Motivação e Aprendizagem em Programação Orientada a Objetos: Efeitos do Uso de Material Instrucional Baseado em Metodologias Ativas

Tallys A. D. Santos[1], **Fabrício V. A. Guerra**[1], **Flavius L. Gorgônio**[1]

[1]Laboratório de Inteligência Computacional Aplicada a Negócios (LABICAN)
Departamento de Computação e Tecnologia (DCT)
Universidade Federal do Rio Grande do Norte (UFRN)
Rua Joaquim Gregório, 296 – 59.300-000 – Caicó – RN – Brasil

tallys.santos.017@ufrn.edu.br
{fabricio.guerra, flavius.gorgonio}@ufrn.br

Abstract. *This article examines the impact of innovative instructional materials, grounded in constructionist, narrative, and self-regulated learning techniques, on the motivation and performance of university students in Object-Oriented Programming. The proposed instructional material is organized into units that progressively develop specific aspects of programming through incremental construction centered on a single application. A case study was conducted with 41 students from a regular third-semester course at a university in Brazil. After the course, data were collected through a questionnaire covering various motivational and performance aspects. The results indicated significant improvements in students' self-assessment of their skills as programmers, in their enjoyment of programming practice, and in intrinsic motivation, reflected by increased time dedicated to programming and greater autonomy in learning. Notably, students who were initially less motivated showed greater improvement in their perception of their skills as programmers.*

Resumo. *Este artigo investiga o efeito de um material didático inovador, baseado em técnicas construcionistas, narrativas e de autorregulação do aprendizado, na motivação e no desempenho de estudantes universitários em Programação Orientada a Objetos. O material instrucional proposto é organizado em unidades que vão desenvolvendo progressivamente aspectos específicos da programação por meio da construção incremental centrada em um mesmo aplicativo. Um estudo de caso foi conduzido com 41 estudantes de uma disciplina regular do terceiro semestre em uma universidade brasileira. Após o curso, foram coletados dados através de um questionário abrangendo diversos aspectos motivacionais e de desempenho. Os resultados indicaram melhorias significativas na autoavaliação das habilidades dos estudantes como programadores, no gosto pela prática da programação e na motivação intrínseca, refletida pelo aumento no tempo dedicado à programação e maior autonomia no aprendizado. Notavelmente, estudantes inicialmente menos motivados tiveram maior evolução na percepção de suas habilidades enquanto programadores.*

1. Introdução

Os altos índices de desistência e reprovação em disciplinas introdutórias de programação têm sido problemas persistentes nos cursos superiores de computação. Pesquisas recen-

tes têm apontado dificuldades técnicas na aprendizagem de programação como um fator determinante desses fenômenos, resultando em elevadas taxas de abandono e baixo desempenho acadêmico de estudantes em diversas universidades [Obaido et al. 2023, Rabelo et al. 2018]. No contexto da Programação Orientada a Objetos (POO), os problemas de ensino-aprendizagem persistem ou até se agravam, na medida em que temas de mais alto nível de abstração entram em cena[Gutiérrez et al. 2022]. Tais dificuldades têm efeitos negativos amplos, afetando não apenas o percurso acadêmico individual, mas também causando prejuízos institucionais e sociais decorrentes da formação incompleta ou inadequada de futuros profissionais da área.

Nesse contexto, a motivação dos estudantes desempenha papel crucial na aprendizagem eficaz de programação. Estudos têm demonstrado que a percepção de prazer e utilidade do conteúdo ensinado influencia diretamente o desempenho acadêmico e o interesse dos estudantes em disciplinas de programação [Zataraín-Cabada et al. 2018]. Além disso, a motivação intrínseca, caracterizada pela curiosidade natural e pelo desejo genuíno de aprender, destaca-se como um dos principais fatores que impulsionam o engajamento e o sucesso acadêmico dos estudantes de computação [Farooq and Anwar 2024]. Em contrapartida, a ausência desses elementos motivacionais pode resultar em baixo comprometimento dos alunos e, consequentemente, baixo desempenho acadêmico.

