Concepção de um catálogo de métodos HCD para o ensino de programação introdutória

Katyeudo Karlos de Sousa Oliveira - Universidade de São Paulo; Instituto Federal do Ceará - karlos_oliveira@usp.br
Orcid: 0000-0002-4414-1599
Anderson da Silva Marcolino - Universidade Federal do Paraná - anderson.marcolino@ufpr.br
Orcid: 0000-0002-4014-1882
William Simão de Deus - Instituto Federal do Paraná - william.deus@ifpr.edu.br
Orcid: 0000-0001-7540-3742
Gustavo Martins Nunes Avellar - Universidade de São Paulo - gustavo.avellar@usp.br
Orcid: 0000-0002-3340-6766
Taciana Pontual Falcão - Universidade Federal Rural de Pernambuco - taciana.pontual@ufrpe.br
Orcid: 0000-0003-2775-4913
Ellen Francine Barbosa - Universidade de São Paulo - francine@icmc.usp.br
Orcid: 0000-0003-3275-2293

Resumo: Estudantes do ensino superior frequentemente enfrentam desafios significativos em disciplinas de programação introdutória, caracterizadas por elevadas taxas de abandono e insucesso. Esses problemas decorrem, em parte, da complexidade inerente aos conteúdos, da ausência de habilidades de resolução de problemas e de lacunas na formação docente. Neste contexto, processos de *design* centrados no ser humano (*Human-Centered Design* - HCD) têm emergido como abordagens promissoras para o ensino de programação. Este estudo explorou o potencial do HCD para aprimorar a aprendizagem nessas disciplinas, identificando 27 métodos HCD por meio de levantamento documental. Esses métodos foram organizados em um catálogo, validado com professores em uma etapa inicial de avaliação. Os resultados indicaram que os métodos são aplicáveis e relevantes, promovendo o desenvolvimento de habilidades colaborativas e analíticas. Com base no *feedback* recebido, recomendou-se a simplificação e a adaptação dos métodos, culminando em um catálogo aprimorado, com descrições acessíveis e sugestões adaptáveis às diversas realidades educacionais.
Palavras-chave: Ensino de Programação; *Human-Centered Design*; Programação Introdutória

Computational Thinking in Introductory Programming and 21st Century Skills: A Systematic Literature Mapping

Abstract: Higher education students often face significant challenges in introductory programming courses, which are characterised by high dropout and failure rates. These issues stem partly from the inherent complexity of the content, the lack of problem-solving skills, and gaps in teacher training. In this context, Human-Centered Design (HCD) processes have emerged as promising approaches to programming education. This study explored the potential of HCD to enhance learning in such courses by identifying 27 HCD methods through a document review. These methods were organised into a catalogue and underwent an initial validation phase with teachers. The findings indicated that the methods are applicable and relevant, fostering the development of collaborative and analytical skills. Based on the feedback received, simplification and adaptation of the methods were recommended, resulting in an improved catalogue with accessible descriptions and flexible suggestions tailored to diverse educational contexts.
Keywords: Programming Teaching; Human-Centered Design; Introductory Programming

1 Introdução

No ensino superior, no decorrer dos cursos em que as habilidades de programação de computadores são essenciais (por exemplo, Ciência da Computação, Sistemas de Informação, Engenharia de Computação, etc.), os estudantes podem ter dificuldades para aprender e

desenvolver programas de computador (Pedrosa *et al*., 2017; *Du et al*., 2016; Oliveira *et al*., 2023a). Além disso, as disciplinas de programação introdutória normalmente apresentam uma alta taxa de abandono e insucesso em função de vários fatores, como a complexidade, a natureza técnica e a possível ausência de habilidades de resolução de problemas (Medeiros *et al*., 2021).

No contexto das disciplinas de programação introdutória, o termo "alta taxa de abandono" refere-se à proporção significativa de estudantes que desistem ou não obtêm sucesso ao final dessas disciplinas (Alturki, 2016; Rõõm *et al*., 2021). Estudos como os de Bennedsen e Caspersen (2019) apontam que as taxas de reprovação podem variar entre 30% e 50% em diferentes instituições ao redor do mundo, caracterizando um problema recorrente na área. Essas taxas podem ser atribuídas a diversos fatores, como desmotivação, dificuldades em acompanhar o ritmo das aulas, falta de suporte adequado, e até questões extracurriculares, como sobrecarga de trabalho ou problemas pessoais (Cheah, 2020; Villamor, 2020; Rõõm *et al*., 2021). Embora a complexidade e a ausência de habilidades de resolução de problemas sejam elementos relevantes, de acordo com Duran *et al.* (2023), é importante reconhecer que o abandono também pode estar relacionado a questões pedagógicas e contextuais que extrapolam a natureza técnica do conteúdo abordado.

Grande parte das pesquisas se concentra nas dificuldades que os estudantes encontram ao aprender programação (Robins, 2010; Watson, Li 2014; Medeiros *et al*., 2021). No entanto, é importante considerar que alguns desses problemas podem estar relacionados ao processo de ensino, aos professores e à sua formação (Changpetch *et al*., 2022). Portanto, é necessário investigar a relação entre as dificuldades enfrentadas pelos professores e sua formação para o ensino de programação (Berssanette; Francisco 2021).

Recentemente, processos de *design* (por exemplo, *design thinking*, *human-centered design*, etc.) tornaram-se presentes em cursos de programação (Lai *et al*., 2022). Um processo de *design* é um processo de pensamento que engloba a concepção e materialização de uma ideia em forma tangível (IDEO.org 2015). Os processos de *design* podem preencher uma lacuna entre a pedagogia teórica e a experiência prática no processo de ensino e aprendizagem (Scheer *et al*., 2012).

A ideia do HCD especifica que os usuários finais estejam no centro dos projetos de sistemas técnicos (Revano; Garcia 2020). O processo de programação de computadores é basicamente um processo de *design* de resolução de problemas (Simonsen; Robertson 2013). De acordo com Park e McKilligan (2018), o processo HCD compartilha elementos semelhantes ao Pensamento Computacional. Por exemplo, a decomposição de um problema no Pensamento Computacional é semelhante à definição de um problema no processo de pensamento de *design*. Ambos os processos definem questões a partir da análise e abstração (Park; McKilligan, 2018). Portanto, algumas relações podem existir entre a disposição do HCD e a programação de computadores.

Fomentar habilidades de programação introdutória, como abstração e raciocínio lógico, é crucial para capacitar estudantes a resolver problemas complexos de maneira eficiente e inovadora (Oliveira *et al*., 2023b; Oliveira *et al*., 2024). Essas habilidades permitem a decomposição de problemas em componentes manejáveis e a criação de soluções algorítmicas precisas (Xie *et al*., 2019). O HCD pode ter a capacidade de complementar esse desenvolvimento, pois coloca o usuário no centro do processo de design, promovendo a compreensão profunda dos problemas e a criação de soluções mais intuitivas e relevantes (IDEO.org 2015).

Apesar da crescente aplicação dos processos de *design*, como o HCD, não foi possível identificar na literatura pesquisada a existência de um catálogo similar que utilize esses métodos especificamente no contexto da programação introdutória. Esta lacuna representa uma oportunidade significativa para avançar nesta área, desenvolvendo e indicando um catálogo de métodos HCD que possa auxiliar no fomento das habilidades de programação introdutória dos estudantes.

