos, dada a natureza ordinal dos itens individuais. Para as questões de pesquisa QP1 e QP2, foram calculadas estatísticas descritivas (média e desvio padrão) por habilidade. Para QP3, a discrepância entre autopercepção e habilidade demonstrada foi testada por meio do teste de Wilcoxon para amostras pareadas (dados ordinais) e quantificada pelo d de Cohen para o tamanho do efeito. Para QP4, foi utilizado o teste de Mann-Whitney para comparar as médias Likert entre gêneros. O nível de significância adotado foi $\alpha = 0,05$ [Field 2018].

4. Resultados

4.1. QP1: Perfil de Autopercepção (Likert)

A média global de autopercepção foi de 3,89 (DP = 0,47) na escala de 1 a 5, indicando que os estudantes se percebem, em média, entre "Às vezes" e "Frequentemente" no

exercício das habilidades de pensamento crítico. A Figura 1 apresenta a distribuição das respostas Likert por habilidade.

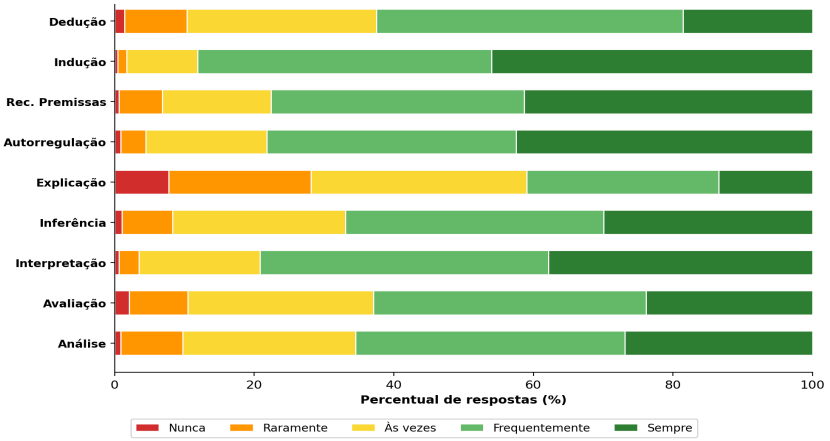

Figura 1. Distribuição percentual das respostas Likert por habilidade

As habilidades com maior autopercepção foram: Indução (M = 4,32), Autorregulação (M = 4,15) e Interpretação (M = 4,13). Nessas dimensões, mais de 70% das respostas concentram-se nas categorias "Frequentemente" e "Sempre". Em contrapartida, Explicação (M = 3,19) apresentou a menor média, com expressiva concentração nas categorias "Raramente" e "Às vezes", especialmente no item "Documento o raciocínio das escolhas técnicas" (M = 2,84), o item com menor pontuação de todo o instrumento.

A Figura 2 apresenta o mapa de calor com as médias individuais dos 27 itens Likert, organizados por habilidade, evidenciando que os itens relacionados à documentação e comunicação (Explicação) e à antecipação de problemas (Dedução, Inferência) representam os pontos mais críticos na autopercepção dos estudantes.

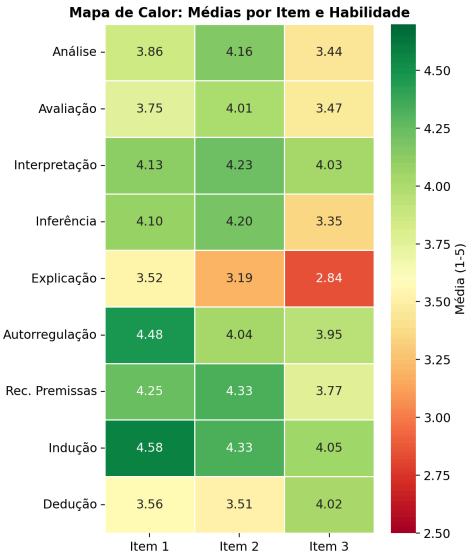

Figura 2. Mapa de calor das médias Likert por item e habilidade

4.2. QP2: Habilidade Demonstrada (Rubrica)

As médias de habilidade demonstrada, avaliadas por rubrica para todos os 384 participantes, variaram entre 2,92 (Indução) e 3,45 (Explicação). A Tabela 2 apresenta os resultados descritivos por habilidade.

