Análise de Modelos Mentais e Equívocos no Aprendizado de Funções de Ordem Superior: Um Relato de Experiência com Simulação Visual

Marcos Rogério Martins[1],**Rodrigo Silva Duran**[2]

[1]Universidade Estadual de Feira de Santana (UEFS)
Feira de Santana – BA – Brasil

[2]Instituto Federal de Brasília (IFB)
Taguatinga – DF – Brasil

martinsmrogerio@gmail.com, rodrigo.duran@ifb.edu.br

Abstract. *The demand for programming skills from non-developers, especially in data-related tasks, has increased considerably in recent times. Higher-Order Functions (HOFs) have emerged as popular tools in this context, but learners still face challenges in fully understanding these constructs. This work presents an experience report on the asynchronous use of a visual simulation tool adapted for teaching HOFs. Given a scenario of autonomous and asynchronous learning, the study adopted a mixed-methods approach to map how learners interact with the visualizations and build their mental models. Although quantitative tests did not reveal significant disparities in overall performance between the primary groups analyzed, detailed analysis of the responses allowed for the identification and categorization of common conceptual errors and conflicting mental models adopted by the students. The results reveal specific error patterns linked to the type of visual support used, offering practical insights for the design of pedagogical materials and tools aimed at teaching functional programming within the computer science in education track.*

Resumo. *A demanda por programação por não desenvolvedores, especialmente em tarefas relacionadas a dados, aumentou consideravelmente nos últimos tempos. As Funções de Ordem Superior (FOS) surgiram como ferramentas populares nesse contexto, mas os aprendizes ainda enfrentam desafios na compreensão desses constructos. Este trabalho apresenta um relato de experiência sobre o uso assíncrono de uma ferramenta de simulação visual adaptada para o ensino de FOS. Diante de um cenário de aprendizado autônomo e assíncrono, o estudo adotou uma abordagem mista para mapear como os aprendizes interagem com as visualizações e constroem seus modelos mentais. Embora os testes quantitativos não tenham apontado disparidades significativas de desempenho geral entre os principais grupos analisados, a análise detalhada das respostas permitiu identificar e categorizar equívocos conceituais comuns e modelos mentais conflitantes adotados pelos estudantes. Os resultados revelam padrões específicos de erro atrelados ao tipo de suporte visual utilizado, oferecendo subsídios práticos para o design de materiais pedagógicos e ferramentas voltadas ao ensino de programação funcional na trilha de Computação na Educação.*

DOI: 10.5753/wie.2026.27810

1. Introdução

A demanda por competências em programação expandiu-se para além do desenvolvimento tradicional de software, abrangendo profissionais não desenvolvedores de diversas áreas [Scaffidi et al. 2005, Guzdial and Shreiner 2021]. Esse cenário impulsionou iniciativas "CS + X" [Brodley et al. 2022], pautadas na Centralidade de Dados [Krishnamurthi and Fisler 2020], onde o foco desloca-se do controle de fluxo tradicional para a transformação e limpeza de dados.

Para trabalhar com dados com relacionamentos complexos em grande escala é fundamental prover ferramentas computacionais eficientes que possam realizar tarefas de forma simples. As Funções de Ordem Superior (FOS), funções que operam em outras funções, seja tomando-as como argumentos ou retornando-as como resultado, emergem como uma solução para essas demandas, pois possuem uma grande capacidade de composição mantendo o código conciso. Linguagens modernas como Java, JavaScript e R incorporam amplamente estas funcionalidades oriundas da programação funcional. Dentre as FOS mais utilizadas, destacam-se filter(), map() e reduce(). Essas funções filtram, transformam e agregam dados, simplificando tarefas comuns na análise de dados e potencialmente reduzindo a carga cognitiva do desenvolvedor [Duran et al. 2018].

Contudo, pouca pesquisa tem sido dedicada ao aprendizado ou à compreensão de estudantes sobre *FOS*. Estudos recentes mostram que estudantes conseguem compreender os objetivos de uma *FOS*, reconhecê-las individualmente e utilizá-las para planejar soluções corretas para certos problemas de processamentos de dados [Krishnamurthi and Fisler 2021]. Entretanto, os estudantes ainda têm dificuldades na compreensão de funções como o reduce() [Krishnamurthi and Fisler 2021] e na composição de funções [Rivera et al. 2022].

Dificuldades semelhantes na composição e compreensão de código já foram observadas no passado. Para mitigar tais dificuldades, estudos sugerem que a visualização e simulação de programas podem ser eficazes e melhorar o processo de compreensão de programas [Sorva and Sirkiä 2010, Sorva et al. 2012] ao tornar concreta a *Notional Machine* [Boulay 1986, Sorva et al. 2012], uma abstração pedagógica com determinado nível de detalhamento da semântica de um programa em execução [Duran et al. 2021]. Embora ferramentas como JSVEE [Sirkiä 2013] e Python Tutor [Guo 2013] ofereçam recursos para visualizar a execução de um programa, o JSVEE não possui suporte explícito a FOS, enquanto o Python Tutor oferece uma visualização limitada da pilha de chamadas e carece de elementos visuais para representar relações entre funções e argumentos.