Assim sendo, este estudo visa o estabelecimento de um catálogo de métodos HCD para orientar o desenvolvimento de experiências de ensino e aprendizagem para o fomento das habilidades de programação introdutória dos estudantes do ensino superior. Para isso, um levantamento documental foi conduzido, identificando materiais técnicos relevantes do HCD. Após o levantamento, os métodos HCD foram identificados para integrar o ensino de programação e compor um catálogo, totalizando 27 métodos que, por sua vez foram analisados através de uma validação inicial com professores de Computação.

O restante deste trabalho está organizado em: a Seção 2 revisa os estudos correlatos; a Seção 3 detalha a metodologia utilizada; a Seção 4 apresenta os resultados alcançados na entrevista com os professores; a Seção 5 expõe o catálogo, principal objeto deste estudo; a Seção 6 discute as possíveis aplicações do catálogo; e a Seção 7 encerra o artigo.

2. Estudos Correlatos

Virguez *et al*., (2020) avaliaram a percepção dos participantes de um curso introdutório de *design* de engenharia quanto ao valor e utilidade do curso. Os alunos utilizaram software de modelagem, programação introdutória, plataforma de *hardware* baseada em Arduino e impressão 3D e responderam a uma pesquisa com uma abordagem de *design* centrado no ser humano. Os resultados indicaram um aumento na capacidade dos participantes de aplicar conceitos de *design* de engenharia. Embora não tenha sido abordado o aprendizado de programação introdutória, sugeriu-se que a abordagem pode ser aplicada no desenvolvimento de currículos de programação mais motivadores e engajantes, alinhados às necessidades dos alunos.

Tsai e Wang (2021) exploraram a relação entre a disposição de pensamento de *design* e a autoeficácia na programação de computadores. O estudo, conduzido com alunos do ensino fundamental na Turquia, revelou uma correlação positiva significativa entre a disposição de *design* e a autoeficácia na programação. Os resultados sugerem que o pensamento de *design* pode melhorar a autoeficácia na programação e, consequentemente, o desempenho dos alunos. Embora não se concentrem apenas na programação, o estudo traz indicam implicações importantes para o ensino dessa habilidade.

Grace *et al*., (2022) apresentaram uma abordagem de ensino que combina aprendizagem online e presencial para ensinar *design* de interação e programação. A abordagem, baseada no HCD, foi implementada em uma universidade na Dinamarca, resultando em alunos mais capacitados na aplicação de habilidades de programação em projetos de *design* de interação. Já Blanco *et al*. (2017) descreveram um estudo que utilizou uma abordagem de *design* HCD para melhorar as habilidades de empatia e trabalho em equipe de estudantes de informática. A abordagem envolveu a criação de equipes multidisciplinares que trabalharam juntas para desenvolver um aplicativo de software, resultando em melhorias significativas nessas habilidades.

Embora esses estudos ofereçam contribuições importantes para o ensino de programação, é necessário um esforço para desenvolver e testar abordagens que abranjam todas as fases do HCD. Especificamente, há uma lacuna no entendimento de como os métodos do HCD podem ser aplicados para promover as habilidades de programação introdutória, uma vez que nenhum dos estudos analisados abordou essa questão até agora. Portanto, há uma oportunidade para explorar essa interseção entre o HCD, o ensino de programação e as habilidades de programação, visando aprimorar ainda mais os métodos educacionais e atender às demandas do século atual (Oliveira; Souza, 2022).

3. Metodologia

Para a elaboração do catálogo de métodos HCD, inicialmente, foi realizado um levantamento documental com foco em documentos técnicos e literaturas relevantes na área de HCD destinado à aplicação nas disciplinas de programação introdutória. Com esse catálogo, nosso objetivo é fornecer um conjunto de ferramentas e técnicas que promovam uma abordagem centrada no aluno, contribuindo para o aprimoramento do ensino de programação nos cursos superiores de Computação.

A escolha pelo levantamento documental em detrimento de, por exemplo, um mapeamento ou revisão sistemática se justifica pela natureza do objetivo da pesquisa. Como o foco está na seleção de métodos em documentos técnicos do HCD, o levantamento documental permite uma abordagem mais direcionada e qualitativa (Silva *et al*., 2009). Dessa forma, foi possível realizar uma análise dos métodos disponíveis em tais documentos, identificando aqueles mais adequados às necessidades e objetivos específicos da pesquisa (Silva *et al*., 2009). Além disso, o levantamento documental oferece flexibilidade para explorar uma ampla gama de fontes e autores relevantes (Silva *et al*., 2009), possibilitando uma abordagem mais abrangente e contextualizada na seleção dos métodos HCD.

A pesquisa concentrou-se em fontes como Vianna *et al*., (2012), IDEO.org (2015), Stanford (2016, 2018), Schools2030 (2021) e NESTA (2019) devido à importância e abrangência na aplicação do HCD em diversas áreas, desde a pesquisa até a implementação de soluções.

O trabalho de Vianna *et al*., (2012) foi selecionado devido à sua contribuição para o pensamento de *Design* e sua aplicação prática em inovação nos negócios, ressaltando a importância do pensamento centrado no humano e da colaboração em equipe. O IDEO.org (2015) oferece um *kit* de ferramentas que facilita a aplicação do pensamento de design HCD em projetos e soluções de problemas, sendo útil para profissionais e educadores. O *D.School Design Project Guide* da *Stanford University* (Stanford, 2016) fornece uma estrutura e atividades práticas para a realização de projetos de HCD, adaptáveis para uso em projetos educacionais. O *Design Thinking Bootleg* da *Stanford University* (Stanford, 2018) e o *Schools 2030 Human-Centered Design Toolkit* (Schools2030, 2021) oferecem ferramentas e recursos específicos para a aplicação do HCD na educação, desde a pesquisa até a implementação de soluções pedagógicas. Por fim, *o Collective Intelligence Design Playbook* (NESTA, 2019) apresenta métodos de *design thinking* para projetos colaborativos, úteis para o desenvolvimento de soluções inovadoras envolvendo diferentes partes interessadas, incluindo educadores e alunos.

Na sequência, os métodos HCD foram elencados, cada um acompanhado de suas respectivas fontes. Esse processo de catalogação foi essencial para garantir a transparência e a rastreabilidade das informações, permitindo que os métodos fossem facilmente identificados e associados às fontes originais. Os métodos encontrados e inseridos no catálogo e suas respectivas fontes documentais são apresentados no Quadro 1.

Quadro 1. Métodos HCD e Fontes Documentais.