Tabela 2. Estatísticas descritivas por habilidade (Likert e Rubrica N=384)

Habilidade	Média Likert	DP	Média Rubrica	DP	Gap	Cohen d	Wilcoxon
Análise	3,82	0,67	3,11	0,93	+0,71	0,87	***
Avaliação	3,74	0,70	3,02	1,11	+0,73	0,78	***
Interpretação	4,13	0,64	2,99	1,07	+1,14	1,29	***
Inferência	3,88	0,64	3,13	1,03	+0,75	0,87	***
Explicação	3,19	0,86	3,45	0,95	–0,26	–0,29	***
Autorregulação	4,15	0,63	3,14	1,01	+1,02	1,20	***
Rec. Premissas	4,11	0,71	3,24	1,14	+0,88	0,92	***
Indução	4,32	0,54	2,92	1,13	+1,40	1,58	***
Dedução	3,69	0,70	2,99	1,14	+0,70	0,74	***

*Nota: * $p < 0,05$; *** $p < 0,001$. Gap = Média Likert – Média Rubrica. Cohen d positivo indica superestimação*

4.3. QP3: Discrepância Autopercepção vs. Demonstração

A Figura 3 apresenta a comparação entre autopercepção e habilidade demonstrada nas nove dimensões, evidenciando o padrão de discrepância.

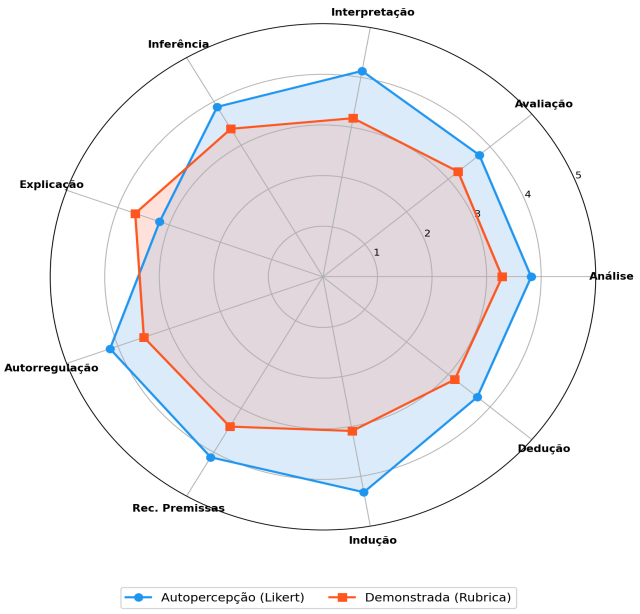

Figura 3. Perfil do Pensamento Crítico: Autopercepção (Likert) vs. Habilidade Demonstrada (Rubrica).

O teste de Wilcoxon para amostras pareadas revelou diferenças estatisticamente significativas entre autopercepção e demonstração em todas as nove habilidades ($p < 0,05$). Em oito das nove dimensões, os estudantes superestimam suas habilidades,

com tamanhos de efeito variando de moderado a grande (d = 0,74 a 1,58) (Tabela 2). A Figura 4 ilustra a magnitude e a direção dos *gaps*.