Diversas pesquisas têm indicado que o uso de metodologias inovadoras ou novos materiais didáticos pode impactar positivamente a motivação e o aprendizado em programação. Por exemplo, estratégias que promovem a aprendizagem autorregulada têm mostrado resultados promissores, aumentando significativamente o desempenho acadêmico e o engajamento estudantil, em comparação a métodos tradicionais [Öztürk 2021]. Da mesma forma, a adoção de abordagens como a programação baseada em ambientes visuais, a exemplo do Scratch, demonstrou melhorias tanto na motivação quanto no desempenho em exames de programação [Wen et al. 2023].

O presente estudo propôs diretrizes para a elaboração e a utilização de um material didático, centrado em metodologias ativas de ensino-aprendizagem [Paiva et al. 2016], para o ensino de programação orientada a objetos, acompanhado por mudanças metodológicas na condução das aulas, visando responder às seguintes questões de pesquisa:

Questão de Pesquisa 1 (QP1): As mudanças propostas afetaram a motivação dos estudantes em relação à programação?

Questão de Pesquisa 2 (QP2): As mudanças propostas afetaram o desempenho dos estudantes em relação à programação?

2. Fundamentação Teórica

Esta seção tem por objetivo deixar explícitas as diretrizes, as bases científicas que orientaram a elaboração do material didático proposto[1], bem como a formatação das aulas desenvolvidas no presente estudo.

O material didático foi concebido tendo por base mais fundamental o Construcionismo [Papert and Harel 1991], enfatizando o aprendizado através da construção ativa de um mesmo aplicativo de software, que adquire mais e mais funcionalidades,

[1] `https://encurtador.com.br/4404n`.

ao longo do tempo, conforme os estudantes avançam no conteúdo. Nesse sentido, cada segmento do material, denominado *receita*, combina instruções claras, explicações teóricas, imagens ilustrativas ou animações e exercícios práticos, sempre girando em torno da construção e evolução do aplicativo em pauta. A ideia é a de que, durante a implementação das receitas, os discentes aprendam conceitos fundamentais de POO *durante* a construção incremental do aplicativo. Além disso, todas as receitas envolvem interfaces gráficas, no intuito de aumentar o engajamento e o desempenho estudantil [Hosseini et al. 2020, Koren 2024, Hany et al. 2023].

Por fim, o material utiliza técnicas narrativas como diálogos informais e ganchos "dramáticos" entre as receitas, criando uma continuidade natural e despertando maior interesse e motivação dos estudantes [Mou 2024]. Desta forma, cada receita produz, em seu desenvolvimento, a solução de um problema, mas o texto é direcionado de forma que mesmo essa solução encontrada apresente, em si, um defeito claro. Esse defeito será endereçado da receita seguinte e a ferramenta para sua resolução será algum assunto da Orientação a Objetos. Tomemos um exemplo para deixar mais claro esse ponto: numa receita em que se introduz no aplicativo uma tabela passível de ordenação por diversas colunas, percebe-se, na solução encontrada, a repetição de um mesmo algoritmo de ordenação por diversas vezes, mudando apenas o critério de ordenação entre as repetições. A solução para esse problema, na receita seguinte, se dará através do uso de polimorfismo, que constitui, em si, o objeto de estudo da receita. Assim, o assunto polimorfismo não vai ser utilizado como uma forma genérica e fora de contexto de poder representar "Gato" e "Cachorro" numa mesma classe, mais abstrata, "Animal", mas sim como uma ferramenta para resolver um problema de Engenharia de Software (replicação de código) identificado num aplicativo em desenvolvimento.

