

Ampliando a Compreensão de Problemas em Programação: Avaliação de uma Estratégia Educacional Baseada em Teste de *Software*

André Almeida¹, Wilkerson L. Andrade¹, Dalton D. S. Guerrero¹

¹Universidade Federal de Campina Grande, Campina Grande - Paraíba, Brasil

andrealmeida@copin.ufcg.edu.br, wilkerson@computacao.ufcg.edu.br,

dalton@dsc.ufcg.edu.br

Abstract. *Programming requires the ability to convert initial problems into clear, well-defined instructions that produce results, an essential skill in introductory courses. However, beginning students often face difficulties, especially in initially understanding the problems. This study examines the use of a software testing-based strategy that helps clarify problem specifications. We investigated the effectiveness of this approach in improving students' understanding and favoring the correct resolution of exercises. In the empirical study carried out, we observed a significant improvement in student performance, highlighting the proposed strategy as valuable for the process of solving problems in programming.*

Resumo. *Programar requer a habilidade de converter problemas iniciais em instruções claras e bem definidas que produzem resultados, uma competência essencial em cursos introdutórios. No entanto, alunos iniciantes frequentemente enfrentam dificuldades, especialmente na compreensão inicial dos problemas. Este estudo examina o uso de uma estratégia baseada em teste de software que ajuda a esclarecer as especificações dos problemas. Investigamos a eficácia dessa abordagem em melhorar a compreensão dos alunos e favorecer a correta resolução dos exercícios. No estudo empírico realizado, observamos uma melhora significativa no desempenho dos alunos, destacando a estratégia proposta como valiosa para o processo de resolução de problemas em programação.*

1. Introdução

A atividade de programar, transformar um problema inicial em um conjunto de etapas bem definidas que quando executadas produzem um resultado, é uma das competências básicas que alunos de cursos introdutórios de programação precisam desenvolver [Özmen and Altun 2014]. No entanto, antes mesmo de produzir o programa, o aluno precisa compreender bem qual o problema apresentado e o que precisa ser realizado a partir do enunciado do exercício. É necessário entender a especificação como primeiro passo para a efetiva resolução de problemas de programação.

A metodologia de Polya [Polya and Conway 2004] para resolução de problemas consiste em quatro etapas que, quando aplicadas em programação, se traduzem em (1) entender o problema definindo entrada, processamento e saída; (2) planejar a solução escolhendo algoritmos e estruturas de dados adequados; (3) codificar a solução; e (4) revisar

a solução testando e depurando o código para garantir sua correção e eficiência. Este processo pode ser visto como um caminho natural e eficaz para os alunos realizarem suas tarefas de programação. No entanto, todo o processo pode ser muito complexo para os alunos iniciantes enquanto enfrentam suas primeiras experiências de programação, aumentando sua necessidade de orientação e assistência, em especial na primeira fase “entender o problema”, onde os alunos devem identificar os aspectos relevantes da definição do problema e entender o cenário apresentado.

Para atender esta demanda de orientação e assistência aos alunos, além de facilitar a correção dos exercícios, geralmente os instrutores utilizam sistemas de avaliação automática com o intuito de fornecer *feedback* rápido e frequente. No entanto, os alunos ainda precisam de um suporte que vá além da correção funcional dos programas para lidar com a complexidade da programação, pois dificuldades no processo podem impedir que os alunos possam progredir de maneira autônoma ou levar à implementação incorreta da solução do problema [Wrenn and Krishnamurthi 2019].

À luz do exposto e similar ao nosso trabalho anterior [Almeida et al. 2023], esta pesquisa visa lidar com o problema de fornecer suporte no esclarecimento da especificação do problema inspirado na tutoria individual. Este suporte é viabilizado através do emprego de uma estratégia baseada na interação com uma solução de referência (sem acesso ao código) e que permite que o aluno verifique a saída de um problema mediante o fornecimento de uma entrada, ou seja, trabalhando na criação e execução de testes. Dessa forma, o objetivo do presente trabalho é investigar a efetividade dessa abordagem que visa melhor esclarecer os exercícios e, com isso, favorecer a sua correta resolução. Para direcionar a investigação desta pesquisa, foram formuladas as seguintes questões de pesquisa específicas e as correspondentes hipóteses.

- QP1: Os alunos que resolvem os exercícios apoiados pela estratégia apresentam maior taxa de acerto nos exercícios de programação?
 - H1.0: Não há diferença significativa na taxa de acerto nos exercícios de programação entre os alunos que utilizam a estratégia e aqueles que não a utilizam.
 - H1.1: Os alunos que utilizam a estratégia apresentam uma taxa de acerto significativamente maior nos exercícios de programação em comparação com os alunos que não a utilizam.
- QP2: Os alunos que resolvem os exercícios apoiados pela estratégia apresentam redução significativa no tempo de codificação?
 - H2.0: Não há diferença significativa no tempo de codificação entre os alunos que utilizam a estratégia e aqueles que não a utilizam.
 - H2.1: Os alunos que utilizam a estratégia apresentam uma redução significativa no tempo de codificação em comparação com os alunos que não a utilizam.
- QP3: Os alunos que resolvem os exercícios apoiados pela estratégia apresentam redução significativa no número de submissões de suas soluções?
 - H3.0: Não há diferença significativa no número de submissões de soluções entre os alunos que utilizam a estratégia e aqueles que não a utilizam.
 - H3.1: Os alunos que utilizam a estratégia apresentam uma redução significativa no número de submissões de suas soluções em comparação com os alunos que não a utilizam.

