APAP: Arquitetura Pedagógica para Aprendizagem de Programação baseada no Pensamento Computacional

Francisco Xavier da Silva[1], Crediné Silva de Menezes[2], Alberto Nogueira de Castro Junior[3],
[1] IFMS - Campus Coxim - Coxim - MS – Brasil
[2] PPGIE - UFRGS - Porto Alegre, RS – Brasil
[3] IComp - UFAM - Manaus, AM – Brasil
Francisco.silva@ifms.edu.br, credine@gmail.com,
alberto@icomp.ufam.edu.br

Abstract. *This article presents the conception of a Pedagogical Architecture based on Computational Thinking to support Learning Programming (APAP). APAP aims to help students learn to program, understanding problems and building solutions cooperatively, using Computational Thinking. The framework is anchored in the theoretical assumptions of Piaget's genetic epistemology and is mediated by peer review, promoting cognitive development and cooperation among students. Experimental results indicate that, by using a Pedagogical Architecture for problem solving integrated with Computational Thinking, students have been shown to internalize the concepts learned and develop critical skills for programming.*

Resumo. *Este artigo apresenta a concepção de uma Arquitetura Pedagógica baseada no Pensamento Computacional para apoio à Aprendizagem de Programação (APAP). A APAP visa ajudar os estudantes a aprender a programar, compreendendo problemas e construindo soluções de forma cooperativa, utilizando o Pensamento Computacional. A estrutura é ancorada nos pressupostos teóricos da epistemologia genética de Piaget e é mediada pela revisão por pares, promovendo o desenvolvimento cognitivo e a cooperação entre estudantes. Resultados experimentais indicam que, ao usarem uma Arquitetura Pedagógica para a resolução de problemas integrada com o Pensamento Computacional, os estudantes demonstraram internalizar os conceitos aprendidos e desenvolver habilidades críticas para a programação.*

1. Introdução

A disciplina de Introdução à Programação está presente em vários cursos técnicos de computação e, normalmente, é oferecida no primeiro ano, com o objetivo de desenvolver a lógica de programação do estudante, tornando-o capaz de construir algoritmos corretos e eficientes. Porém, aprender programação é muito mais do que escrever um conjunto de linhas de código em uma dada linguagem. Requer uma grande diversidade de pensamentos e compreensões. Em outras palavras, é também uma arte, pois existem muitas maneiras de entender, combinar, reconhecer padrões e codificar a solução dos problemas com criatividade.

Essa disciplina, na maioria das vezes, não considera os conhecimentos prévios dos estudantes, o que frequentemente os deixam frustrados. Dentre os motivos para essa

DOI: 10.5753/wie.2026.28067

frustração, destacam-se: dificuldade em compreender o problema; inadequação dos métodos pedagógicos aos estilos de aprendizagem dos estudantes; preocupação excessiva com detalhes de sintaxe da linguagem utilizada; e punição ao erro, inviabilizando o pensamento crítico e criativo. Essas dificuldades culminam em baixa motivação, baixa autoestima e elevados índices de reprovação e desistência [Prolux 2000].

Na busca por contribuir para a resolução desses problemas, diferentes abordagens pedagógicas têm sido estudadas e propostas na literatura, tais como: salas de fuga educacionais [López-Pernas et al. 2019]; uso de linguagens especiais, como Scratch [Morales-Urrutia et al. 2021]; sala de aula invertida e programação no contexto de histórias digitais [Zha et al., 2020]; aprendizagem cooperativa [Garcia 2021]; e aprendizagem baseada em jogos [Giannakoulas; Xinogalos 2018].

Dessa forma, nossa contribuição baseia-se em uma Arquitetura Pedagógica [Carvalho; Nevado; Menezes 2005], ancorada nos pressupostos teóricos da Epistemologia Genética [Piaget 2016] e no Pensamento Computacional [Wing 2006], sendo mediada pela revisão por pares [Mazur 1999].

Este artigo está organizado da seguinte forma: a Seção 2 contém a fundamentação teórica; a Seção 3 descreve uma Arquitetura Pedagógica Baseada no Pensamento Computacional para Apoio à Aprendizagem de Programação (APAP); a Seção 4 Experimento de Ensino; E, na Seção 5, são expostas as considerações finais.

2. Fundamentação teórica

Nesta seção, serão abordados conceitos essenciais que constituem a fundamentação teórica da proposta, destacando-se interações na aprendizagem do ponto de vista piagetiano, pensamento computacional e a arquitetura pedagógica.

2.1. Tríade da aprendizagem de Piaget

Segundo Piaget (2013), o conhecimento prévio do estudante é fundamental nas interações com seus pares, nas trocas de informações e na forma de revisar as atividades, construindo, assim, situações de desequilíbrios momentâneos e reequilíbrios. A teoria da equilibração proposta por Piaget (2013) e as etapas de construção do conhecimento descritas por Piaget e García (1982) buscam identificar leis gerais que permitam compreender a sucessão das formas de conhecimento e dos mecanismos que as engendram. Os autores identificam, tanto na psicogênese quanto na história das ciências, mecanismos comuns envolvidos na construção desses conhecimentos.