Já no que diz respeito às aulas, elas foram estruturadas em três etapas distintas, sempre centradas nas receitas. Inicialmente, o professor realiza uma exposição oral curta, com duração aproximada de 20 minutos, introduzindo o tema específico da receita e explicando brevemente o aplicativo a ser construído pelos estudantes. Segue-se, então, um debate interativo no qual os estudantes discutem dúvidas e aspectos práticos relacionados à explicação inicial, etapa baseada na importância do diálogo construtivo e interativo para potencializar o aprendizado e a motivação [Tao and Chen 2024, Setiawati 2012]. Essa etapa se entende enquanto houver interação dos estudantes nas discussões.

Finalizado o debate, segue-se para a implementação prática das receitas e exercícios, momento em que o professor atua de forma mais próxima aos estudantes, oferecendo orientações personalizadas conforme as demandas individuais surgem, o que melhora significativamente a proximidade emocional e acadêmica entre professor e aluno, o que, por sua vez, pode influenciar motivação aprendizado [Christophel 1990]. Dependendo da complexidade do assunto abordado e do desempenho da turma na resolução das tarefas inerentes à receita, essa etapa pode estender-se por 2 ou 3 aulas. Ao final da implementação, promove-se outro debate, desta vez focado nas limitações identificadas nas soluções desenvolvidas, motivando organicamente o conteúdo que será abordado na receita subsequente. Esse debate funciona como um gancho narrativo (expediente comum em livros, novelas e seriados), gerando expectativa para "o próximo episódio", que é justamente a implementação da receita seguinte.

Durante todas as atividades, os estudantes foram encorajados a desenvolver um

aprendizado autorregulado, trabalhando em ritmo próprio e autônomo, no intuito de influenciar positivamente seu desempenho [Gill and Holton 2006]. O conjunto das receitas cobriu os principais tópicos clássicos de POO, como classes, encapsulamento, herança e polimorfismo, e também abordou tópicos contemporâneos essenciais, tais como gerência de estados, programação assíncrona, programação declarativa e funções de alta ordem.

Ao todo, foram produzidas 15 receitas didáticas para desenvolvimento ao longo das aulas. Além disso, foi proposto um mini-projeto complementar para tratar tópicos residuais que não se encaixavam organicamente na narrativa principal das receitas e para dar aos estudantes uma experiência fora do contexto do aplicativo desenvolvido.

3. Metodologia

Foi planejado e executado um estudo de caso com o objetivo de avaliar a solução proposta em um contexto real de ensino e aprendizado. Esse estudo foi realizado na disciplina de Programação Orientada a Objetos, ministrada no curso superior de Sistemas de Informação da UFRN, Campus Caicó. Todos os estudantes participantes possuíam, no mínimo, um ano de experiência prévia com programação procedural. A disciplina foi oferecida regularmente, com 38 encontros de 1h40m cada, às segundas e quartas-feiras, distribuídos em cerca de 5 meses, no período da tarde, respeitando todas as avaliações institucionais exigidas e toda a ementa e bibliografia regulares da disciplina.

Ao todo, 41 estudantes estavam aptos a participar do estudo. Houve a desistência de dois deles, sem justificativas formais: um não frequentou nenhuma aula e o outro desistiu durante o primeiro mês de atividades. As aulas e o material didático utilizado seguiram rigorosamente as diretrizes descritas na seção anterior.

Ao término da disciplina, os 39 estudantes que permaneceram até o final responderam a um formulário que abordava diversos aspectos relacionados à percepção individual deles sobre o próprio desempenho em programação e sobre aspectos motivacionais. O formulário continha duas questões de natureza quantitativa, voltadas à autoavaliação enquanto programador e ao gosto pela programação. Em ambas as questões, foram coletadas métricas referentes aos momentos anterior e posterior à realização da disciplina.