O presente trabalho apresenta várias contribuições significativas para o campo da educação em computação. Primeiramente, ele propõe esta estratégia como um suporte no esclarecimento de especificações de problemas. Em segundo lugar, oferece uma análise detalhada da efetividade da mesma, fornecendo dados empíricos sobre seu impacto no desempenho dos alunos. Além disso, a pesquisa contribui para a literatura ao explorar a relação entre o uso de ferramentas automatizadas e a melhoria na compreensão e resolução de exercícios de programação.

2. Trabalhos Relacionados

Nesta seção, revisamos alguns dos trabalhos relacionados mais relevantes, que fornecem um contexto para nossa pesquisa e destacam diferentes métodos e ferramentas que têm sido utilizados para melhorar o processo de ensino e aprendizagem de programação através de testes de *software*, os quais vão ao encontro do mapeamento sistemático realizado por Feitosa et al. [Feitosa et al. 2021] e também contribuem para a revisão.

Basu et al. [Basu et al. 2015] introduzem uma abordagem aplicada em MOOCs (*Massive Open Online Courses*), onde propõem o Sistema OK. Este sistema automatiza a avaliação de exercícios de programação usando suítes de testes para verificar as soluções em diversos cenários. Além disso, os autores apresentam um recurso que permite aos alunos validar sua compreensão dos exercícios. Por meio de um processo interativo, os alunos respondem a perguntas curtas sobre o comportamento esperado do código, desbloqueando casos de teste ao responder corretamente, o que lhes permite testar suas soluções antes da submissão final.

A estratégia de Denny et al. [Denny et al. 2019] é fortemente baseada no *Test Driven Development* (TDD) [Beck 2003] e incorpora este método no processo de desenvolvimento dos alunos. Um dos principais benefícios do TDD é incentivar o programador a considerar casos de entrada especiais que podem surgir. Além disso, pedir aos alunos que desenvolvam casos de teste primeiro pode ajudar a melhorar sua compreensão do problema em questão. Uma das ferramentas de avaliação automatizada mais populares, o WebCAT [Edwards and Perez-Quinones 2008], foi desenvolvida precisamente para integrar o TDD no contexto educacional dos alunos.

Baseando-se no IPO Model [Grady 1995], uma abordagem amplamente utilizada para modelar estruturas de programação em três etapas - identificar as entradas, realizar o processamento e obter as saídas - Cabo [Cabo 2019] busca quantificar o entendimento de problemas e a performance dos alunos em um curso introdutório de *Problem Solving* utilizando Python. No estudo, os alunos recebem um problema de programação, sugerem três possíveis entradas e encontram as saídas manualmente (ou com uma calculadora). Posteriormente, codificam suas soluções. A ideia é verificar se a dificuldade de escrever programas corretos é identificada antes da codificação e propor a extensão do entendimento dos problemas por meio da definição de entradas e saídas.

Wrenn e Krishnamurthi [Wrenn and Krishnamurthi 2019] propõem um mecanismo que oferece *feedback* instantâneo aos alunos sobre se eles exploraram completamente um problema, independentemente do progresso da implementação. Nesta abordagem, os alunos fornecem tanto a entrada quanto a saída esperada para os problemas, ajudando a verificar a compreensão dos mesmos antes da codificação. Posteriormente, os autores investigam quais dúvidas persistem e a influência do *feedback* automático

[Wrenn and Krishnamurthi 2021]. Eles observam benefícios, mas também que alguns alunos testam exaustivamente todas as entradas e saídas sem a devida compreensão. Ao final, concluem que a compreensão de problemas exige novas habilidades não bem abordadas nos currículos atuais, e que o *feedback* automatizado não elimina as perguntas dos alunos sobre o comportamento esperado dos problemas.

Em nosso trabalho mais recente [Almeida et al. 2023], investigamos a efetividade de uma estratégia muito semelhante à ferramenta apresentada por Wrenn e Krishnamurthi, onde os alunos podem validar seu entendimento interagindo com uma solução de referência por meio da proposição de pares entrada-saída, recebendo um *feedback* de correto ou incorreto. No entanto, verificamos que muitos dos participantes tinham dificuldades em propor corretamente o par entrada-saída e ainda permaneciam com dúvidas sobre as especificações dos problemas. Portanto, neste trabalho, incluímos a possibilidade de verificar o gabarito dos exercícios mediante o fornecimento de entradas, ou seja, os alunos fornecem uma entrada para o problema e recebem como resposta a saída esperada, aprimorando assim nosso trabalho anterior com uma nova forma de interação.