Diante disso, este trabalho apresenta um relato de experiência focado no desenvolvimento e na aplicação de extensões visuais para o simulador JSVEE, projetadas especificamente para apoiar o ensino de FOS. Esta investigação consolida e estende a proposta preliminar de design e simulação de FOS apresentada em [Martins and Duran 2023], executando o estudo empírico com aprendizes de programação. Avaliou-se, por meio de um estudo piloto exploratório, como diferentes designs de simulação impactam a construção de modelos mentais dos estudantes, mapeando os principais equívocos (*misconceptions*) cometidos no processo, de forma a extrair lições para o refinamento de designs subsquentes. Especificamente, as seguintes questões de pesquisa foram abordadas:

1. **QP1**. Quais as potenciais características de design que podem ser utilizadas na

simulação e nas representações visuais de Funções de Ordem Superior tais como o Filter, Map e Reduce?
2. **QP2**. A partir dessas características de design, de que maneira elas auxiliam os estudantes na identificação e superação de equívocos conceituais ao analisarem programas utilizando Funções de Ordem Superior escritos em Python?

2. Revisão de Literatura

As Funções de Ordem Superior (FOS) destacam-se como um conceito-chave, tanto na programação funcional quanto na Ciência de Dados [Krishnamurthi and Fisler 2020]. Ao permitir a manipulação de dados de forma concisa e expressiva, as FOS contribuem para a escrita de código mais eficiente e a resolução de problemas complexos [Duran et al. 2018].

[Krishnamurthi and Fisler 2021] mostram que, mesmo correlacionando os exemplos de entrada e saída à *FOS*, os alunos enfrentaram dificuldades em classificar e compreender a aplicabilidade de algumas funções, especialmente o reduce(). [Rivera et al. 2022] analisaram a capacidade dos alunos de planejar e implementar soluções utilizando a composição de filter(), map() e reduce(). Seus resultados indicaram que os estudantes conseguiram reconhecer *FOS* individuais e construir planos de solução, mas apresentaram dificuldades ao lidar com composições mais complexas. Com base nesses estudos, [Rivera and Krishnamurthi 2022] investigaram dois padrões de composições de funções (estrutural e pipeline), identificando as vantagens e desafios de cada abordagem. Esses trabalhos demonstram o potencial das FOS como ferramentas de ensino para auxiliar na compreensão de conceitos de programação funcional, mas também indicam que os alunos apresentam dificuldades em compreender o funcionamento e a aplicação das FOS, especialmente a composição de funções.

Para superar esses desafios no aprendizado de FOS, a literatura sugere como alternativa pedagógica o uso de sistemas de visualização e simulação, que podem auxiliar na compreensão de conceitos abstratos e na detecção de erros. Instrutores frequentemente utilizam algum tipo de visualização em seus cursos [Naps et al. 2002], e a visualização de uma *Notional Machine* pode tornar processos e tempo de execução ocultos em focais, explícitos e controláveis, servindo como modelos conceituais que ajudam os alunos a construir um conhecimento de programação possível [Naps et al. 2002]. Sistemas como o Jeliot [Eliot et al. 1997, Levy et al. 2003] e o Python Tutor [Guo 2013] demonstram a importância da visualização para o ensino de programação. Embora tenham contribuído significativamente para o campo, essas ferramentas apresentam limitações em termos de complexidade, suporte a linguagens e paradigmas de programação, e capacidade de visualizar conceitos mais avançados tais como FOS.

O JSVEE [Sirkiä 2016] surge como uma ferramenta de simulação oferecendo uma biblioteca JavaScript para criar animações personalizadas de programas independentemente da linguagem de programação, incluindo a avaliação em nível de expressões [Sorva et al. 2012]. Embora possua resultados promissores, principalmente quando aliada a ferramentas que promovam o engajamento ativo com a visualização [Sirkiä 2016], o JSVEE ainda apresenta desafios de usabilidade [Sirkiä et al. 2017].

Para aprimorar aspectos de usabilidade, [Mutlu-Bayraktar et al. 2019] afirmam que a estrutura da cognição humana deve ser considerada no design de sis-

temas multimídia. Portanto, ao projetar animações ou visualizações, os designers instrucionais devem considerar como os alunos processam novas informações e como técnicas de design específicas facilitam o aprendizado [Renkl and Scheiter 2017, Schroeder and Cenkci 2018]. [Duran et al. 2022] enfatizam como a Teoria da Carga Cognitiva (TCC), a qual destaca a necessidade de controlar a carga cognitiva de materiais instrucionais de forma que não sobrecarreguem a capacidade de processamento cognitivo do estudante, pode ser utilizada como guia para design de sistemas de visualização e simulação de código. De acordo com a TCC, a memória de trabalho é pequena e possui curta duração. Portanto, qualquer informação processada deve estar dentro de seus limites. Como exemplo, a TCC aponta que resolver cálculos exige menos esforço mental quando utilizamos artefatos auxiliares para armazenar resultados parciais, como uma folha de papel, principalmente quando a expressão aritmética envolve vários elementos e lembrar os resultados parciais da expressão torna-se parte da tarefa [Beckmann 2010].