Além disso, o formulário continha 9 itens qualitativos que abordavam aspectos motivacionais. Esses itens do formulário foram propositivos, solicitando que os estudantes se posicionassem em relação a afirmações específicas. As respostas foram coletadas por meio de uma escala Likert de cinco opções, sendo uma neutra, duas positivas de diferentes intensidades (“concordo” e “concordo fortemente”) e duas negativas também com variação de intensidade (“discordo” e “discordo fortemente”). Os enunciados de cada um dos itens qualitativos do formulário são debatidos na seção seguinte, na medida em que reportamos os resultados do trabalho.

4. Resultados

A partir dos dados coletados no estudo de caso, foram testadas diversas hipóteses, no intuito de responder às questões de pesquisa através de análises estatísticas que lancem projeções inferenciais para além dos participantes do estudo. Nesse sentido, organizamos as descrições dos dados coletados, as hipóteses e os testes estatísticos de acordo com cada questão de pesquisa.

4.1. QP1 – As mudanças propostas afetaram a motivação dos estudantes em relação à programação?

Inicialmente, analisaremos uma questão numérica, respondida pelos estudantes, quantificando, numa escala de 1 a 5, seu gosto pela programação. Foram coletadas as métricas sobre o gosto por programação *antes* e *depois* do estudo de caso e foi testada a *hipótese nula* de que a média dos valores antes e depois seriam iguais.

Gosto por Programação (antes vs. depois): Não havendo normalidade em ambos os conjuntos de métricas, aplicou-se o teste *Wilcoxon Signed-Rank* (pareado). A hipótese alternativa é a de que a média das métricas representando o gosto pela programação após o estudo de caso foi maior do que a média antes. Os valores resultantes são descritos a seguir:

$$W = 18{,}5; \quad p = 5{,}76 \times 10^{-4}; \quad r = 0{,}7221 \text{ (grande)}.$$

Conclui-se, portanto, que o gosto por programação aumentou significativamente, com tamanho de efeito *grande* (vide o tamanho de efeito **r**), o que já é um indício forte de que a motivação, como um todo, aumentou. Aprofundaremos as análises para entender melhor algumas perspectivas dessa motivação com os itens qualitativos.

Itens qualitativos de motivação: Ao todo, 9 itens qualitativos foram respondidos pelos estudantes. Para avaliar a confiabilidade e a consistência interna desses itens, foi utilizado o coeficiente Cronbach's α, que indica o quanto os itens de um questionário estão correlacionados entre si e se, juntos, eles realmente medem o mesmo conceito ou construto. O alfa de Cronbach é definido como:

$$\alpha = \frac{N \cdot \bar{c}}{\bar{v} + (N-1) \cdot \bar{c}}$$

onde, N representa o número de itens do questionário, $\bar{c}$ representa a covariância média entre os itens e $\bar{v}$ é a variância média dos itens. Para o questionário avaliado, obteve-se:

$$\alpha = 0{,}74.$$

Esse valor indica consistência interna aceitável (maior que $0,7$) para os itens qualitativos, viabilizando as análises que se seguem.

Os itens de 1 a 9 contaram com respostas qualitativas sobre aspectos motivacionais. Para proceder com as análises, as respostas foram codificadas da seguinte forma: 1 = "discordo fortemente"; 2 = "discordo"; 3 = "neutro"; 4 = "concordo"; 5 = "concordo fortemente".

Foram calculadas as médias das respostas em cada item e o respectivo intervalo de confiança, com o objetivo de estimar a dispersão das médias amostrais. A Tabela 1 apresenta, para cada item, sua descrição, a média de suas respostas $\bar{x}$ e o respectivo intervalo de confiança de 95% ($\bar{x} \pm t_{0{,}975} \cdot \text{SEM}$).