Diferentemente das abordagens anteriores que focam na validação de casos de teste por meio de um processo iterativo, esta pesquisa propõe uma interação mais profunda e detalhada com a solução de referência. Essa nova funcionalidade visa mitigar as dificuldades encontradas por alunos ao propor pares entrada-saída, ao mesmo tempo em que melhora a compreensão das especificações dos problemas.

3. Metodologia

A metodologia deste estudo foi cuidadosamente planejada e executada para garantir a validade dos resultados. A seguir, descrevemos detalhadamente o desenho experimental, os materiais e os instrumentos utilizados.

3.1. Desenho Experimental

O público-alvo deste estudo consiste em alunos matriculados em disciplinas introdutórias de programação. Esses alunos estão no início de sua formação e possuem diferentes níveis de familiaridade com conceitos de programação. Um exemplo de ementa de uma disciplina introdutória de programação, que inclui os tópicos abordados (geralmente em linguagem Python), está apresentado na Tabela 1.

Tabela 1. Ementa da disciplina de Introdução à Ciência da Computação.

Disciplina:	Introdução à Ciência da Computação
Ementa:	Informática Básica. Arquitetura básica de um Computador. Evolução dos Computadores. Aplicações. Sistemas de Numeração e Representação de Dados. Introdução à Lógica de Programação. Estrutura Lógica de um Algoritmo. Estrutura de Decisão. Estrutura de Repetição. Modularização. PseudoLinguagem. Metodologia de Desenvolvimento de Algoritmos.

O estudo adotou um desenho experimental controlado, comparando dois grupos: o grupo de controle e o grupo experimental. Para ambos os grupos, foram apresentados quatro exercícios de programação, com um tempo máximo de duas horas para realizá-los, no formato de miniteste. Esses exercícios foram aplicados somente após a apresentação formal dos conteúdos sobre estruturas de decisão e estruturas de repetição em Python.

O aspecto que diferenciou o grupo de controle e o grupo experimental foi a utilização de nossa estratégia, a qual foi utilizada apenas pelo grupo experimental. Com esta ferramenta o aluno tem a possibilidade de interagir com uma solução de referência sem visualizar o código, quantas vezes julgar necessário, enviando uma entrada para um problema e verificando qual a saída esperada (ver Figura 1). O grupo de controle seguiu a forma mais tradicional de resolução de exercícios de programação, na qual geralmente o aluno passa para a codificação após ler o enunciado e posteriormente submete sua solução ao sistema de avaliação.

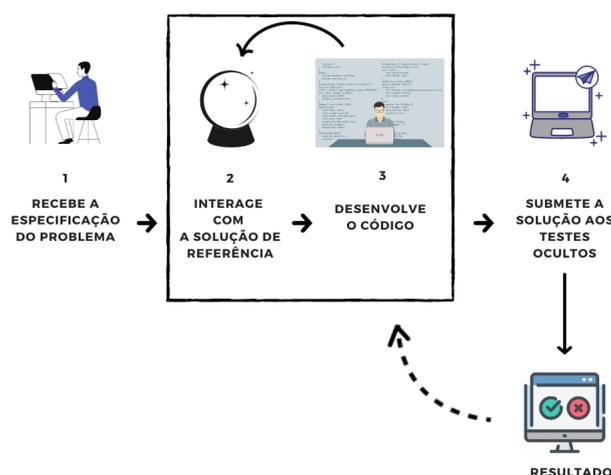


Figura 1. Processo de resolução com a inclusão da estratégia proposta.

Para avaliar as questões de pesquisa propostas nesse estudo, procuramos observar as seguintes variáveis, que são comumente coletadas em ambientes de submissão de atividades de programação.

- Taxa de acerto: a distribuição da quantidade de alunos que acertaram um determinado número de exercícios.
- Tempo de resolução dos exercícios: duração (em minutos) entre o momento em que o aluno recebe a especificação (acessa o exercício) e a submissão da última solução ao sistema.
- Número de submissões do código: quantidade de vezes que o aluno verifica se sua solução é aceita pelos testes ocultos criados pelo professor.

3.2. Seleção dos Exercícios

Os exercícios aplicados neste estudo foram selecionados com base em critérios específicos para garantir a adequação e relevância das atividades em relação aos objetivos de pesquisa. Primeiramente, foram consideradas as habilidades de programação que os exercícios deveriam avaliar, tais como o emprego de estruturas condicionais e estruturas de repetição em Python, assegurando uma cobertura abrangente dos principais conceitos e técnicas.