Algumas estratégias de controle de carga cognitiva consistem em apresentar explicitamente os passos envolvidos na tarefa como soluções prontas, tais como os chamados Worked Examples [Muldner et al. 2022], a exemplo da *Notional Machine* utilizada pelo JSVEE. Designers instrucionais também devem observar aspectos relacionados com Pistas Visuais [Wang et al. 2020], "dicas, cores, palavras e setas, ou informações adicionadas que podem fornecer andaimes visuais" e ajudar a direcionar a atenção dos usuários para os aspectos que são importantes nos materiais de aprendizagem e ajudar a orientar os processos cognitivos [Mayer 1999]. [de Koning et al. 2010] atribuem os efeitos das pistas visuais às limitações perceptivas que permitem aos humanos concentrar-se em uma pequena parte da exibição visual de cada vez com base nas propriedades do elemento, como contraste visuoespacial (cor) e contraste dinâmico (movimento repentino). [Kong et al. 2017] argumentam que estas técnicas reduzem as buscas visuais e, portanto, "menos recursos visuoespaciais são necessários para direcionar a execução dos movimentos oculares".

3. Materiais

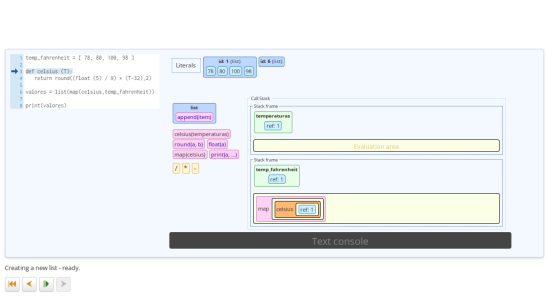
Uma vez que a concepção da visualização utilizada e dos materiais de apoio e avaliação são essenciais para a caracterização do estudo, apresentaremos estes elementos com algum detalhamento. Para mais informações, todos os materiais estão disponíveis para consulta online.[1]. O JSVEE foi escolhido como sistema de visualização por fornecer uma *Notional Machine* robusta e flexível, permitindo a criação de novos modelos de execução e fornecendo detalhes do funcionamento de FOS que exploram os designs propostos.

Para investigar o impacto de ferramentas de visualização na compreensão de FOS, exploramos dois designs para simulação das funções filter(), map() e reduce(), bem como da composição destas funções.

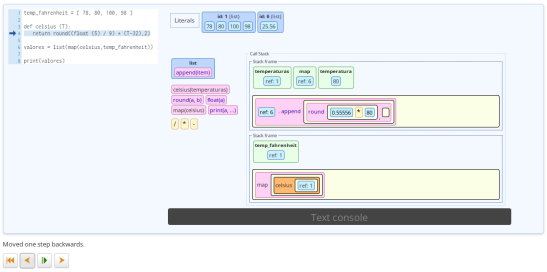
O primeiro design, denominado *Array intermediário*, foi inspirado nos preceitos da Teoria da Carga Cognitiva (TCC). O objetivo é apresentar de forma concreta os passos para construção do array resultante da execução da FOS embutido na representação do JSVEE. Desta forma, o array serviria como um artefato de cognição externa que ilustraria a semântica do funcionamento da FOS, liberando recursos cognitivos anteriormente utilizados para construção de uma representação mental dessa semântica, viabilizando a

[1] https://form.jotform.com/240555359200653

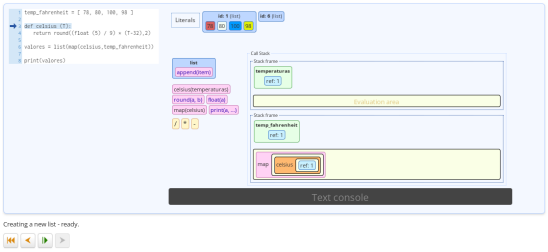
construção de um *schema* destes construtos [Duran et al. 2018]. O segundo design, denominado *Pistas Visuais* apresenta uma *Notional Machine* mais simplificada ao suprimir os passos utilizados na construção do array intermediário, i.e., a avaliação elemento por elemento do array original. De outra forma, este design se vale de elementos visuais (cores) para apresentar a transformação dos elementos no array original e resultante em um único passo. A hipótese deste design remete a algumas observações apresentadas por [Krishnamurthi and Fisler 2021] e [Rivera et al. 2022] onde estudantes eram capazes de compreender o funcionamento de FOS e conjecturaram que poderia haver uma melhor compreensão ao trabalhar em um nível simbólico mais abstrato. Ambos os designs são apresentados na Figura 1.