Método	**Documentos Técnicos**					
	(Vianna *et al*., 2012)	**(IDEO.org, 2015)**	**(Stanford, 2016)**	**(Stanford, 2018)**	**(Schools2030, 2021)**	**(NESTA, 2019)**
Brainstorming	X	X	X	X	X	
Cardápio de Ideias	X				X	
Como Nós Podemos?		X	X	X	X	
Desafio de Design	X	X		X	X	
Desenho		X				
Diagrama de Afinidades	X					
Imersão no Contexto		X		X		

Inspiração Análoga	X			X		
Jornada do Usuário	X	X	X	X	X	
Mapa da Empatia	X		X			
Mapa de Problemas						X
Mapa Mental		X				
Matriz 2x2		X	X	X		
MVP - Mínimo Produto Viável						X
Obter Feedback		X	X	X		
Personas	X					X
Pesquisa *Desk*	X				X	
Pesquisa Exploratória	X	X		X	X	
Piloto		X			X	X
Pitch	X					
Protótipo		X	X	X	X	X
Prova de Conceito		X		X		X
Resumo de Soluções					X	X
Roadmap		X				
Sessão de Cocriação	X	X				
Storyboard	X	X			X	X
Storytelling		X	X	X	X	X

A partir da análise dessas fontes, foi extraída a descrição de cada um dos métodos HCD, que posteriormente foi organizada e sistematizada no catálogo. Esse catálogo visa fornecer aos professores uma referência abrangente e acessível, contendo informações essenciais sobre cada método e diretrizes práticas para sua aplicação no contexto educacional.

3.1. Validação do Catálogo

Para validar o catálogo desenvolvido, foi realizado uma validação inicial com professores com o objetivo de avaliar a eficácia e a aplicabilidade dos métodos HCD no ensino de programação introdutória. As questões de pesquisa (**QP**) investigadas foram: **QP01:** Os métodos HCD possuem aplicabilidade para contexto do processo de ensino e aprendizagem de programação introdutória?; **QP02**: Os métodos HCD fomentam as habilidades de programação?

3.2. Amostragem da Pesquisa

Seis professores de disciplinas de programação foram convidados para participar do estudo. Os professores possuem (em média) sete anos de experiência na condução de pesquisas e no ensino de programação introdutória. A participação foi voluntária.

Três sessões *online* via *Google Meet* foram realizadas para apresentar os métodos HCD aos participantes. Os seis professores foram divididos em duplas e cada dupla participou de uma sessão. Durante a sessão, cada método foi explicado detalhadamente, com exemplos práticos de aplicação no ensino de programação introdutória. Para auxiliar na discussão, foi utilizada uma apresentação visual. Para capturar as percepções dos professores, foram feitas perguntas-chave, como: **(i)** *Como vocês veem a aplicação do método [nome do método] no ensino de programação introdutória?*; **(ii)** *De que forma a utilização do método [nome do método] pode contribuir para o desenvolvimento das habilidades de programação?*.

Apesar de os professores possuírem experiência significativa no ensino de programação introdutória — todos com doutorado na área de computação e, em média, sete anos de atuação —, reconhecemos que nem todos estavam previamente familiarizados com os métodos HCD apresentados. Para minimizar essa limitação, as sessões foram planejadas para garantir uma introdução detalhada de cada método, incluindo exemplos específicos de aplicação no ensino de programação. Durante as apresentações, buscou-se conectar os métodos a práticas pedagógicas com as quais os professores já estivessem familiarizados, facilitando a compreensão e a contextualização. No entanto, é importante ressaltar que as percepções dos

participantes podem ter sido influenciadas por sua experiência pessoal e pela interpretação dos exemplos fornecidos. Dessa forma, as avaliações refletem uma análise inicial, baseada na eficácia percebida, e não necessariamente na aplicação prática direta dos métodos em sala de aula. Esse aspecto será abordado em estudos futuros, com o objetivo de ampliar a validação por meio de testes empíricos em ambientes educacionais reais.

Após a sessão, foi disponibilizado um formulário *online* para que os professores classificassem cada método com base em sua eficácia percebida na promoção das habilidades de programação.

3.3. Análise de Dados

Os dados coletados foram analisados de maneira sistemática e objetiva por todos os autores do estudo. As respostas dos professores foram compiladas e examinadas para identificar padrões e percepções sobre a aplicabilidade e eficácia dos métodos HCD no ensino de programação.

3.4. Ameaças à Validade

A validade desta etapa da pesquisa pode ser comprometida por algumas ameaças. Primeiro, a **validação interna** pode ser afetada pela subjetividade na percepção dos professores sobre a aplicabilidade dos métodos HCD. Para mitigar isso, foram utilizados questionários estruturados e sessões de feedback detalhadas para assegurar uma compreensão clara e uniforme dos métodos. Segundo, a **validação externa** pode ser limitada pela pequena amostra de apenas seis professores, o que pode não representar a diversidade de contextos de ensino de programação. Para reduzir essa limitação, selecionamos professores com uma experiência média de pelo menos sete anos na área de programação introdutória. Terceiro, a **validade de construto** pode ser ameaçada pela interpretação e implementação dos métodos HCD. Para minimizar este risco, os métodos foram apresentados com exemplos práticos e detalhados. Por fim, a **validade de conclusão** pode ser afetada por um viés na interpretação dos dados coletados. Para abordar essa ameaça, os dados foram analisados de maneira sistemática e objetiva por todos os autores deste estudo. Além disso, foi adotada uma abordagem de triangulação de pesquisadores, onde cada autor revisou independentemente os dados, discutindo e comparando suas interpretações para minimizar possíveis vieses individuais.

4. Resultados

Após a condução da validação com os professores, as respostas foram analisadas para compreender suas percepções em relação aos métodos HCD propostos. Para melhor distinguir entre as respostas dos professores, cada um foi identificado por meio de um código único, sendo denominados como prof-01, prof-02, etc. Essa abordagem permite uma análise mais precisa das diferentes perspectivas e opiniões apresentadas pelos participantes do estudo.

Os professores reconhecem a variedade e pertinência dos métodos, destacando sua importância não apenas para o desenvolvimento das habilidades dos alunos, mas também para aprimorar o processo de ensino. O prof-01 e o prof-02 ressaltam a necessidade de avaliar a receptividade e o engajamento dos alunos com as atividades, sugerindo a mensuração do progresso das habilidades adquiridas. Já o prof-03 destaca a importância de métodos como *Brainstorming*, Cardápio de Ideias, pesquisa *Desk* e Pesquisa exploratória, fundamentais para analisar e discutir ideias, promovendo habilidades como trabalho em grupo e análise de requisitos.

Os professores concordam que a diversidade de métodos pode enriquecer o processo de ensino, permitindo uma abordagem mais variada e profunda. No entanto, o prof-04 ressalta a necessidade de simplificar a aplicação de alguns métodos, considerando especialmente a complexidade inerente ao ensino de programação introdutória. Isso sugere a importância de adaptar os métodos para torná-los viáveis e acessíveis aos alunos, sem sobrecarregar o conteúdo da disciplina.

Segundo os professores prof-05 e prof-06, métodos como *Brainstorming*, Obter Feedback e Pesquisa *Desk* foram especialmente valorizados por potencialmente promoverem habilidades colaborativas e analíticas, fundamentais no aprendizado da programação introdutória. No entanto, apesar do potencial identificado por esses professores, é necessário realizar mais pesquisas para validar a aplicabilidade e eficácia desses métodos no contexto do ensino de programação introdutória e para entender melhor como eles podem ser otimizados e adaptados às necessidades específicas da disciplina e dos alunos.

Por fim, os professores mencionam desafios relacionados ao tempo disponível para aplicação dos métodos e à complexidade de alguns deles. O prof-04 destaca a necessidade de equilibrar a variedade de métodos com o tempo disponível em sala de aula, enquanto o prof-02 sugere a simplificação de métodos mais complexos e a ampliação do período de aplicação para garantir a compreensão mais profunda por parte dos alunos.