Posteriormente, os exercícios foram selecionados como base em nível de dificuldade e tempo necessário para resolução, devido ao tempo total estipulado para realização do estudo (duas horas). Dessa forma, utilizamos como referência os exercícios propostos no *URI Online Judge - beecrowd*¹, uma vez que estão organizados por assunto, nível

¹<https://www.becrowd.com.br/judge/en/login>

de dificuldade e possuem enunciados detalhados sobre cada problema. Selecionamos os exercícios ‘Quadrado de Pares’², ‘Múltiplos de 13’³, ‘Intervalo’⁴ e ‘Aumento de Salário’⁵ para aplicar em ambos os grupos, de controle e experimental.

Por fim, foram propostos alguns exercícios no mesmo formato utilizado na plataforma aos participantes, antes do início do experimento e sem contabilização das métricas, para que se familiarizassem com o estilo de enunciado adotado durante o experimento, que geralmente consiste em uma parte textual acompanhada de exemplos de entrada e saída que ilustram o comportamento esperado.

3.3. Plataformas Utilizadas

Com os exercícios selecionados, partimos para a definição e preparação dos ambientes que os alunos utilizariam para aplicar os procedimentos propostos pelo estudo e coletar as variáveis de interesse. Como ambiente de submissão de atividades adotamos o OnlineGDB⁶, um compilador online que permite a criação de salas de aula e cadastro de exercícios com avaliação automática baseada em casos de teste.

No que se refere a abordagem alvo desta pesquisa, desenvolvemos uma ferramenta capaz de realizar o comportamento esperado: possuindo uma solução de referência, a ferramenta oferece uma interface que permite que o usuário forneça uma entrada para determinado problema e tenha a resposta, ou seja, a saída correta que é esperada para aquela entrada fornecida. Na Figura 2 são apresentadas as interfaces acessadas para a realização desse processo.

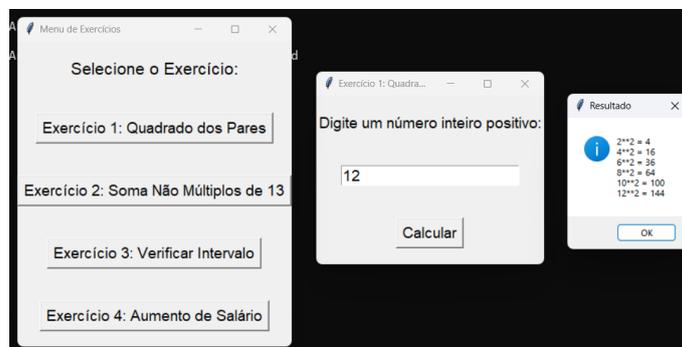


Figura 2. Menu e caixas de interação com a ferramenta.

4. Resultados e Discussão

Participaram efetivamente 28 alunos, 50% destes compondo o grupo de controle e 50% participando do grupo experimental, distribuídos aleatoriamente. Ambos os grupos tiveram o mesmo tempo para resolução dos quatro exercícios propostos, com o acesso ao ferramental (Figura 2) incluído no processo de resolução apenas no grupo experimental.

Nesta seção, apresentamos os principais achados da pesquisa e discutimos suas implicações. Para analisar os dados, foram empregados diversos testes estatísticos. O

²<https://judge.beecrowd.com/pt/problems/view/1073>

³<https://judge.beecrowd.com/pt/problems/view/1132>

⁴<https://judge.beecrowd.com/pt/problems/view/1037>

⁵<https://judge.beecrowd.com/pt/problems/view/1048>

⁶<https://www.onlinegdb.com/>

teste de qui-quadrado foi utilizado para verificar se havia uma diferença significativa nas taxas de acerto entre os grupos. O teste de Mann-Whitney foi aplicado para comparar os tempos de resolução entre os grupos, uma vez que os dados não seguiram uma distribuição normal, conforme indicado pelo teste de Shapiro-Wilk. Para verificar a correlação entre a taxa de acertos e o tempo de resolução, foi utilizado o coeficiente de correlação de Spearman. Todas as análises foram realizadas utilizando a linguagem Python e as bibliotecas estatísticas associadas, permitindo uma avaliação detalhada das variáveis coletadas e das hipóteses formuladas, conforme descrito a seguir.

4.1. Taxa de Acerto (TA)

A taxa de acerto tem como objetivo principal medir o desempenho dos participantes no contexto do estudo, mensurando a distribuição da quantidade de alunos que acertaram um determinado número de exercícios.

Na Figura 3 é possível observar, por exemplo, que aproximadamente 79% dos alunos pertencentes ao grupo de controle acertaram apenas de 1 a 2 exercícios dos quatro que foram propostos, enquanto que aproximadamente 79% do grupo experimental de 3 a 4 exercícios.