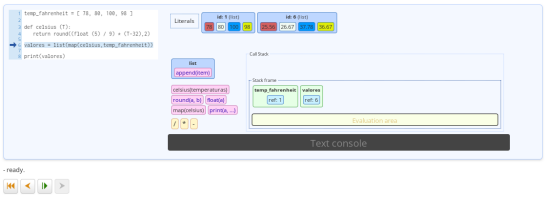
(a) Array Intermediário - Estado Inicial



(b) Array Intermediário - Avaliando Elementos



(c) Animação Colors - Estado Inicial



(d) Animação Colors - Avaliando Elementos

Figura 1. Animações das Avaliações da FOS Map no simulador.

Como parte do processo de investigação, foi desenvolvido um material didático projetando o ensino de FOS em Python para estudantes de programação. O material explora de forma detalhada as funções filter(), map() e reduce(), além de apresentar a técnica de composição de funções. Nosso material utiliza exemplos práticos e animações para ilustrar conceitos complexos de forma clara e intuitiva.

Disponível como formulário web assíncrono sem limite de tempo para conclusão, o material inclui quatorze exemplos práticos que cobrem uma ampla variedade de aplicações das FOS. Todas as versões do material possuem quizzes para engajar e motivar os estudantes, além de vídeos explicativos. Todos os termos utilizados no pós-teste foram apresentados no material de ensino.

Confeccionaram-se três versões do material de ensino: a primeira sem utilizar visualizações embutidas no material, fazendo uso de imagens e vídeos para apresentar os conceitos relacionados às FOS. As outras duas versões, além dos vídeos, utilizam a adaptação das visualizações dos designs de “array intermediário” e “pistas visuais”.

Visando verificar a efetividade do material didático e dos designs de visualização, e identificar possíveis *equívocos* relacionados às FOS, aplicou-se um pós-teste composto

por oito questões com diferentes graus de dificuldade. Algumas questões do pós-teste foram estruturadas para avaliar a capacidade dos participantes de interpretar representações visuais (formas geométricas coloridas) para avaliar a compreensão intuitiva das FOS em um contexto mais abstrato. Os estudantes foram instados a, por exemplo, determinar a coleção resultante de círculos coloridos após um filter que seleciona apenas as formas com cor azul. As demais questões restantes apresentaram trechos de código para avaliar a habilidade de raciocinar sobre a execução de programas e prever os resultados da aplicação de FOS. Apesar de o instrumento não possuir limite de tempo para ser respondido, foi coletado o tempo de resposta dos estudantes. O pós-teste está embutido no mesmo formulário do material de apoio[2] sem necessidade de redirecionamento ou identificação do estudante.

4. Metodologia

Para investigar como diferentes abordagens visuais apoiam a compreensão de Funções de Ordem Superior (FOS) em Python, o estudo operacionalizou experimentalmente o plano metodológico originalmente delineado em [Martins and Duran 2023], adotando uma abordagem mista (quantitativa e qualitativa). Dado o caráter inovador da inserção de simulações de FOS no simulador JSVEE, o formato de estudo piloto permitiu avaliar a viabilidade dos designs visuais e mapear qualitativamente as dificuldades dos estudantes em um cenário real de aplicação.

A concepção do instrumento de avaliação baseou-se na identificação de *equívocos* (*misconceptions*) documentados na literatura de Educação em Computação, especificamente nos trabalhos de [Krishnamurthi and Fisler 2021] e [Rivera et al. 2022]. Para garantir a validade de conteúdo, o instrumento passou por um processo de refinamento por meio de entrevistas e rodadas de feedback com especialistas na área de simulação e visualização de programas [Sirkiä and Sorva 2015].

4.1. Participantes e Aspectos Éticos

O projeto de pesquisa e seus instrumentos de coleta (TCLE, questionários e formulários) foram submetidos ao Comitê de Ética e Pesquisa da Universidade Estadual de Feira de Santana (CEP/UEFS), sob o parecer nº 6.961.536 e CAAE nº 78712124.0.0000.0053, atendendo às Resoluções CNS nº 466/2012 e nº 510/2016.

A pesquisa contou com a participação voluntária de estudantes de graduação de cursos da área de computação (Ciência da Computação, Engenharia da Computação e Engenharia de Software). O recrutamento ocorreu de forma assíncrona entre junho e agosto de 2024, por meio de convites em listas de discussão acadêmicas, redes de pesquisadores em Educação em Computação e turmas de graduação de instituições públicas de ensino superior do Nordeste e Centro-Oeste do Brasil. Como incentivo institucional, foi realizado o sorteio de livros sobre a história da computação entre os participantes que optaram por informar o e-mail, mantendo-se o anonimato das respostas atreladas ao experimento.