Com base nas indicações dos professores, foram aplicadas correções e melhorias nos exemplos apresentados para cada método HCD, resultando no catálogo de métodos HCD, apresentado a seguir.

5. Catálogo de métodos HCD para o ensino de programação introdutória

O catálogo apresentado no Quadro 2 inclui o nome de cada método HCD, uma descrição detalhada do mesmo e uma sugestão de como utilizá-lo no processo de ensino e aprendizagem de programação introdutória. É importante destacar que estas são apenas sugestões; os educadores têm a flexibilidade de adaptar cada método de acordo com o contexto específico de suas turmas e alunos. Além disso, não é necessário aplicar todos os métodos apresentados. Os educadores podem escolher e utilizar os métodos de forma independente, conforme as necessidades e circunstâncias de sua prática educativa.

Quadro 2. Catálogo para o Ensino de Programação.

Brainstorming: Sessão colaborativa para gerar e explorar ideias sobre um tema específico. Mediada por um moderador, visa estimular a criatividade, promover a troca de conhecimento e desenvolver soluções inovadoras através da contribuição aberta e sem julgamentos de todos os participantes. **Sugestão de Tarefa:** O professor pode organizar sessões de *brainstorming* em equipe para resolver problemas específicos de programação. Durante essas sessões, os alunos podem ser incentivados a contribuir com ideias, percepções e soluções para desafios relacionados à programação.
Cardápio de Ideias: Um catálogo estruturado que compila e organiza as ideias do projeto, apresentando-as de maneira clara e acessível. Este método facilita a visualização e comparação das opções disponíveis, apoiando a tomada de decisões. **Sugestão de Tarefa:** Os professores podem criar um "Cardápio de Ideias", onde os alunos registram de forma sintética todas as ideias geradas para a resolução de problemas específicos de programação. Este catálogo incluirá não apenas as ideias em si, mas também observações relacionadas, possíveis desenvolvimentos e potenciais oportunidades de aplicação prática.
Como Nós Podemos?: Um método que converte *insights* em oportunidades de *design* através da formulação de perguntas desafiadoras que começam com "Como nós podemos...?". Esta abordagem promove o pensamento inovador e orientado à solução, incentivando a equipe a explorar novas possibilidades e desenvolver ideias criativas. **Sugestão de Tarefa:** Após a apresentação de um conceito, os alunos podem ser encorajados a transformar insights em oportunidades de *design*. Cada *insight* será acompanhado por uma pergunta desafiadora iniciada com "Como nós podemos...?". Essas perguntas não devem sugerir soluções específicas, mas sim abrir espaço para a criatividade dos alunos ao explorar diferentes abordagens para resolver o desafio proposto.
Desafio de *Design*: Uma proposição que articula uma questão central voltada para as necessidades humanas, servindo como um ponto de referência crucial para orientar e inspirar o desenvolvimento do projeto. **Sugestão de Tarefa:** Durante as aulas os alunos podem ser incentivados a formular proposições que resumam questões significativas ou necessidades implícitas das pessoas relacionadas à programação.

Desenho: Criação de representações visuais, como desenhos, gráficos ou diagramas, que facilitam a comunicação de ideias e mantêm um registro visual das etapas e descobertas da pesquisa. **Sugestão de Tarefa:** Durante as aulas de programação introdutória, os professores podem encorajar os alunos a criar desenhos, gráficos ou diagramas, utilizando papel e caneta ou algum software específico, para representar visualmente a lógica de algoritmos simples.
Diagrama de Afinidades: Um método de agrupamento que organiza *insights* por similaridade, facilitando a identificação de padrões e a análise de dados. Essencial para estruturar informações de forma lógica e coesa, promovendo uma compreensão mais profunda do conteúdo analisado. **Sugestão de Tarefa:** Os alunos podem ser incentivados a organizar ideias relacionadas à programação com base em afinidades e similaridades. Isso possibilitará a identificação de macro áreas temáticas, subdivisões e interdependências, oferecendo uma visão clara e organizada dos conceitos-chave.
Imersão no Contexto: Observação aprofundada e envolvimento direto no cotidiano das pessoas para compreender suas rotinas, formas de trabalho e interações sociais, visando obter uma compreensão mais rica e detalhada do contexto humano. **Sugestão de Tarefa:** Durante as aulas, os professores podem promover uma imersão simulada ou real no contexto da resolução de problemas de programação. Os alunos podem ser orientados a se dedicarem a observar, de maneira próxima, um cenário fictício ou real relacionado ao desafio de design proposto, permitindo que os alunos compreendam melhor o ambiente em que suas soluções de programação serão aplicadas.
Inspiração Análoga: Exploração de contextos externos ao ambiente de trabalho para enriquecer o desafio de *design*. Ao identificar e isolar elementos de experiências, interações ou produtos em diferentes ambientes, elas podem ser estrategicamente aplicadas para inovar e aprimorar soluções no projeto em questão. **Sugestão de Tarefa:** Os professores podem incentivar os alunos a saírem do ambiente tradicional de aula e buscar inspiração análoga para os desafios de programação. A ideia é que eles observem elementos, interações ou padrões interessantes nesses lugares e identifiquem como esses conceitos podem ser aplicados analogamente aos problemas de programação em discussão.
Jornada do Usuário: Mapeamento detalhado dos principais pontos de interação do usuário com um produto ou serviço, destacando momentos críticos da experiência para identificar oportunidades de melhoria e inovação. **Sugestão de Tarefa:** Durante a aula, os alunos podem ser incentivados a mapear os diferentes estágios que um usuário passa, desde a descoberta do programa até o uso contínuo. Isso envolveria a identificação de pontos cruciais, como a interface de entrada, interações do usuário, possíveis desafios e como a solução aborda as necessidades do usuário em cada etapa.
Mapa da Empatia: Método visual que captura e organiza características, atitudes e comportamentos do público-alvo, proporcionando uma compreensão abrangente e profunda das suas necessidades, desejos e experiências. **Sugestão de Tarefa:** Durante a aula, os alunos podem ser divididos em grupos e orientados a coletar informações sobre as características, atitudes, comportamentos, pensamentos e sentimentos dos usuários em relação ao programa ou aplicação em desenvolvimento. Os alunos podem preencher as seções do mapa de empatia com base em pesquisas, entrevistas ou observações.
Mapa de Problemas: Um método que permite representar de forma concisa problemas interconectados. Serve como uma ferramenta para compreender a complexidade de um sistema, identificar suas causas raiz e potenciais soluções. **Sugestão de Tarefa:** Os professores podem instruir os alunos a criar mapas de problemas relacionados a um desafio específico de programação. um grupo de alunos pode ser designado para identificar problemas interconectados associados ao desenvolvimento de um programa ou aplicativo. Podem utilizar *post-its*, papel ou ferramentas *online* para mapear visualmente os problemas e suas inter-relações em um *pôster*.
Mapa Mental: Método que possibilita a organização e conexão de descobertas, ideias e informações de forma concisa e estruturada. Ao representar conceitos de maneira gráfica e interconectada, ele simplifica a comunicação e a compreensão de tópicos complexos, enquanto também facilita a avaliação e o refinamento de ideias. **Sugestão de Tarefa:** Os professores podem orientar os alunos a criar mapas mentais para organizar conceitos e lógica de programação. Durante uma aula prática, os alunos podem escolher um tópico específico, como estruturas de controle de fluxo ou manipulação de dados, e desenvolver um mapa mental que represente visualmente os principais conceitos e suas inter-relações. Os alunos podem utilizar cores, ícones e palavras-chave para destacar elementos importantes, tornando o aprendizado mais visual e interativo.
Matriz 2x2: Método para o agrupamento e análise que permite uma abordagem sistêmica e abrangente das possíveis soluções para um problema. Nesta matriz, as percepções são organizadas em quatro quadrantes, resultantes da combinação de dois conjuntos de extremos ou variáveis-chave. Essa estruturação facilita a identificação de padrões, tendências e relações entre diferentes abordagens. **Sugestão de Tarefa:** Os professores podem conduzir uma atividade em que os alunos utilizem a Matriz 2x2 para explorar diferentes abordagens na resolução de problemas. Por exemplo, identificar extremidades opostas em dois eixos, como "Complexidade do Código" (horizontal) e "Eficiência do Algoritmo" (vertical). Cada quadrante da matriz representaria uma combinação desses atributos, como "Código Simples e Algoritmo Eficiente", "Código Complexo e Algoritmo Eficiente", "Código Simples e Algoritmo Ineficiente" e "Código Complexo e Algoritmo Ineficiente". Os alunos, então, podem discutir e desenvolver soluções de programação que se encaixem em cada categoria, promovendo uma compreensão mais ampla das escolhas de *design* e eficácia na programação introdutória.
MVP - Mínimo Produto Viável: Versão inicial e simplificada de um produto ou solução, projetada especificamente para testar seu núcleo funcional com usuários reais. É uma manifestação tangível ou funcional que visa validar hipóteses e coletar *feedback* para orientar o desenvolvimento futuro. **Sugestão de Tarefa:** Os professores podem orientar os alunos a desenvolver um MVP em programação introdutória. Os alunos seriam incentivados a identificar e implementar apenas os recursos essenciais necessários para resolver um problema,