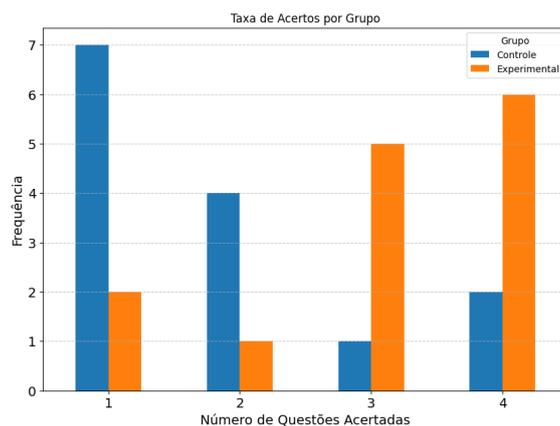


Figura 3. Distribuição da quantidade de alunos sobre o número de exercícios corretos.

Aplicando o teste de qui-quadrado para verificar se há uma diferença significativa na distribuição das respostas corretas entre os dois grupos e a hipótese estabelecida, considerando um nível de significância de 0.05, obtivemos um p-valor = 0.026, o que permite rejeitar a hipótese nula e afirmar que os alunos que utilizam a estratégia apresentam uma taxa de acerto significativamente maior nos exercícios de programação em comparação com os alunos que não a utilizam.

4.2. Tempo de Resolução (T)

Neste ponto analisamos a duração entre o momento em que o aluno recebe a especificação (acessa o exercício) e a submissão da última solução ao sistema, para ambos os grupos. O objetivo é verificar se os alunos do grupo experimental gastam menos tempo em suas soluções, visto que utilizaram uma ferramenta para verificar se compreenderam de fato os enunciados dos exercícios.

No gráfico de boxplots a seguir (Figura 4) são apresentados os resultados por grupo. É possível observar que a mediana do tempo para o grupo de controle (20,5 minutos) é razoavelmente maior que a mediana do grupo experimental (18 minutos). Outra visualização que contribui para a análise é a Figura 5, na qual se torna mais evidente, agora por exercício, o tempo de resolução. A diferença na mediana, para os exercícios 1 e 2 especialmente, se torna mais evidente. Isto também revela que a adoção da estratégia não chega a impactar negativamente nesta variável, uma vez que se trata de mais uma ferramenta que os alunos precisam utilizar dentro do seu processo de resolução de problemas e que poderia facilmente acarretar em um crescimento considerável da duração da escrita das soluções

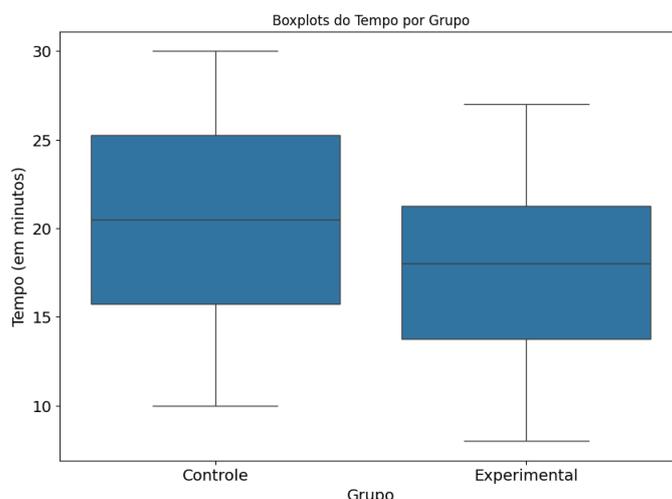


Figura 4. Distribuição do tempo para cada grupo.

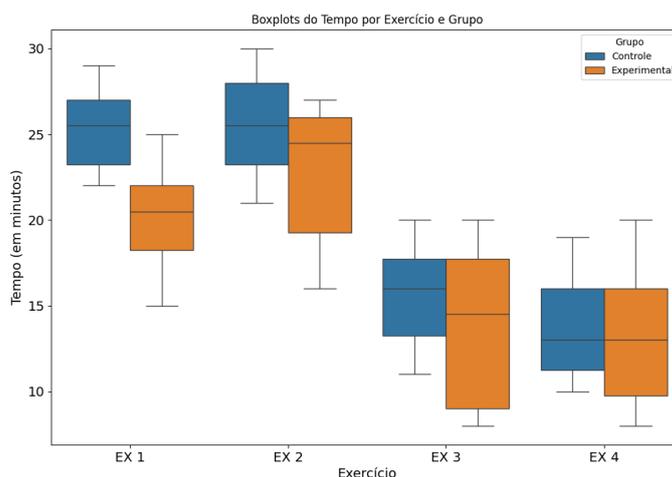


Figura 5. Distribuição do tempo por exercício e grupo.

A fim de verificar nossa hipótese, realizamos o teste de Mann-Whitney com nível de significância 0.05 e obtivemos um p-valor = 0.042, o que permite que a hipótese nula seja rejeitada e que possamos afirmar, também com base nas medianas, que os alunos que utilizam a estratégia apresentam uma redução significativa no tempo de codificação em comparação com os alunos que não a utilizam.