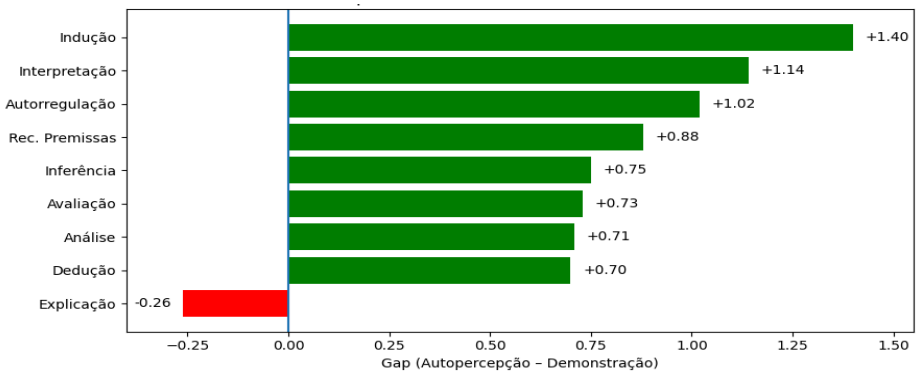

Figura 4. Magnitude da discrepância entre Autopercepção e Demonstração por habilidade (gaps ordenados)

Os maiores gaps foram observados em Indução (gap = +1,40; d = 1,58), onde os estudantes relatam "Sempre" aprender com experiências anteriores, mas demonstram dificuldade em articular como essas experiências foram efetivamente aplicadas; e Interpretação (gap = +1,14; d = 1,29), onde a alta autopercepção sobre entender o código não se traduz proporcionalmente na capacidade de descrever processos interpretativos.

O achado mais notável é que Explicação é a única dimensão com gap negativo (gap = −0,26; d = −0,29): os estudantes se autoavaliam com a menor média Likert do instrumento (M = 3,19), mas quando solicitados a explicar seu código para leigos, demonstram desempenho superior ao que acreditam possuir (M = 3,45).

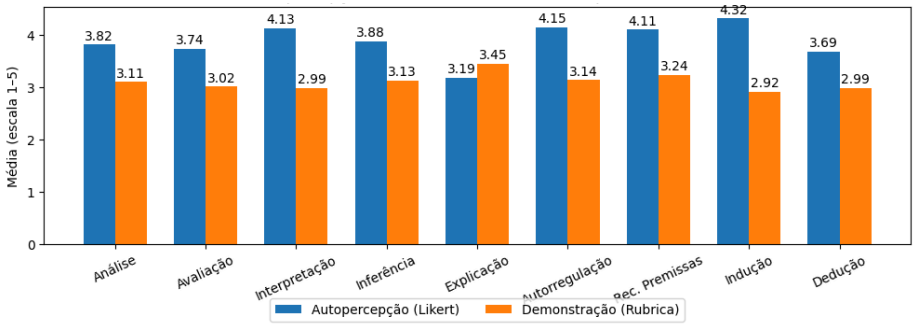

Figura 5. Comparação das médias de Autopercepção vs Habilidade Demonstrada por habilidade: Likert *vs*. Rubrica

4.4. QP4: Diferenças por Gênero

O teste de Mann-Whitney não revelou diferenças estatisticamente significativas entre estudantes do gênero masculino e feminino em nenhuma das nove dimensões (todos p > 0,05). As médias foram consistentemente similares entre os grupos, sugerindo que o perfil de autopercepção do pensamento crítico não varia por gênero nesta amostra. Este achado reforça que as discrepâncias observadas são um fenômeno transversal ao corpo discente.

5. Discussão

Os resultados deste estudo revelam achados sobre o perfil de pensamento crítico de estudantes de programação, com implicações diretas para a prática pedagógica e para a pesquisa em Educação em Computação.

O achado de que estudantes superestimam sistematicamente suas habilidades em oito das nove dimensões é consistente com a literatura sobre o efeito Dunning-Kruger [Kruger e Dunning 1999] e com estudos sobre calibração metacognitiva em domínios técnicos. No caso da Indução (d = 1,58) e Interpretação (d = 1,29), os tamanhos de efeito são grandes, indicando que os estudantes não apenas se percebem melhor do que demonstram, mas o fazem com magnitude substancial. Esses resultados sugerem que atividades aparentemente familiares, como "aprender com erros" ou "entender o código", podem gerar uma falsa sensação de domínio que não se sustenta quando o estudante precisa articular seu raciocínio.