sem adicionar funcionalidades excessivas. Após a conclusão do MVP, os alunos podem testar e avaliar a eficácia da solução, além de coletar *feedback* de colegas.
Obter *Feedback*: Processo de utilizar abordagens sistemáticas para testar ideias, produtos ou soluções com usuários reais, com o objetivo de receber retornos e esclarecimentos. Essencial para avaliar a eficácia, usabilidade e relevância de uma proposta, permitindo ajustes e melhorias. **Sugestão de Tarefa:** Os professores podem instruir os alunos a realizar sessões de obtenção de *feedback* para os projetos de programação que desenvolveram. Isso pode envolver a apresentação de seus programas para outros colegas de classe, permitindo que eles testem as funcionalidades e forneçam *feedback* construtivo. Além disso, os alunos podem criar questionários simples ou formulários eletrônicos para coletar *feedback* mais estruturado.
Personas: Arquétipos fictícios baseados na análise de comportamentos e características demográficas dos usuários reais. Importantes para compreender e representar as necessidades, motivações e expectativas do público-alvo de um produto ou serviço, permitindo uma abordagem mais centrada no usuário durante o processo de *design* desenvolvimento. **Sugestão de Tarefa:** Os professores podem orientar os alunos a criar personas para os projetos de programação que estão desenvolvendo. Isso envolverá identificar os potenciais usuários de seus programas, compreender suas características, necessidades e comportamentos. Os alunos podem utilizar essas personas como referência ao tomar decisões de *design* e implementação
Pesquisa *Desk:* Método que envolve a busca sistemática de informações em diversas fontes, como documentos, artigos, bancos de dados e publicações *online*, para aprofundar o conhecimento sobre o tema específico de um projeto. **Sugestão de Tarefa:** Os professores podem incentivar os alunos a realizar pesquisas em fontes variadas para aprofundar seu entendimento sobre os problemas ou desafios que estão tentando resolver por meio da programação. Isso pode incluir a busca por informações em livros, artigos *online*, revistas especializadas e outros recursos disponíveis.
Pesquisa Exploratória: Abordagem que envolve a observação e interação direta com as pessoas envolvidas no contexto do problema, visando obter percepções sobre suas necessidades, desafios e comportamentos. **Sugestão de Tarefa:** Os professores podem incentivar os alunos a realizar pesquisas exploratórias em campo para entender melhor o contexto em que os problemas de programação estão inseridos. Isso pode envolver a observação de usuários em situações reais, a realização de entrevistas ou até mesmo a participação em eventos relacionados ao tema do projeto.
Piloto: Versão inicial e parcialmente implementada da solução que está sendo desenvolvida, disponibilizada para avaliação e teste por usuários reais antes da conclusão. **Sugestão de Tarefa:** Os professores podem orientar os alunos a criar versões parciais implementadas de suas soluções de programação. Essas implementações podem ser avaliadas por um grupo seleto de usuários reais, como colegas de classe ou membros da comunidade acadêmica. Essa abordagem permite que os alunos testem a usabilidade, identifiquem possíveis melhorias e avaliem a eficácia de suas soluções em um ambiente controlado antes de uma implementação completa.
Pitch: Apresentação breve e direta de uma ideia ou solução. Geralmente, é estruturado de forma a destacar os pontos-chave da proposta, como o problema a ser resolvido, a solução oferecida, o mercado-alvo, o modelo de negócios e os diferenciais competitivos. **Sugestão de Tarefa:** Os professores podem incentivar os alunos a realizar *pitches* de seus projetos de programação, destacando os principais pontos, como a finalidade da solução, os benefícios oferecidos e a lógica por trás do código desenvolvido.
Protótipo: Iteração ágil de um conceito ou ideia, que utiliza uma versão simplificada da solução para testar sua viabilidade e aceitação. **Sugestão de Tarefa:** Os professores podem orientar os alunos a criar protótipos simples de suas soluções de programação. Isso envolve a criação de versões iniciais funcionais do programa ou aplicativo que estão desenvolvendo. Os alunos podem apresentar esses protótipos em sala de aula, permitindo que seus colegas forneçam *feedback*.
Prova de Conceito: Avaliação preliminar do potencial de uma ideia ou conceito em seus estágios iniciais de concepção, com o objetivo de determinar sua viabilidade técnica, operacional e econômica. Ao fornecer percepções sobre os desafios e oportunidades associados à ideia, ajuda a orientar decisões futuras sobre investimentos e desenvolvimento. **Sugestão de Tarefa:** Os professores podem orientar os alunos a realizar uma prova de conceito para validar a viabilidade de uma ideia de programação que estão explorando. Isso envolve criar uma versão simplificada do código para testar a funcionalidade central da ideia. Os alunos podem realizar experimentos, analisar resultados e determinar se a abordagem inicial é viável antes de prosseguir com o desenvolvimento completo.
Resumo de Soluções: Definição clara e organizada da solução desejada através de uma planilha estruturada. Essa planilha articula os requisitos essenciais e o nível de desenvolvimento necessário para alcançar a solução pretendida. **Sugestão de Tarefa:** Os professores podem orientar os alunos a criar um resumo de soluções para um problema específico de programação. Os alunos podem utilizar uma planilha organizada para detalhar os requisitos essenciais da solução, descrever as funcionalidades desejadas e estabelecer o nível de desenvolvimento esperado.
Roadmap: Descreve as ações e responsabilidades necessárias para a implementação bem-sucedida do produto final em um período de tempo específico. Essa ferramenta fornece uma visão geral estratégica do processo de desenvolvimento, destacando marcos importantes, prazos e recursos necessários. **Sugestão de Tarefa:** Os professores podem instruir os alunos a desenvolverem um *roadmap* para a implementação de um projeto de programação ao longo de um período definido. Os alunos devem identificar as etapas-chave do desenvolvimento, atribuir responsabilidades a diferentes membros da equipe (ou a si mesmos) e estabelecer marcos importantes para avaliação do progresso.
Sessão de Cocriação: Encontro participativo onde diversos *stakeholders* se envolvem ativamente na geração e avaliação de ideias, visando a produção colaborativa de soluções ou conceitos. Essa abordagem promove a inclusão de perspectivas diversas e facilita a construção de consenso, resultando em propostas mais alinhadas com as necessidades e expectativas de todos os envolvidos.