Como critério de inclusão, os estudantes deveriam possuir conhecimento equivalente à conclusão de uma disciplina introdutória de programação (CS1) utilizando a linguagem Python. Esse nivelamento foi controlado pela aplicação do pré-teste validado

[2] https://form.jotform.com/240555359200653

por [Duran et al. 2019]. Foram adotados como critérios de exclusão: (i) indicação de proficiência nula nos constructos básicos de Python; (ii) preenchimento incompleto do instrumento de pós-teste; e (iii) conhecimento prévio avançado em FOS, visando isolar o efeito das visualizações em aprendizes do conceito. Desse modo, a mensuração obtida por meio do pré-teste assegurou a homogeneidade da amostra, garantindo que os participantes do estudo detivessem proficiência compatível com os fundamentos da programação imperativa e estruturada em linguagem python.

4.2. Delineamento do Estudo Piloto e Limitações da Amostra

Os participantes foram alocados aleatoriamente em três grupos distintos para interagir com o material didático: o Grupo Controle ($n = 8$), composto por estudantes que realizaram as atividades sem o suporte de animações; o Grupo Array Intermediário ($n = 11$), exposto a animações que explicitavam os estados intermediários; e o Grupo Pistas Visuais ($n = 2$), cujos integrantes interagiram com animações focadas em sinalizações gráficas e cores. Devido ao tamanho diminuto resultante do processo de alocação automatizada, este último grupo foi desconsiderado das análises inferenciais e tratado estritamente sob uma abordagem qualitativa de estudo de caso na Seção 5.

Embora a amostra inicial contasse com 25 respondentes, a aplicação rigorosa dos critérios de exclusão resultou na eliminação de 4 participantes (três por preenchimento incompleto e um por desempenho insuficiente no pré-teste). Como o experimento ocorreu de forma assíncrona e voluntária, a distribuição dos participantes foi totalmente automatizada, o que gerou um desbalanceamento severo na amostra final ($n = 21$) e concentrou as exclusões involuntariamente no grupo de Pistas Visuais. Essa assimetria impede a generalização dos resultados por meio de testes estatísticos de hipóteses inferenciais robustos (como a ANOVA), configurando-se como uma limitação assumida do estudo piloto. Diante disso, a análise de dados foi direcionada para a estatística descritiva e para a taxonomia qualitativa dos erros, uma abordagem considerada altamente valiosa para o desenho de novas intervenções pedagógicas.

4.3. Análise de Dados

Os dados quantitativos e o mapeamento dos erros foram tabulados e analisados com o auxílio da linguagem R (versão 4.4.1). Foram extraídas as frequências de acertos e erros por questão do pós-teste para cada grupo. A análise qualitativa concentrou-se na categorização dos modelos mentais incorretos manifestados pelos estudantes ao responderem às questões conceituais sobre *filter(), map() e reduce()*, confrontando a presença ou ausência dos artefatos visuais com a natureza dos equívocos identificados.

5. Resultados e Análise dos Equívocos

A pontuação média geral obtida no experimento foi relativamente baixa ($M = 4{,}92$; $Mdn = 4{,}06$; $DP = 2{,}44$; $min = 1{,}25$; $máx = 9{,}60$), com os *scores* normalizados para uma escala de 0 a 10 pontos de modo a viabilizar a análise comparativa direta. Ao analisar o desempenho por agrupamento, constatou-se que o grupo controle apresentou métricas levemente superiores às do grupo exposto aos *arrays* intermediários; contudo, ambos os tratamentos exibiram pontuações inferiores àquelas registradas pelo grupo submetido às pistas visuais cromáticas (Tabela 1).

Diante do desbalanceamento amostral severo e do reduzido tamanho do grupo Pistas Visuais ($n = 2$), a aplicação de testes de hipóteses inferenciais (como a ANOVA) mostrou-se inadequada. Por conseguinte, direciona-se o foco científico para a análise estatística descritiva dos grupos viáveis e para o mapeamento qualitativo detalhado dos equívocos conceituais identificados (Figura 2).

Tabela 1. Estatísticas descritivas do desempenho por grupo de análise.

Grupo	Média	Mediana	Desvio Padrão	Amplitude (Range)
Controle	5,39	3,95	2,84	[2,29; 9,60]
Array Intermediário	4,10	4,06	2,02	[1,25; 8,12]
Pistas Visuais*	6,97	6,97	0,43	[6,66; 7,27]

*Nota: Os dados do grupo Pistas Visuais ($n = 2$) possuem caráter estritamente ilustrativo/qualitativo e não viabilizam inferências comparativas de desempenho.