A Figura 1 exibe essas médias junto aos respectivos intervalos de confiança, com a linha tracejada em $x = 3$, marcando o ponto neutro. Podemos perceber que todas as projeções inferenciais têm seu limite inferior acima do valor de neutralidade (3), que está em destaque. O item com menos aprovação diz respeito ao formato textual, mas ainda assim está acima do valor de neutralidade. Ademais, 3 itens se destacam com valores do

Tabela 1. Avaliação dos itens qualitativos das questões 1 a 9

#	Item	$\bar{x}$	*IC 95%*
1	Meu tempo programando aumentou com a disciplina de POO	3,7949	[3,5696 – 4,0202]
2	Achei mais motivante estudar programação nesse formato	4,1026	[3,7802 – 4,4249]
3	Seria interessante testar as mudanças em disciplinas anteriores de programação	3,8974	[3,5927 – 4,2022]
4	O formato textual do material utilizado é melhor do que vídeo-aulas	3,5897	[3,1776 – 4,0019]
5	O formato das aulas foi mais interessante para meu aprendizado	4,3846	[4,0815 – 4,6878]
6	Meu papel mais ativo nas aulas auxiliou no meu aprendizado	4,1026	[3,8796 – 4,3257]
7	Eu me senti confortável em seguir o conteúdo no meu próprio ritmo	4,5385	[4,3439 – 4,7330]
8	O formato da disciplina ajuda a quem perder aula ou estiver com dificuldade	4,6154	[4,4241 – 4,8067]
9	Durante a disciplina, melhorei minha habilidade de auto-aprendizado	4,1282	[3,8796 – 4,3769]

limite inferior acima do valor de aprovação (4), que são os Itens 6, 7 e 8. Esses intervalos de confiança indicam que, caso a proposta seja testada com uma outra amostra de estudantes com as mesmas características, em 95% dos casos a média estará acima da neutralidade para todos os itens avaliados. Esses resultados qualitativos, aliados ao tamanho de efeito grande observado na questão quantitativa de motivação, nos permitem responder afirmativamente à questão **QP1:** *As mudanças propostas afetaram positivamente a motivação dos estudantes em relação à programação*.

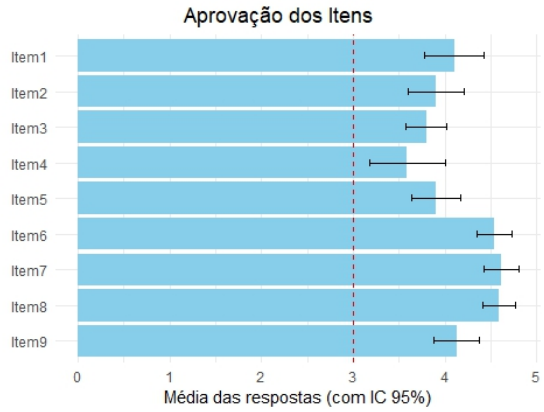

Figura 1. Médias ± IC 95% para os Itens 1–9 (Escala Likert).

Podemos perceber que todas as projeções inferenciais têm seu limite inferior acima do valor de neutralidade (3), que está em destaque na Figura 1. O item com menos aprovação diz respeito ao formato textual, mas ainda assim está acima do valor de neutralidade. Ademais, 3 itens se destacam com valores do limite inferior acima do valor de aprovação (4), que são os Itens 5, 7 e 8. Esses intervalos de confiança indicam que, caso a proposta seja testada com uma outra amostra de estudantes com as mesmas características, em 95% dos casos a média estará acima da neutralidade para todos os itens avaliados. Esses resultados qualitativos, aliados ao tamanho de efeito grande observado na questão quantitativa de motivação, nos permitem responder afirmativamente à questão **QP1:** *As mudanças propostas afetaram positivamente a motivação dos estudantes em relação à programação*.

4.2. QP2 – As mudanças propostas afetaram o desempenho dos estudantes em relação à programação?

Os estudantes responderam a outra questão numérica, quantificando, numa escala de 1 a 10 sua autoavaliação de desempenho enquanto programadores. Foram coletadas as

métricas sobre a autoavaliação que eles tinham de si *antes* e *depois* de participarem do estudo de caso e foi testada a *hipótese* nula de que as médias dos valores antes e depois seriam iguais.