Sugestão de Tarefa: Os professores podem organizar uma sessão de cocriação em sala de aula, dividindo os alunos em grupos e atribuindo a cada grupo um problema específico de programação para resolver. Durante a sessão, os alunos são incentivados a gerar ideias colaborativamente, compartilhar conhecimentos e discutir abordagens para solucionar o problema. Ao final, cada grupo apresenta suas soluções e recebe *feedback* construtivo dos colegas e do professor.
Storyboard: Consiste em criar um protótipo de baixo nível, composto por uma série de quadros estáticos, para visualizar e discutir uma ideia ou conceito de forma sequencial. Permite contar uma história visualmente, delineando a narrativa, os eventos e as interações entre os elementos-chave da proposta. **Sugestão de Tarefa:** Os professores podem orientar os alunos a criar *storyboards* para representar visualmente o processo de desenvolvimento de um programa ou aplicação. Os alunos podem dividir o projeto em etapas, representando cada passo por meio de quadros, desenhos ou diagramas.
Storytelling: Método que utiliza narrativas animadas e dinâmicas, em vez de elementos estáticos, para comunicar e testar uma ideia ao longo do tempo. Pode assumir diversas formas, como uma apresentação, vídeo ou até mesmo uma encenação ao vivo. **Sugestão de Tarefa:** Os professores podem desafiar os alunos a criar um projeto de programação acompanhado por uma narrativa ou história. Cada etapa do código pode ser associada a um desenvolvimento específico na história. Os alunos podem utilizar animações simples ou áudio para criar uma experiência mais envolvente.

O ensino tradicional de programação, amplamente utilizado, baseia-se predominantemente em aulas expositivas e exercícios práticos focados em sintaxe, estrutura de algoritmos e resolução de problemas específicos (Edwards *et al*., 2020). Nessa abordagem, os alunos geralmente são expostos a conteúdos de maneira sequencial e linear, com pouca ênfase na contextualização ou aplicação prática em cenários mais amplos (Tareq; Yusof, 2024). A avaliação do aprendizado tende a ser feita por meio de tarefas individuais, como a implementação de programas simples e testes escritos, que medem a compreensão técnica de conceitos, mas nem sempre promovem o desenvolvimento de habilidades colaborativas ou criativas (Mehmood *et al*., 2020). Embora eficiente para transmitir fundamentos técnicos, essa metodologia pode apresentar limitações, como a dificuldade em engajar os estudantes e a ausência de estratégias para abordar diferentes estilos de aprendizado (De Vega, 2022). Esses fatores podem contribuir para o desinteresse e, possivelmente, para as altas taxas de abandono em disciplinas introdutórias de programação (Figueiredo; García-Peñalvo, 2021).

Com o uso do catálogo de métodos HCD, o ensino de programação pode adotar uma abordagem mais centrada no aluno, promovendo a construção de conhecimento por meio de atividades colaborativas, práticas e contextualizadas. Os métodos identificados no catálogo podem incentivar o envolvimento ativo dos estudantes, desafiando-os a resolver problemas reais e significativos, ao mesmo tempo em há a possibilidade desenvolver habilidades analíticas. Exemplos incluem atividades de *brainstorming* para solução de problemas, construção de protótipos para simulação de sistemas e análise de casos práticos que conectam conceitos teóricos à sua aplicação no mundo real. Essa abordagem tem a capacidade de possibilitar uma maior flexibilidade no ensino, adaptando-se às necessidades de diferentes contextos educacionais e estilos de aprendizado. Além disso, ao envolver os alunos de maneira mais participativa, o catálogo visa reduzir a sensação de desconexão e complexidade frequentemente associada às disciplinas introdutórias, criando um ambiente mais engajador e propício ao aprendizado significativo.

A concepção do catálogo de métodos HCD para o ensino de programação introdutória ainda é uma proposta inicial. Este catálogo, inédito no contexto acadêmico e educacional, serve como um ponto de partida para a integração de métodos HCD no contexto educacional, fornecendo descrições detalhadas e sugestões de uso, mas reconhecendo a flexibilidade necessária para adaptação às especificidades de cada turma e aluno. A criação de um catálogo dessa natureza é uma contribuição significativa, uma vez que não existe uma compilação sistemática e acessível de métodos HCD especificamente direcionada ao ensino de programação introdutória. Esse recurso oferece aos educadores uma base inicial para experimentar e integrar práticas inovadoras, enriquecendo o processo de ensino-aprendizagem e potencialmente melhorando os resultados educacionais.

6. Discussão

Os resultados preliminares indicam que os métodos HCD possuem potencial para tornar o ensino de programação mais acessível e compreensível. A utilização de métodos variados permite aos educadores adaptar suas práticas pedagógicas às necessidades específicas dos alunos, promovendo uma aprendizagem mais personalizada. Este aspecto é particularmente relevante no ensino de programação introdutória, onde os alunos frequentemente enfrentam desafios relacionados à abstração e compreensão de conceitos complexos.

A aplicação do catálogo de métodos HCD no ensino de programação introdutória demonstra uma promessa significativa. Ao integrar esses métodos, os educadores podem diversificar suas abordagens de ensino. Utilizando diferentes métodos, os professores conseguem abordar os diversos estilos de aprendizagem dos alunos, aumentando a acessibilidade e a compreensão dos conteúdos. Além disso, esses métodos estimulam a criatividade e a inovação, incentivando os alunos a explorar diferentes soluções para problemas de programação, desenvolvendo suas habilidades criativas e de resolução de problemas. Muitos métodos HCD também favorecem o trabalho em grupo, fortalecendo as habilidades de comunicação e colaboração entre os alunos.

Entretanto, os resultados indicam que, embora a diversidade de métodos seja vantajosa, há uma necessidade de adaptação e simplificação para torná-los mais acessíveis e menos morosos para os educadores e estudantes de programação introdutória. É essencial equilibrar a variedade de métodos com o tempo disponível para sua aplicação em sala de aula. Esta adaptação deve considerar o nível de maturidade dos alunos e a complexidade dos métodos para garantir que o aprendizado não seja comprometido por uma sobrecarga de informações ou práticas inadequadas.