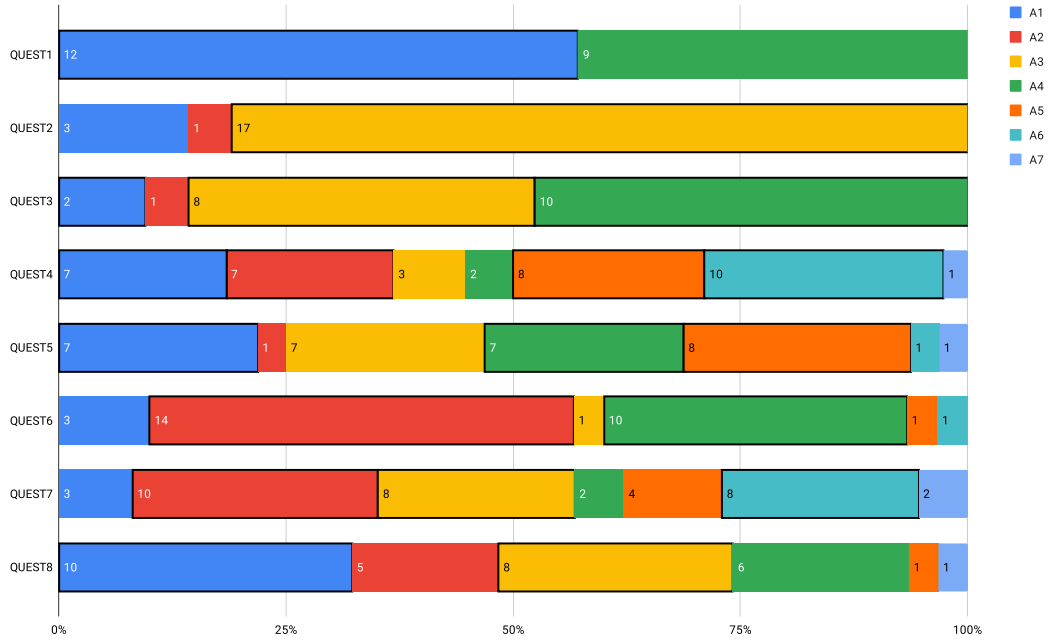


Figura 2. Distribuição e percentual de desempenho por alternativa em cada questão do pós-teste (destaques em preto denotam as alternativas corretas).

5.1. O Equívoco Crônico da Mutabilidade (Questões 2, 3 e 6)

Um dos achados pedagógicos mais expressivos reside na persistência da crença de que as Funções de Ordem Superior alteram de forma destrutiva a estrutura de dados de entrada. Na Questão 3, que avaliava o comportamento da função *map()* na formatação de uma lista de *strings*, 47% dos participantes ($n = 10$) demonstraram o equívoco de que a coleção original seria modificada diretamente na memória (mutabilidade), assinalando a alternativa incorreta (A4). Apenas 38% ($n = 8$) reconheceram o princípio da imutabilidade e o retorno de uma nova coleção isolada (A3).

Esse modelo mental incorreto ecoou em outras tarefas. Na Questão 2 (focada em *filter()*), 4% dos alunos ($n = 2$) indicaram que a função não apenas filtraria os elementos desejados (círculos vermelhos), mas percorreria e alteraria permanentemente a propriedade de cor dos demais itens para vermelho (A2). Analogamente, na Questão 6 (também referente ao *map()*), embora 33% tenham reconhecido a imutabilidade (A4), 10% previram transformações parciais na coleção original (A1) e 3% postularam o esvaziamento completo do *array* de origem (A3). Esses indicadores sugerem uma tendência recorrente de transferência intuitiva da semântica de laços imperativos tradicionais (*for/while* com modificações *in-place*) para o paradigma declarativo das FOS.

5.2. Semântica de Filtros Gráficos e Parâmetros Vazios (Questão 1)

A tradução de critérios lógicos representados graficamente revelou barreiras cognitivas relevantes na Questão 1. Ao aplicar a função *filter()* para extrair círculos de uma lista de formas mistas, 42% dos estudantes ($n = 9$) interpretaram erroneamente que a ausência de preenchimento de cor no elemento gráfico utilizado como parâmetro resultaria em uma lista vazia (A4). Esse resultado evidencia uma fragilidade na construção da *Notional Machine* conceitual: em vez de interpretarem o parâmetro como um filtro de tipo geométrico abstrato, os alunos atribuíram valor semântico literal à ausência de cor na representação visual, obscurecendo a lógica de casamento de padrões da função.

5.3. Modelos Mentais na Inicialização do Acumulador no Reduce (Questão 5)

A compreensão dos estados internos de fluxo e redução de dados na função *reduce()* (Questão 5) consolidou-se como o constructo de maior complexidade para os participantes. Apenas 25% conseguiram rastrear com precisão o estado das variáveis acumuladoras (*acc*) e de iteração em seu passo final, assinalando a alternativa correta (A5).

No que tange estritamente à mecânica de inicialização do acumulador, observou-se que 42% dos estudantes compreenderam a necessidade de inicialização, mas dividiram-se simetricamente quanto ao valor padrão (0 vs primeiro elemento); 25% deduziram uma herança implícita (A3); e 3% ($n = 1$) assumiram erroneamente o retorno de uma lista (A6).