Nota Autoavaliativa de Programação (antes vs. depois de participar do estudo): Não havendo normalidade em ambos os conjuntos de métricas, aplicou-se o teste *Wilcoxon Signed-Rank* (pareado). A hipótese alternativa é a de que a média das métricas representando o desempenho de programação após o estudo de caso foi maior do que a média antes. Os valores resultantes são descritos a seguir:

$$W = 5{,}5; \quad p = 2{,}03 \times 10^{-7}; \quad r = 0{,}8575 \text{ (muito grande)}.$$

Isso demonstra um aumento significativo e de magnitude grande (vide o tamanho de efeito **r**) na autoavaliação do estudante em relação ao desenvolvimento de suas habilidades em programação. Muito mais que isso, o teste indica que se a proposta for aplicada a amostras com as mesmas características, esse aumento também será verificado. Assim, respondemos também a questão **QP2:** *As mudanças propostas afetaram positivamente o desempenho dos estudantes em relação à programação, e em magnitude grande.*

4.3. Agrupamento por Atitude Prévia

Em face dos resultados positivos, e em especial do **Item 8**, que diz respeito ao formato da proposta ajudar estudantes com dificuldade, ter obtido a maior aprovação geral entre todos os itens qualitativos, uma análise *post-hoc* foi realizada. O intuito é o de avaliar se o estudo beneficiou igualmente estudantes com diferentes atitudes prévias em relação ao gosto pela programação. Os resultados correntes já nos dão um efeito geral de que os estudantes evoluíram na sua própria autoavaliação de desempenho como programadores. Queremos, agora, verificar se os estudantes que já gostavam de programar (os que responderam 4 ou 5 para a questão de *gosto por programação (antes)* evoluíram nas mesmas proporções que os demais (os que responderam 1, 2 ou 3).

Diferença em nota de programação (Δ = depois – antes)

- Média de Δ no grupo que já gostava de programar = 1,571.
- Média de Δ no grupo complementar = 2,500 .
- Teste de Mann–Whitney (não pareado):

$$U = 108; \quad p = 0{,}0193.$$

Portanto, a partir do valor-p significativo ($p < 0{,}05$), é possível concluir que o *grupo complementar*, o dos estudantes que, antes de iniciar o estudo, não apresentavam uma atitude de afetividade em relação à programação, teve aumento ainda maior em sua nota autoavaliativa de programação, indicando que a intervenção foi particularmente eficaz para estudantes com essas características.

5. Discussão

Os resultados relacionados ao aprimoramento da motivação e desempenho na aprendizagem de programação têm mostrado uma variedade considerável de desfechos, refletindo a complexidade do processo educacional e as particularidades metodológicas envolvidas.

Alguns estudos não obtiveram resultados significativos com abordagens inovadoras ou alternativas de ensino. Por exemplo, a utilização de *badges* digitais como forma de aumentar a motivação intrínseca dos estudantes não apresentou resultados efetivos em contextos específicos de programação introdutória [Facey-Shaw et al. 2019]. Da mesma forma, revisões sobre o uso de técnicas colaborativas, como a programação em pares, apontam que falhas no design instrucional podem comprometer o desenvolvimento de habilidades e a motivação, resultando em efeitos inexpressivos ou até negativos [Hawlitschek et al. 2022]. Esses resultados mistos indicam que a eficácia de abordagens pedagógicas alternativas é sensível ao contexto e à qualidade do projeto instrucional.

Por outro lado, diversos estudos demonstram efeitos positivos consideráveis em relação à motivação e desempenho acadêmico quando abordagens cuidadosamente planejadas são implementadas. Estudos utilizando a instrução incremental em habilidades básicas de programação têm apresentado melhorias significativas na conclusão das atividades práticas, redução de erros frequentes e melhor compreensão dos conceitos envolvidos [Xie et al. 2019].