Para aumentar a eficácia do catálogo, algumas evoluções podem ser consideradas. Primeiramente, o desenvolvimento de tutoriais, guias passo a passo e exemplos práticos de aplicação dos métodos pode ajudar os professores a implementar tais técnicas. Além disso, oferecer workshops e treinamentos específicos sobre como integrar os métodos HCD no ensino de programação pode capacitar os professores a usar essas ferramentas de maneira otimizada, aumentando a confiança e a habilidade em aplicá-los. Outra estratégia importante é a implementação de um processo de avaliação contínua do impacto dos métodos HCD no aprendizado dos alunos, ajustando e refinando o catálogo com base no feedback e nos resultados observados.

O catálogo apresentado neste estudo é uma proposta que pode servir como base para o desenvolvimento de abordagens e ferramentas mais específicas, como diretrizes pedagógicas e *frameworks* de ensino. A elaboração deste catálogo abre novas oportunidades para a aplicação prática dos métodos HCD no ensino de programação, permitindo, por exemplo, que educadores adaptem suas práticas pedagógicas. Ao reforçar a ideia do desenvolvimento de abordagens e ferramentas mais específicas, este catálogo pode ser transformado e aplicado na prática, proporcionando um guia prático e flexível que auxilie tanto professores quanto alunos no desenvolvimento de habilidades de programação introdutória.

7. Conclusão

Este estudo apresentou um Catálogo de Métodos HCD destinado ao ensino de programação introdutória. Este catálogo inclui uma variedade de métodos, cada um detalhado com descrições e sugestões de aplicação prática. A diversidade e a flexibilidade dos métodos oferecem aos educadores uma ampla gama de ferramentas para enriquecer suas práticas pedagógicas, promovendo uma abordagem centrada no aluno e potencialmente aumentando o engajamento no ensino de programação.

Os resultados da validação com os professores revelaram que os métodos HCD selecionados são relevantes e aplicáveis ao processo de ensino e aprendizagem de programação introdutória. Os professores participantes reconheceram a importância e a variedade dos métodos, destacando como estes podem não apenas desenvolver habilidades específicas de

programação, mas também melhorar o processo de ensino em geral. Métodos como *Brainstorming*, Cardápio de Ideias, Obter *Feedback*, Pesquisa *Desk* e Pesquisa Exploratória foram especialmente valorizados na avaliação dos educadores por promoverem habilidades colaborativas e analíticas, fundamentais no aprendizado da programação.

Contudo, foram identificados desafios relacionados à aplicação prática dos métodos, especialmente quanto à complexidade de alguns deles e à limitação de tempo disponível em sala de aula. Isso levou à recomendação de simplificação dos métodos e à adaptação dos mesmos para melhor se adequarem ao contexto específico de ensino de programação introdutória. A necessidade de equilibrar a diversidade de métodos com o tempo de aula disponível foi uma preocupação central expressa pelos professores, indicando que a flexibilidade na aplicação e a adaptação dos métodos às necessidades dos alunos são cruciais.

Com base no *feedback* dos professores, foram realizadas correções e melhorias nos exemplos apresentados inicialmente, tornando-os mais acessíveis e práticos para os educadores. O catálogo resultante oferece descrições detalhadas de cada método HCD e sugestões de aplicação, permitindo aos educadores escolher e adaptar os métodos conforme as necessidades específicas de suas turmas.

Portanto, este catálogo de métodos HCD pode representar uma ferramenta para educadores que buscam enriquecer o processo de ensino e aprendizagem de programação introdutória. Embora não seja necessário aplicar todos os métodos, a flexibilidade e a adaptabilidade do catálogo permitem que os educadores selecionem e utilizem os métodos que melhor se alinhem com suas práticas pedagógicas e com as necessidades de seus alunos.

Embora os resultados sejam promissores, algumas limitações do estudo devem ser mencionadas. A aplicação do estudo de caso com apenas seis professores de computação pode limitar a abrangência dos resultados. No entanto, a experiência significativa desses professores, com média de 7 anos de docência, contribuiu para o refinamento do catálogo. Além disso, a aplicação prática do catálogo em sala de aula ainda não foi realizada, o que sugere a necessidade de estudos futuros para validar sua eficácia diretamente com os alunos.

Estudos futuros: Para futuras pesquisas, sugere-se a construção de um guia prático para a aplicação dos métodos HCD, com o objetivo de fomentar as habilidades de programação introdutória dos alunos. Estudos futuros devem incluir amostras maiores e a implementação prática desses métodos em sala de aula, permitindo a validação e o refinamento contínuo do catálogo. A análise do impacto dos métodos HCD no desempenho e engajamento dos alunos em diferentes contextos educacionais será fundamental para consolidar a eficácia do catálogo e oferecer orientações mais precisas para educadores. Além disso, vislumbra-se a possibilidade de estabelecer um *framework* baseado nesse catálogo inicial. Esse *framework* pode ser transformado em uma ferramenta prática que permita a personalização e adaptação dos métodos HCD às diversas realidades educacionais. O desenvolvimento do *framework* incluiria diretrizes claras para a aplicação de cada método, exemplos de melhores práticas e recomendações baseadas em dados empíricos.

Agradecimentos

Os autores agradecem ao apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código Financeiro 001, do Conselho Nacional de Desenvolvimento Cientifico e Tecnológico (CNPq), da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP - Processo 2019/26871-4), e da Especialização em Computação Aplicada à Educação (CAE - ICMC/USP).

Referências

ALTURKI, R. A. Measuring and improving student performance in an introductory programming course. **Informatics in Education-An International Journal**, v. 15, n. 2, p. 183-204, 2016.

BENNEDSEN, J.; CASPERSEN, M. E. Failure rates in introductory programming: 12 years later. **ACM inroads**, ACM New York, NY, USA, v. 10, n. 2, p. 30–36, 2019.

BERSSANETTE, J. H.; FRANCISCO, A. C. Percepção de Docentes que lecionam Programação de Computadores quanto à Formação Pedagógica. **Revista Brasileira de Informática na Educação**, v. 29, p. 133-159, 2021.

BLANCO, T.; LÓPEZ-FORNIÉS, I.; ZARAZAGA-SORIA, F. J. Deconstructing the Tower of Babel: a design method to improve empathy and teamwork competences of informatics students. **International Journal of Technology and Design Education**, v. 27, p. 307-328, 2017.

CHANGPETCH, C.; PANJABUREE, P.; SRISAWASDI, N. A comparison of pre-service teachers' variable misconceptions in various computer-programming preferences: findings to teacher education course. **Journal of Computers in Education**, v. 9, n. 2, p. 149-172, 2022.

CHEAH, C. S. Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. **Contemporary Educational Technology**, v. 12, n. 2, p. ep272, 2020.

DE VEGA, F. F. Teaching Programming in the 21st Century. **Journal of Computer Information Systems**, v. 63, n. 4, p. 841-852, 2023.

DU, J.; WIMMER, H.; RADA, R. "Hour of Code": Can It Change Students' Attitudes Toward Programming?. **Journal of Information Technology Education: Innovations in Practice**, v. 15, p. 53, 2016.