5.4. Composição Sequencial de Funções e o Problema Rainfall (Questões 4, 7 e 8)

A avaliação em cenários avançados buscou verificar a capacidade dos estudantes de inferir fluxos de dados em camadas encadeadas de abstração. Para tanto, a Questão 4 investigou o encadeamento sequencial de múltiplos filtros aplicados a uma mesma coleção, enquanto as Questões 7 (composição híbrida de *filter* e *map*) e 8 (baseada no clássico problema educacional *Rainfall*) exploraram a integração entre diferentes FOS. Na Questão 7, embora 69% tenham obtido acertos parciais, 10% dos estudantes inverteram a ordem de precedência operacional, teorizando que a execução fluiria da função externa para a interna.

Essa mesma dificuldade de encadeamento manifestou-se na Questão 8 [Soloway 1986]. Nela, 19% dos participantes ($n = 6$) erraram o cálculo da média (A4) por dividirem o somatório obtido pelo tamanho da lista de entrada original. Esses alunos falharam em perceber que a operação prévia de filtragem de valores não negativos havia gerado um subconjunto reduzido, alterando diretamente o denominador da operação estatística subsequente.

6. Discussão e Conclusões

A análise das características de design das animações concebidas para o JSVEE (**QP1**) permitiu compreender como diferentes arquiteturas visuais atuam como artefatos pedagógicos no ambiente de aprendizagem de programação.

A abordagem de *Array Intermediário* funciona como um andaime cognitivo explícito (*scaffolding*). Ao externalizar graficamente cada etapa das transformações sequenciais e projetar subestruturas temporárias na tela, a simulação reduz a necessidade de simulação mental por parte do aluno. Contudo, os dados descritivos sugerem que esse excesso de quadros e detalhamento passo a passo pode ter induzido a um efeito de sobrecarga de atenção dividida em estudantes com menor conhecimento prévio, o que justifica o desempenho discretamente inferior desse grupo ($M = 4,10$).

Por sua vez, o grupo exposto à estratégia de Pistas Visuais apresentou uma média descritiva de $M = 6{,}97$. No entanto, devido à amostragem extremamente reduzida ($n = 2$), este dado não deve ser interpretado como evidência da superioridade da intervenção, mas como uma observação preliminar de estudo de caso. Este comportamento atípico sugere um provável viés de seleção decorrente do conhecimento prévio dos participantes, e não necessariamente da eficiência isolada do design visual. Não obstante essa limitação, o achado levanta a hipótese de que a abstração cromática pode otimizar o processamento da carga cognitiva intrínseca ao focar a atenção do aprendiz nos pontos críticos de mutação dos dados.

No escopo da eficácia dessas abordagens no desempenho geral (**QP2**), devido às limitações amostrais inerentes ao caráter piloto e exploratório deste estudo a análise concentrou-se no mapeamento descritivo e qualitativo dos equívocos conceituais. Ao categorizar onde e por que os alunos falham ao processar abstrações como *map() e reduce()*, o relato provê subsídios empíricos essenciais para docentes que enfrentam o desafio de lecionar o paradigma “ Data-Centric ” no ambiente escolar e acadêmico.

6.1. Diretrizes Práticas e Trabalhos Futuros

Com base nos achados, sugerem-se duas diretrizes de design instrucional: (i) Acentuação Visual da Imutabilidade, explicitando graficamente que funções como *filter() e map()* alocam um novo objeto em memória para combater a concepção de mutabilidade imperativa (detectada em 47% da amostra); e (ii) Isolamento da Inicialização no *reduce()*, criando um quadro estático exclusivo para a variável acumuladora para desmistificar heranças implícitas de parâmetros (erro de 25% dos alunos).

Para trabalhos futuros, planeja-se além de expandir a escala desse experimento, a integração do simulador JSVEE a protocolos de rastreamento ocular (*eye-tracking*) e a utilização de algoritmos de alocação em blocos(block randomization) para garantir grupos de tamanhos equivalentes visando mitigar o desbalanceamento automatizado aqui observado.

7. Declaração sobre uso de Inteligência Artificial

Os autores declaram que a IA (Gemini) foi utilizada exclusivamente como suporte à produtividade, auxiliando na correção textual, no uso de sinônimos e na estruturação lógica.

Referências

Beckmann, J. F. (2010). Taming a beast of burden–on some issues with the conceptualisation and operationalisation of cognitive load. *Learning and instruction*, 20(3):250–264.

Boulay, B. D. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73.

Brodley, C. E., Hescott, B. J., Biron, J., Ressing, A., Peiken, M., Maravetz, S., and Mislove, A. (2022). Broadening participation in computing via ubiquitous combined majors (cs+ x). In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, pages 544–550.

de Koning, B. B., Tabbers, H. K., Rikers, R. M., and Paas, F. (2010). Attention guidance in learning from a complex animation: Seeing is understanding? *Learning and Instruction*, 20(2):111–122. Eye tracking as a tool to study and enhance multimedia learning.

Duran, R., Rybicki, J.-M., Sorva, J., and Hellas, A. (2019). Exploring the value of student self-evaluation in introductory programming. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, page 121–130, New York, NY, USA. Association for Computing Machinery.