4.3. Número de Submissões (NSub)

Ao analisar essa variável, objetivamos verificar a frequência com que os alunos verificam se sua solução é aceita pelos testes ocultos criados pelo professor. Isso nos permite investigar se os alunos do grupo experimental realizam esse processo menos vezes devido ao suporte proporcionado pela estratégia.

As Figuras 6 e 7 apresentam a distribuição do número de submissões por grupo e também por exercício. Observamos que a mediana e o número máximo de submissões é consideravelmente menor para o grupo experimental, se tornando mais evidente quando visualizado por exercício.

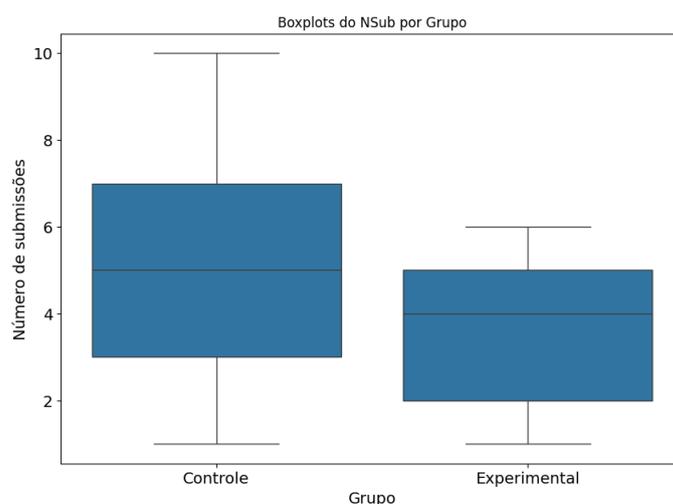


Figura 6. Distribuição do NSub para cada grupo.

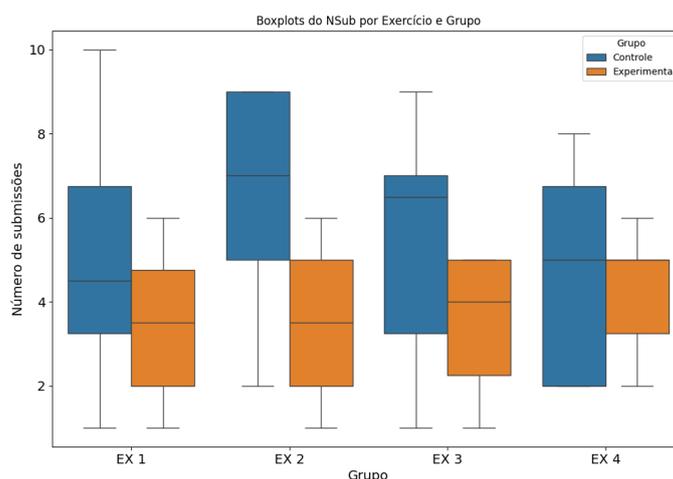


Figura 7. Distribuição do NSub por exercício e grupo.

Para verificar a hipótese relacionada a essa variável, realizamos também o teste de Mann-Whitney com nível de significância 0.05 e obtivemos um p-valor = 0.0003, o que permite que a hipótese nula seja rejeitada e que possamos afirmar, também com base nas medianas, que os alunos que utilizam a estratégia apresentam uma redução significativa no número de submissões de suas soluções em comparação com os alunos que não a utilizam.

4.4. Correlação entre T e NSub

Neste ponto, exploramos a relação entre duas variáveis-chave do estudo: o tempo gasto na resolução dos exercícios (T) e o número de submissões realizadas pelos participantes (NSub). Ao investigar a natureza dessa relação, buscamos compreender como o tempo investido impacta diretamente na eficiência das tentativas de solução e na busca por otimização dos resultados obtidos.

O gráfico de dispersão da Figura 8 ilustra a relação entre as duas variáveis de interesse, permitindo a visualização de cada observação presente nos dados de ambos os grupos participantes do estudo. Após utilizar o teste de Shapiro-Wilk e verificar que os dados não seguem uma distribuição normal, foi calculado o coeficiente de correlação de Spearman e obtivemos $\rho = 0.14$ e p-valor = 0.293 para o grupo de controle, enquanto que para o grupo experimental obtivemos $\rho = -0.11$ e p-valor = 0.384.

Também considerando um nível de significância de 0.05, é possível concluir que as correlações calculadas são não significativas e ambas de força fraca. No entanto, a direção foi identificada como positiva para o grupo de controle e negativa para o grupo experimental. Em outras palavras, enquanto a variável NSub tende a crescer em relação ao tempo T para o grupo de controle, no grupo experimental ocorre o oposto: a variável NSub apresenta uma leve tendência a diminuir com o passar do tempo T. Isso sugere que, embora os alunos do grupo experimental gastem um tempo semelhante na resolução das questões, eles acabam fazendo um número menor de submissões, possivelmente devido a adoção da estratégia, que é o que diferencia um grupo do outro.