O gap negativo em Explicação (d = −0,26) é o achado mais contraintuitivo. Os estudantes demonstram capacidade de explicar seu código para leigos acima do que acreditam possuir. Isso pode estar relacionado a uma autopercepção influenciada pela dificuldade de documentação formal, tarefa percebida como árdua no ambiente profissional. Contudo, quando solicitados a explicar em linguagem natural, os estudantes mobilizam habilidades de comunicação que não reconhecem como "documentação". Para educadores, isso sinaliza que estratégias como *code review* oral, *pair programming* e apresentações de código podem ser mais eficazes do que apenas exigir documentação escrita tradicional.

O item com menor pontuação Likert no instrumento inteiro, "Documento o raciocínio das escolhas técnicas" (M = 2,84), reforça um desafio estrutural no ensino de programação: os estudantes reconhecem que não documentam adequadamente. Isso é consistente com a realidade da indústria, onde a documentação é frequentemente negligenciada [ACM/IEEE-CS 2023]. A consciência dessa limitação pode ser explorada pedagogicamente como ponto de partida para intervenções metacognitivas.

A ausência de diferenças significativas entre gêneros é um resultado positivo que contrasta com a percepção comum de disparidades em cursos de Computação. Embora a amostra seja desbalanceada (81% masculino), o resultado sugere que não fora observadas diferenças das habilidades de pensamento crítico autopercebidas nesta amostra.

Implicações para a prática docente. Os achados sugerem que docentes de programação podem se beneficiar de: (i) incorporar atividades explícitas de metacognição, como autoavaliações com *feedback* contrastando percepção e desempenho; (ii) valorizar a explicação oral de código como estratégia para desenvolver a confiança dos estudantes em suas habilidades de comunicação; (iii) utilizar rubricas explícitas para que os estudantes compreendam os critérios que diferenciam níveis de proficiência em pensamento crítico; e (iv) enfatizar a articulação explícita de raciocínios em atividades de depuração e análise de código.

Limitações. Este estudo apresenta limitações que devem ser consideradas. Primeiro, o desenho transversal não permite inferir causalidade ou evolução temporal. Segundo, a

amostra concentra-se em instituições brasileiras públicas, limitando a generalização a outros contextos. Terceiro, a pontuação por rubrica, embora revisada por dois avaliadores, envolve subjetividade inerente à avaliação de respostas abertas.

6. Conclusão e Trabalhos Futuros

Este estudo apresentou um diagnóstico empírico do perfil de pensamento crítico de 384 estudantes de programação, utilizando o instrumento validado *ProgCTQ* em três instituições brasileiras. O principal achado deste estudo é a identificação de discrepâncias sistemáticas entre autopercepção e habilidade demonstrada.

Os estudantes superestimam sistematicamente suas habilidades, com efeitos grandes em Indução (d = 1,58) e Interpretação (d = 1,29). Paradoxalmente, subestimam sua capacidade de Explicação (d = −0,29), demonstrando melhor desempenho do que acreditam possuir quando precisam comunicar seu raciocínio. A documentação emergiu como o ponto mais crítico autopercebido, e não foram encontradas diferenças significativas por gênero.

Esses achados fornecem evidências acionáveis para docentes: a necessidade de intervenções metacognitivas que ajudem os estudantes a calibrar sua autopercepção, a valorização da explicação oral como estratégia pedagógica, e a incorporação explícita de práticas de pensamento crítico nos currículos de programação.

Como trabalhos futuros, pretende-se: (i) realizar aplicações longitudinais para acompanhar a evolução do pensamento crítico ao longo da formação; (ii) investigar a relação entre o perfil de pensamento crítico e o desempenho acadêmico em programação; (iii) obter uma amostra balanceada de gêneros para somar evidências de que as habilidades de pensamento crítico autopercebidas podem não ser influenciadas pelo gênero.