Duran, R., Sorva, J., and Leite, S. (2018). Towards an analysis of program complexity from a cognitive perspective. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*, ICER '18, page 21–30, New York, NY, USA. Association for Computing Machinery.

Duran, R., Sorva, J., and Seppälä, O. (2021). Rules of program behavior. *ACM Trans. Comput. Educ.*, 21(4).

Duran, R., Zavgorodniaia, A., and Sorva, J. (2022). Cognitive load theory in computing education research: A review. *ACM Trans. Comput. Educ.*, 22(4).

Eliot, Sutinen, E., Tarhio, J., Lahtinen, S.-p., Tuovinen, A.-P., Rautama, E., and Meisalo, V. (1997). Eliot - an algorithm animation environment.

Guo, P. J. (2013). Online python tutor: Embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, page 579–584, New York, NY, USA. Association for Computing Machinery.

Guzdial, M. and Shreiner, T. (2021). Integrating computing through task-specific programming for disciplinary relevance: Considerations and examples. In *Computational Thinking in Education*, pages 172–190. Routledge.

Kong, H.-K., Liu, Z., and Karahalios, K. (2017). Internal and external visual cue preferences for visualizations in presentations. *Comput. Graph. Forum*, 36(3):515–525.

Krishnamurthi, S. and Fisler, K. (2020). Data-centricity: a challenge and opportunity for computing education. *Communications of the ACM*, 63(8):24–26.

Krishnamurthi, S. and Fisler, K. (2021). Developing behavioral concepts of higher-order functions. In *Proceedings of the 17th ACM Conference on International Computing Education Research*, ICER 2021, page 306–318, New York, NY, USA. Association for Computing Machinery.

Levy, R., Ben-Ari, M., and Uronen, P. (2003). The jeliot 2000 program animation system. *Computers & Education*, 40:1–15.

Martins, M. R. and Duran, R. (2023). Ferramentas de visualização de programas na compreensão de funções de alta-ordem. In *Anais Estendidos do III Simpósio Brasileiro de Educação em Computação*, pages 18–19. SBC.

Mayer, R. E. (1999). Multimedia aids to problem-solving transfer. *International Journal of Educational Research*, 31(7):611–623.

Muldner, K., Jennings, J., and Chiarelli, V. (2022). A review of worked examples in programming activities. *ACM Trans. Comput. Educ.*, 23(1).

Mutlu-Bayraktar, D., Cosgun, V., and Altan, T. (2019). Cognitive load in multimedia learning environments: A systematic review. *Computers & Education*, 141:103618.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., and Velázquez-Iturbide, J. A. (2002). Exploring the role of visualization and engagement in computer science education. *SIGCSE Bull.*, 35(2):131–152.

Renkl, A. and Scheiter, K. (2017). Studying visual displays: How to instructionally support learning. *Educational Psychology Review*, 29.

Rivera, E. and Krishnamurthi, S. (2022). Structural versus pipeline composition of higher-order functions (experience report). *Proc. ACM Program. Lang.*, 6(ICFP).

Rivera, E., Krishnamurthi, S., and Goldstone, R. (2022). Plan composition using higher-order functions. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1*, ICER '22, page 84–104, New York, NY, USA. Association for Computing Machinery.

Scaffidi, C., Shaw, M., and Myers, B. (2005). An approach for categorizing end user programmers to guide software engineering research. In *Proceedings of the First Workshop on End-User Software Engineering*, WEUSE I, page 1–5, New York, NY, USA. Association for Computing Machinery.

Schroeder, N. and Cenkci, A. (2018). Spatial contiguity and spatial split-attention effects in multimedia learning environments: a meta-analysis. *Educational Psychology Review*, 30:1–23.

Sirkiä, T. (2013). A javascript library for visualizing program execution. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*, Koli Calling '13, page 189–190, New York, NY, USA. Association for Computing Machinery.

Sirkiä, T. et al. (2017). Creating, tailoring, and distributing program animations-supporting the production process of interactive learning content.

Sirkiä, T. and Sorva, J. (2015). Tailoring animations of example programs. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, pages 147–151.

Sirkiä, T. (2016). Jsvee & kelmu: Creating and tailoring program animations for computing education. In *2016 IEEE Working Conference on Software Visualization (VISSOFT)*, pages 36–45.

Soloway, E. (1986). Learning to program= learning to construct mechanisms and explanations. *Communications of the ACM*, 29(9):850–858.

Sorva, J. et al. (2012). *Visual program simulation in introductory programming education*. Aalto University.

Sorva, J. and Sirkiä, T. (2010). Uuhistle: a software tool for visual program simulation. In *Koli Calling*.

Wang, X., Lin, L., Han, M., and Spector, J. M. (2020). Impacts of cues on learning: Using eye-tracking technologies to examine the functions and designs of added cues in short instructional videos. *Computers in Human Behavior*, 107:106279.