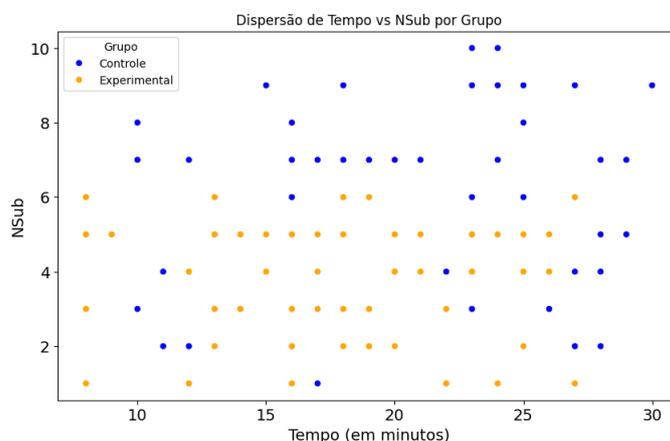


Figura 8. Dispersão das variáveis T e NSub por grupo.

4.5. Demais Observações

Como forma de estender a discussão, podemos realizar a análise dos resultados através de outra visualização, que consiste na taxa de acerto (em percentual) de cada exercício para cada grupo do estudo.

Na Figura 9 é observável que existe um padrão em relação a taxa de acerto em ambos os grupos, nos exercícios 2 e 4 acontece uma declínio considerável na taxa de acerto para estes se compararmos com os demais exercícios. No entanto, de forma geral, a taxa de acerto para o grupo experimental se mostra superior nos quatro, sendo o exercício 3 o que possui maior aproximação da taxa de acerto entre os grupos.

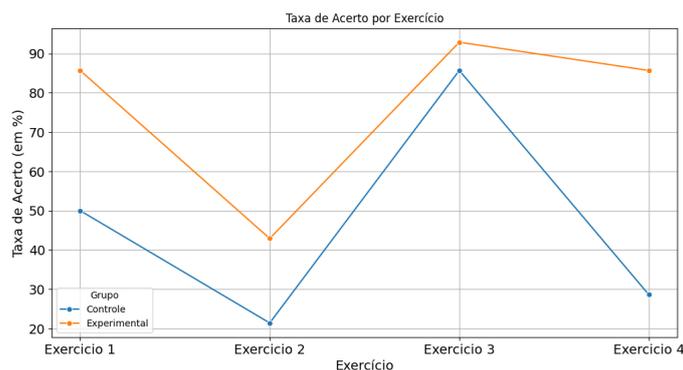


Figura 9. Taxa de acerto (em %) por exercício.

Durante a realização do estudo com cada grupo, diversos aspectos foram observados e os mais relevantes são elencados a seguir:

- Os exercícios 1 e 2 possuíam uma complexidade relativamente maior que os demais, por utilizar estruturas de repetição em conjunto com estruturas condicionais, o que culmina em taxas de acerto relativamente menores;
- Foram detectadas dificuldades em utilizar a função `range()` para “navegar” e realizar as devidas manipulações dentro de um intervalo, o que se fazia necessário nos exercícios 1 e 2;
- Alguns alunos sentem dificuldade em exibir a saída de forma igual ao apresentado no enunciado, o que é algo natural em disciplinas introdutórias de programação, mas que pode influenciar nas variáveis T e possivelmente gerar um maior NSub;
- Dificuldades básicas em cálculos de porcentagem a depender da organização das estruturas condicionais, comportamento necessário para a realização do exercício 3;
- Os alunos que não tiveram acesso à estratégia demonstraram uma frequência significativamente maior de solicitações ao professor responsável pelo experimento para investigar o que havia de errado em seus códigos, visto que o *feedback* dos testes ocultos pouco ajudavam na construção da solução. Em contrapartida, os alunos que utilizaram a ferramenta solicitaram a ajuda do professor em poucas ocasiões e mantiveram-se mais atentos aos detalhes do enunciado, levando mais tempo para iniciar a escrita de suas soluções. Isso porque verificaram o entendimento dos exercícios através da estratégia tanto antes quanto durante a fase de codificação.

5. Conclusão e Trabalhos Futuros

Este artigo analisa a efetividade de uma estratégia baseada em testes de *software* para melhorar o entendimento de enunciados de exercícios de programação. A hipótese é que os alunos que refletem sobre o problema, ampliando a compreensão sobre o seu enunciado, à medida que interagem com uma solução de referência, raciocinam, constroem e propõem soluções melhores.

A ferramenta proposta foi desenvolvida de tal forma que o “teste de entendimento” pudesse ser realizado e incluído no processo de resolução de problemas. Dada a análise estatística realizada para as variáveis de tempo de resolução, número de submissões e taxa

de acerto, podemos concluir que houve diferenças estatisticamente significativas entre os grupos experimentais. O teste de hipótese revelou que o tempo de resolução foi significativamente menor para os estudantes que utilizaram a estratégia. Além disso, o número de submissões mostrou uma redução significativa no grupo experimental, indicando que o uso do ferramental pode ter impactado positivamente na eficiência dos alunos em encontrar a solução correta com menos iterações. Quanto à taxa de acerto, observou-se uma melhoria estatisticamente significativa nos alunos que utilizaram a estratégia, sugerindo que esta pode ter contribuído para uma melhor compreensão e aplicação dos conceitos abordados nos exercícios de programação. Esses resultados indicam que o uso da estratégia proposta pode ser um meio eficaz para melhorar o desempenho e a eficiência dos alunos em cursos de programação introdutória.