DURAN, R.; BIM, S. A.; GIMENES, I.; RIBEIRO, L.; CORREIA, R. C. M. Potential Factors for Retention and Intent to Drop-out in Brazilian Computing Programs. **ACM Transactions on Computing Education**, v. 23, n. 3, p. 1-33, 2023.

EDWARDS, J.; DITTON, J.; TRNINIC, D.; SWANSON, H.; SULLIVAN, S.; MANO, C. Syntax exercises in CS1. In: **Proceedings of the 2020 ACM Conference on International Computing Education Research**. 2020. p. 216-226.

FIGUEIREDO, J.; GARCÍA-PEÑALVO, F. Teaching and learning tools for introductory programming in university courses. In: **2021 International Symposium on Computers in Education (SIIE)**. IEEE, 2021. p. 1-6.

GRACE, K.; KLAASSENS, B.; BRAY, L.; ELTON-PYM, A. An open-ended blended approach to teaching interaction designers to code. **Frontiers in Computer Science**, v. 4, p. 813889, 2022.

IDEO.org. The field guide to human centered design. 2015. Disponível em: https://x.gd/NU4cqx.

LAI, C.; Zhong, H. X.; CHANG, J. H.; CHIU, P. S. Applying the DT-CDIO engineering design model in a flipped learning programming course. **Educational technology research and development**, v. 70, n. 3, p. 823-847, 2022.

MEDEIROS, R. P.; FALCÃO, T. P.; RAMALHO, G. L. Comparação entre o panorama internacional e nacional sobre o Ensino e a Aprendizagem de Introdução à Programação no Ensino Superior. In: **Anais do XXIX Workshop sobre Educação em Computação**. SBC, 2021. p. 478-487.

MEHMOOD, E; ABID, A.; FAROOQ, M. S.; NAWAZ, N. A. Curriculum, teaching and learning, and assessments for introductory programming course. **IEEE Access**, v. 8, p. 125961-125981, 2020.

NESTA. **The collective intelligence design playbook.** Tools, tactics and methods to harness the power of people, data and technology to solve global challenges. 2019. Disponível em: https://media.nesta.org.uk/documents/Nesta_Playbook_001_Web.pdf.

OLIVEIRA, K. K. S.; DEUS, W. S.; AVELLAR, G. M. N., FALCÃO, T. P.; BARBOSA, E. F. Análise dos Cursos de Computação no Brasil: Ensino de Programação sob a perspectiva das Necessidades do Século XXI. In: **Anais do XXXIV Simpósio Brasileiro de Informática na Educação**. SBC, 2023a. p. 1760-1771.

OLIVEIRA, K. K. S.; MARCOLINO, A. D. S.; DEUS, W. S. D.; FALCÃO, T. P. D. R.; Barbosa, E. F. Pensamento computacional na programação introdutória e habilidades do século XXI: um mapeamento

sistemático da literatura. **Revista Novas Tecnologias na Educação-RENOTE**, v. 21, n. 2, p. 519-531, 2023b.

OLIVEIRA, K. K. S.; MARCOLINO, A. S.; FALCÃO, T. P.; BARBOSA, E. F. Ensino e Aprendizagem de Programação na Educação 4.0: Um Mapeamento Sistemático da Literatura. **Simpósio Brasileiro de Educação em Computação (EDUCOMP)**, p. 245-255, 2024.

OLIVEIRA, K. K. S.; SOUZA, R. A. C. Digital transformation towards education 4.0. **Informatics in Education**, v. 21, n. 2, p. 283-309, 2022.

PARK, H.; MCKILLIGAN, S. A systematic literature review for human-computer interaction and design thinking process integration. In: **Design, User Experience, and Usability: Theory and Practice: 7th International Conference, DUXU 2018, Held as Part of HCI International 2018, Las Vegas, NV, USA, July 15-20, 2018, Proceedings, Part I 7**. Springer International Publishing, 2018. p. 725-740.

PEDROSA, D.; CRAVINO, J.; MORGADO, L.; BARREIRA, C. Self-regulated learning in higher education: strategies adopted by computer programming students when supported by the SimProgramming approach. **Production**, v. 27, n. spe, p. e20162255, 2017.

REVANO, T. F.; GARCIA, M. B. Manufacturing design thinkers in higher education institutions: The use of design thinking curriculum in the education landscape. In: **2020 IEEE 12th international conference on humanoid, nanotechnology, information technology, communication and control, environment, and management (HNICEM)**. IEEE, 2020. p. 1-5.

ROBINS, A. Learning edge momentum: A new account of outcomes in CS1. **Computer Science Education**, v. 20, n. 1, p. 37-71, 2010.

RÕÕM, M.; LEPP, M.; LUIK, P. Dropout time and learners' performance in computer programming MOOCs. **Education Sciences**, v. 11, n. 10, p. 643, 2021.

SCHEER, A.; NOWESKI, C.; MEINEL, C. Transforming constructivist learning into action: Design thinking in education. **Design and Technology Education**, v. 17, n. 3, p. 8-19, 2012.

SCHOOLS2030. **Schools 2030 human-centered design toolkit**. 2021. Disponível em: https://schools2030.org/wp-content/uploads/2021/10/Schools-2030-Educator-Toolkit_FINAL.pdf.

SILVA, L. R. C.; DAMACENO, A. D. MARTINS, M. D. C. R.; SOBRAL, K. M.; FARIAS, I. M. S. D. Pesquisa documental: alternativa investigativa na formação docente. In: **Congresso Nacional de Educação**. 2009. p. 4554-4566.

SIMONSEN, J.; ROBERTSON, T. (Ed.). **Routledge international handbook of participatory design**. New York: Routledge, 2013.

STANFORD. **D.school. d.school design project guide**. 2016. Disponível em: https://x.gd/Yfb3o.

STANFORD. **D.school. design thinking bootleg**. 2018. Disponível em: https://x.gd/QMLj3.

TAREQ, Z. A.; YUSOF, R. J. R. Modeling a Problem-Solving Approach Through Computational Thinking for Teaching Programming. **IEEE Transactions on Education**, 2024.

TSAI, M. J.; WANG, C. Y. Assessing young students' design thinking disposition and its relationship with computer programming self-efficacy. **Journal of Educational Computing Research**, v. 59, n. 3, p. 410-428, 2021.

VIANNA, M.; VIANNA Y.; ADLER, I. K.; LUCENA, B.; RUSSO, B. **Design thinking: inovação em negócios**. Design Thinking, 2012.

VILLAMOR, M. M. A review on process-oriented approaches for analyzing novice solutions to programming problems. **Research and Practice in Technology Enhanced Learning**, v. 15, n. 1, p. 8, 2020.

VIRGUEZ, L.; DICKRELL, P. L.; GONCHER, A. Utility Value of an Introductory Engineering Design Course: An evaluation among Course Participants. In: **2020 ASEE Virtual Annual Conference Content Access**. 2020.

WATSON, C.; LI, F. W. B. Failure rates in introductory programming revisited. In: **Proceedings of the 2014 conference on Innovation & technology in computer science education**. 2014. p. 39-44.

XIE, B.; LOKSA, D.; NELSON, G. L.; DAVIDSON, M. J.; DONG, D.; KWIK, H.; KO, A. J. A theory of instruction for introductory programming skills. **Computer Science Education**, v. 29, n. 2-3, p. 205-253, 2019.