Resultados semelhantes foram observados em pesquisas envolvendo ambientes visuais e métodos baseados em blocos de programação. Embora o efeito sobre habilidades diretas de programação e pensamento computacional tenha sido moderado, tais métodos produziram grandes melhorias nas habilidades de resolução de problemas [Chiu and Tsuei 2020]. Além disso, abordagens instrucionais baseadas em linguagens visuais, como o Scratch, quando complementadas por sistemas de recomendação adaptativos, elevaram substancialmente as taxas de aprovação em comparação aos métodos tradicionais [Cárdenas-Cobo et al. 2020].

Os resultados obtidos no presente estudo alinham-se com as evidências positivas apontadas na literatura recente. Especificamente, a melhoria observada na autoavaliação dos estudantes quanto às habilidades de programação e o gosto pela prática da programação refletem um aumento notável na motivação intrínseca, fenômeno consistente com resultados encontrados por outros pesquisadores que utilizaram metodologias instrucionais adaptativas e visuais. Destaca-se, também, o impacto positivo observado na percepção dos estudantes sobre sua capacidade de aprender de forma autônoma, refletindo uma melhoria significativa em aspectos motivacionais relacionados à autonomia no processo de aprendizado.

Um achado particularmente relevante deste estudo refere-se à melhoria mais acentuada na autoavaliação das habilidades de programação entre estudantes que inicialmente não possuíam uma atitude positiva em relação à programação. Esse resultado sugere que estratégias pedagógicas que priorizam aspectos visuais, práticos e incrementais podem ser especialmente eficazes para engajar e motivar estudantes menos predispostos ou que experimentaram dificuldades anteriores no aprendizado da programação.

Embora os resultados do estudo atual sejam bastante promissores, é importante salientar que as conclusões devem ser interpretadas com cautela devido às limitações intrínsecas ao desenho do estudo de caso adotado. A validação dessas descobertas exigirá estudos futuros conduzidos com amostras maiores, em contextos variados e, preferencialmente, por meio de experimentos randomizados. Ainda assim, considerando as limitações metodológicas, as projeções inferenciais derivadas dos resultados observados

e as comparações com a literatura existente sugerem que o formato do material instrucional proposto possui grande potencial para influenciar positivamente a motivação e o desempenho de estudantes de programação.

6. Considerações Finais

Este trabalho apresentou diretrizes de produção de material didático para ensino de POO baseadas em diversos fundamentos científicos, em especial em metodologias ativas. Um material foi produzido com base nas diretrizes e testado num estudo de caso com estudantes universitários. Os resultados apontaram melhorias significativas associadas à utilização do material, tanto na percepção dos estudantes em relação ao próprio desempenho enquanto programadores, quanto em questões motivacionais importantes como o gosto pela prática da programação e o tempo dedicado à sua prática.

Há limitações, como não poderia deixar de ser. Mais estudos são necessários tanto para validar os resultados correntes quanto para tentar aprimorá-los. Nesse quesito, em particular, o fato de estudantes que não tinham uma atitude prévia positiva em relação à programação terem evoluído melhor na percepção de seu próprio desempenho como programadores abre uma possibilidade interessante de novos estudos voltados a observar com mais atenção o comportamento desse grupo específico.

Por fim, mesmo ponderando todas as limitações, acreditamos que as diretrizes estão bem fundamentadas e abrem espaço para sua utilização em novos materiais didáticos, tanto no âmbito docente quanto para embasar outros trabalhos de pesquisa. Ademais, o ensino de programação continua sendo um desafio para estudantes e professores, e o material específico utilizado em nosso estudo de caso está disponível online para uso livre e irrestrito de todos os que se dispuserem a enfrentar esse problema.

Referências

Chiu, J.-I. and Tsuei, M. (2020). Meta-analysis of children's learning outcomes in block-based programming courses. In *Proceedings of the 28th International Conference on Computers in Education*, pages 259–266.

Christophel, D. M. (1990). The relationships among teacher immediacy behaviors, student motivation, and learning. *Communication Education*, 39:323–340.