7. Declaração sobre uso de Inteligência Artificial

A revisão gramatical e ajustes de concordância foram realizados com auxílio de IA generativa (modelo Sonar da Perplexity AI).

8. Referências

ACM/IEEE-CS (2023). Computer Science Curricula 2023. Association for Computing Machinery and IEEE Computer Society.

Arndt, D. M., Martins, R. M. e Hauck, J. C. R. (2025a). Avaliação de Habilidades do Pensamento Crítico no Ensino Técnico e Superior no Ensino de Computação: Um Mapeamento Sistemático. In: Revista Brasileira de Informática na Educação (RBIE), v. 33.

Arndt, D. M., Martins, R. M. e Hauck, J. C. R. (2025b). Avaliação de Habilidades do Pensamento Crítico no Ensino de Computação: Um Mapeamento Sistemático. Simpósio Brasileiro de Informática na Educação (SBIE 2025 - Journal Fisrt).

Arndt, D. M., Martins, R. M. e Hauck, J. C. R. (2025c). Avaliação do Desenvolvimento do Pensamento Crítico no Ensino de Programação: Uma Proposta de Modelo. In: Anais do XXXIII Workshop sobre Educação em Computação (WEI). SBC.

Arndt, D. M., Martins, R. M. e Hauck, J. C. R. (2025d). Avaliação do Desenvolvimento do Pensamento Crítico no Ensino de Programação: uma aplicação piloto. In: Anais do XXXVI Simpósio Brasileiro de Informática na Educação (SBIE). SBC.

Bourque, P.; Fairley, R. E. (2014). Guide to the Software Engineering Body of Knowledge (SWEBOK v3.0). IEEE Computer Society.

Facione, P. A. (1990). Critical thinking: A statement of expert consensus for purposes of educational assessment and instruction (The Delphi Report). California Academic Press.

Facione, P. A. (1994). California Critical Thinking Skills Test – CCTST. California Academic Press.

Field, A. (2018). Discovering Statistics Using IBM SPSS Statistics. 5th ed. Sage.

Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive-developmental inquiry. American Psychologist, 34(10):906-911.

Guggemos, J., Seufert, S. e Sonderegger, S. (2023). Computational thinking assessment – Towards a pedagogically meaningful assessment. Journal of Computing in Higher Education.

Gil, A. C. (2008). Métodos e Técnicas de Pesquisa Social. 6. ed. Atlas.

Harpe, S. E. (2015). How to analyze Likert and other rating scale data. *Currents in Pharmacy Teaching and Learning*, 7(6):836–850.

Huh, I. e Gim, J. (2025). Exploration of Likert scale in terms of continuous variable with parametric statistical methods. *BMC Medical Research Methodology*, 25.

Huang, W. e Qiao, Z. (2024). Integrating critical thinking into computer science education. Education and Information Technologies.

Korkmaz, O., Çakir, R. e Özden, M. Y. (2017). A validity and reliability study of the Computational Thinking Scales (CTS). Computers in Human Behavior, 72:558-569.

Kruger, J. e Dunning, D. (1999). Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments. Journal of Personality and Social Psychology, 77(6):1121-1134.

Lakatos, E. M. e Marconi, M. A. (2021). Fundamentos de Metodologia Científica. 9. ed. Atlas.

Mislevy, R. J., Almond, R. G. e Lukas, J. F. (2003). A brief introduction to evidence-centered design. ETS Research Report Series.

SBC (2021). Grandes Desafios da Educação em Computação no Brasil 2025–2035. Sociedade Brasileira de Computação.

Voskoglou, M. e Buckley, S. (2012). Problem solving and computational thinking in a learning environment. Egyptian Computer Science Journal, 36(4).

Yeh, Y. C. (2003). Integrating collaborative PBL with blended learning to explore a cognitive load perspective. Innovations in Education and Teaching International, 40(3).