Observar como os alunos interagiram com a ferramenta revela que, apesar das dificuldades que vão além da compreensão do problema, como manipulações matemáticas, conformidade com as saídas esperadas e sintaxe da linguagem, eles demonstram maior confiança ao tentar resolver os problemas, especialmente os mais tímidos. Para trabalhos futuros, planejamos realizar estudos adicionais para obter resultados mais robustos e confiáveis, analisando detalhadamente os benefícios dessa estratégia. Será que um número limitado de interações com as soluções de referência afeta o desempenho dos alunos em comparação com um cenário sem limitações? Os testes realizados pelos alunos por meio da ferramenta realmente abrangem diferentes cenários para um problema? Pretendemos investigar esses e outros aspectos, além de coletar *feedback* dos alunos.

6. Ameaças à Validade

Vale ressaltar alguns potenciais desafios que podem comprometer a validade dos resultados encontrados. Uma dos principais é o viés de seleção, dado que a amostra utilizada pode não ser totalmente representativa da população-alvo, o que limita a generalização dos resultados. Estudos que envolvem indivíduos naturalmente apresentam o viés relacionado às características específicas desses participantes, como o nível de conhecimento prévio. Portanto, não podemos estabelecer qualquer relação de causalidade com a população geral dos aprendizes de programação introdutória.

Os participantes foram selecionados e distribuídos de forma aleatória. Essa abordagem visou garantir a heterogeneidade da amostra, permitindo uma avaliação mais ampla da eficácia da estratégia proposta em diferentes níveis de familiaridade com a programação. No entanto, essa aleatoriedade pode ter gerado vantagens e desvantagens para ambos os grupos, controle e experimental. Portanto, é fundamental considerar a importância de realizar um balanceamento nos grupos para que os resultados possam ser comparados de maneira mais precisa.

Ainda podemos mencionar o tempo limite do estudo que pode ter impactado no processo reflexivo dos alunos e na qualidade das respostas, bem como a necessidade de utilizar mais de uma ferramenta durante a resolução dos exercícios, produzindo uma carga de trabalho maior e diferente do processo ao qual estavam habituados.

Referências

Almeida, A., Araújo, E., and Figueiredo, J. (2023). Investigando o uso de testes para apoiar a resolução de problemas de programação. In *Anais do XXXIV Simpósio Brasi-*

- leiro de Informática na Educação*, pages 357–367, Porto Alegre, RS, Brasil. SBC.
- Basu, S., Wu, A., Hou, B., and DeNero, J. (2015). Problems before solutions: Automated problem clarification at scale. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*, pages 205–213.
- Beck, K. (2003). *Test-driven development: by example*. Addison-Wesley Professional.
- Cabo, C. (2019). Fostering problem understanding as a precursor to problem-solving in computer programming. In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.
- Denny, P., Prather, J., Becker, B. A., Albrecht, Z., Loksa, D., and Pettit, R. (2019). A closer look at metacognitive scaffolding: Solving test cases before programming. In *Proceedings of the 19th Koli Calling international conference on computing education research*, pages 1–10.
- Edwards, S. H. and Perez-Quinones, M. A. (2008). Web-cat: automatically grading programming assignments. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 328–328.
- Feitosa, Y. R. G., Silva, M. A. G., and Fabri, J. A. (2021). Mapeamento sistemático sobre resolução de problemas em disciplina introdutória de programação com teste de software. In *Anais do XXIX Workshop sobre Educação em Computação*, pages 358–367. SBC.
- Grady, J. O. (1995). *System engineering planning and enterprise identity*, volume 7. Crc Press.
- Özmen, B. and Altun, A. (2014). Undergraduate students’ experiences in programming: difficulties and obstacles. *Turkish Online Journal of Qualitative Inquiry*, 5(3):1–27.
- Polya, G. and Conway, J. (2004). *How to Solve It: A New Aspect of Mathematical Method*. Penguin mathematics. Princeton University Press.
- Wrenn, J. and Krishnamurthi, S. (2019). Executable examples for programming problem comprehension. In *Proceedings of the 2019 ACM Conference on International Computing Education Research, ICER ’19*, page 131–139, New York, NY, USA. Association for Computing Machinery.
- Wrenn, J. and Krishnamurthi, S. (2021). Reading between the lines: Student help-seeking for (un) specified behaviors. In *Proceedings of the 21st Koli Calling International Conference on Computing Education Research*, pages 1–6.