Cárdenas-Cobo, J., Puris, A., Novoa-Hernández, P., Galindo, J. A., and Benavides, D. (2020). Recommender systems and scratch: An integrated approach for enhancing computer programming learning. *IEEE Trans. Learn. Technol.*, 13(2):387–403.

Facey-Shaw, L., Specht, M., van Rosmalen, P., and Bartley-Bryan, J. (2019). Do badges affect intrinsic motivation in introductory programming students? *Simulation & Gaming*, 51:33 – 54.

Farooq, U. and Anwar, S. (2024). Students motivation to learn programming: A systematic review. *2024 IEEE Front. Educ. Conf. (FIE)*, pages 1–9.

Gill, G. and Holton, C. F. (2006). A self-paced introductory programming course. *J. of Information Technology Education: Research*, 5(1):95–105.

Gutiérrez, L. E., Guerrero, C. A., and López-Ospina, H. A. (2022). Ranking of problems and solutions in the teaching and learning of object-oriented programming. *Education and Information Technologies*, 27:7205 – 7239.

Hany, A., Ramadan, E., Akl, A., and Atia, A. (2023). The effect of using tangible user interfaces compared to traditional learning for teaching programming in higher education: An experimental study. *Intell. Methods Syst. Appl. (IMSA)*, pages 514–519.

Hawlitschek, A., Berndt, S., and Schulz, S. (2022). Empirical research on pair programming in higher education: a literature review. *Comp. Sci. Education*, 33:400 – 428.

Hosseini, R., Akhuseyinoglu, K., Brusilovsky, P., Malmi, L., Pollari-Malmi, K., Schunn, C., and Sirkiä, T. (2020). Improving engagement in program construction examples for learning python programming. *Int. J. Artif. Intell. Educ.*, 30:299 – 336.

Koren, M. (2024). Graphical user interfaces as a method to encourage beginners in learning programming. *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, pages 1439–1444.

Mou, T.-Y. (2024). The practice of visual storytelling in stem: Influence of creative thinking training on design students' creative self-efficacy and motivation. *Thinking Skills and Creativity*.

Obaido, G., Agbo, F. J., Alvarado, C., and Oyelere, S. (2023). Analysis of attrition studies within the computer sciences. *IEEE Access*, 11:53736–53748.

Paiva, M. R. F., Parente, J. R. F., Brandão, I. R., and Queiroz, A. H. B. (2016). Metodologias ativas de ensino-aprendizagem: revisão integrativa. *SANARE-Revista de Políticas Públicas*, 15(2).

Papert, S. and Harel, I. (1991). Situating constructionism. *Constructionism*, 36(2):1–11.

Rabelo, A., Maia, L., and Parreiras, F. S. (2018). Performance analysis of computer science students in programming learning. *Anais do WEI'2018*.

Setiawati, L. (2012). A descriptive study on the teacher talk at eyl classroom. *Indonesian Journal of Applied Linguistics*, 1(2):141–152.

Tao, Y. and Chen, G. (2024). The relationship between teacher talk and students' academic achievement: A meta-analysis. *Educational Research Review*.

Wen, F.-H., Wu, T., and Hsu, W. (2023). Toward improving student motivation and performance in introductory programming learning by scratch: The role of achievement emotions. *Science Progress*, 106.

Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., Tan, A. H., Hwa, L., Li, M., and Ko, A. J. (2019). A theory of instruction for introductory programming skills. *Computer Science Education*, 29:205 – 253.

Zataraín-Cabada, R., Estrada, M. L. B., Ríos-Félix, J. M., and Alor-Hernández, G. (2018). A virtual environment for learning computer coding using gamification and emotion recognition. *Interact. Learn. Environ.*, 28:1048 – 1063.

Öztürk, M. (2021). The effect of self-regulated programming learning on undergraduate students' academic performance and motivation. *Interact. Technol. Smart Educ.*, 19:319–337.