Para esses autores, o elemento de maior importância nesse estudo comparativo consiste na tríade dialética constituída pelas etapas INTRA, INTER e TRANS. Tais etapas ajudam no desenvolvimento de novos esquemas, dinâmicos e mutáveis, que crescem de acordo com certos mecanismos, desenvolvendo todos os domínios do pensamento humano.

Ainda segundo esses autores, o primeiro elemento da tríade INTRA, é característico das primeiras abordagens de um novo domínio, facilmente observável nas representações infantis do estágio pré-operatório. Nesse estágio, os elementos essenciais são os observáveis diretamente. À medida que o desenvolvimento avança para o nível INTER, surgem relações e transformações a partir da multiplicação dos esquemas identificados no nível anterior, gerando contradições que o sujeito precisa superar.

Nesse esforço, o indivíduo começa a trabalhar com relações entre variáveis, em vez de analisar cada uma isoladamente, promovendo um entendimento mais complexo da realidade.

Finalmente, no nível TRANS, há uma maior generalidade e uma compreensão mais articulada, permitindo que o modelo teórico não só constate regularidades, mas também preveja e demonstre fenômenos, refletindo um avanço significativo na compreensão e aplicação do conhecimento.

2.2. Pensamento Computacional

O artigo seminal de Wing (2006), definiu o Pensamento Computacional (PC) como: uma habilidade fundamental para todos, não apenas para cientistas da computação, sendo constituído por um processo mental que generaliza um problema do mundo real a um problema computacional, o pensamento de solução do problema que envolve habilidades como abstração, decomposição, algoritmos, iteração, avaliação, a resolução de problemas e a concepção de sistemas. Trata-se de um processo de pensamento envolvido na formulação e reformulação de um problema e na expressão de sua(s) solução(ões) de forma que um computador — humano ou máquina — possa efetivamente executar.

Adicionalmente, Roman-Gonzales et al. (2017) identificaram uma correlação significativa entre as habilidades de Pensamento Computacional e a sintaxe de programação. Segundo os autores, a resolução de problemas exerce maior influência sobre o uso adequado da sintaxe de programação do que outras habilidades relacionadas ao Pensamento Computacional.

De acordo com Motil e Epstein (2000), a maioria das linguagens de programação utilizadas em disciplinas introdutórias apresenta sintaxe extensa e complexa. Os autores destacam que, para muitos estudantes, essa complexidade pode gerar frustração e contribuir para a desistência dos cursos de programação.

2.3. Arquiteturas Pedagógicas

Segundo Carvalho, Nevado e Menezes (2005), as Arquiteturas Pedagógicas (APs) são definidas como suportes estruturantes, maleáveis e adaptáveis a diferentes enfoques temáticos, incluindo pedagogias relacionais, métodos ativos, abordagens cooperativas, tecnologias digitais, inteligência artificial e reconfigurações dos tempos e espaços de aprendizagem.

De acordo com Menezes, Castro Junior e Aragón (2021), os elementos essenciais para a descrição e concepção de uma Arquitetura Pedagógica são: domínio de conhecimento; objetivos educacionais; conhecimento prévio; dinâmicas interacionista-problematizadoras; suporte computacional (tecnológico); mediações pedagógicas distribuídas (aprendizagens cooperativas); e avaliação processual e cooperativa das aprendizagens.

Nesse contexto, Carvalho, Nevado e Menezes (2005) destacam que um dos mecanismos pedagógicos fundamentais das Arquiteturas Pedagógicas é a aprendizagem cooperativa, que favorece a construção coletiva do conhecimento por meio da interação entre os estudantes.

3. Uma Arquitetura Pedagógica Baseada no Pensamento Computacional para Apoio à Aprendizagem de Programação

A Arquitetura Pedagógica baseada no Pensamento Computacional para Apoio à Aprendizagem de Programação (APAP) foi concebida a partir dos princípios do Pensamento Computacional (PC) aplicados à resolução de problemas. Sua estrutura integra um conjunto de habilidades que favorecem a participação ativa dos estudantes na construção de soluções cooperativas e no enfrentamento de desafios de programação.

Com base nos trabalhos de Selby e Woollard (2013), Barr e Stephenson (2011) e Dagienė, Futschek e Stupurienė (2019), foram agrupadas onze habilidades de PC que fundamentam a APAP: decomposição, reconhecimento de padrões, abstração, modelagem e simulação, algoritmos, implementação, sistemas digitais, especificação, interpretação de dados, representação de dados e avaliação.

A APAP busca promover a construção do conhecimento por meio da compreensão de problemas, da elaboração de soluções e da implementação computacional. Nesse processo, os estudantes desenvolvem habilidades cognitivas e estratégias de resolução de problemas, transformando erros e reflexões em oportunidades de aprendizagem.

A APAP se constitui em uma AP complexa e é composta de três microarquiteturas que fluem umas para as outras na operação real do desenvolvimento da aprendizagem de programação. As microarquiteturas AP1, AP2 e AP3 possibilitam a reflexão sobre escolhas feitas e interação entre as APs, levando-os a refletir, interagir e cooperar.

Podemos definir a APAP como uma a composição interativa das microarquiteturas AP1, AP2 e AP3, conforme Figura 1.

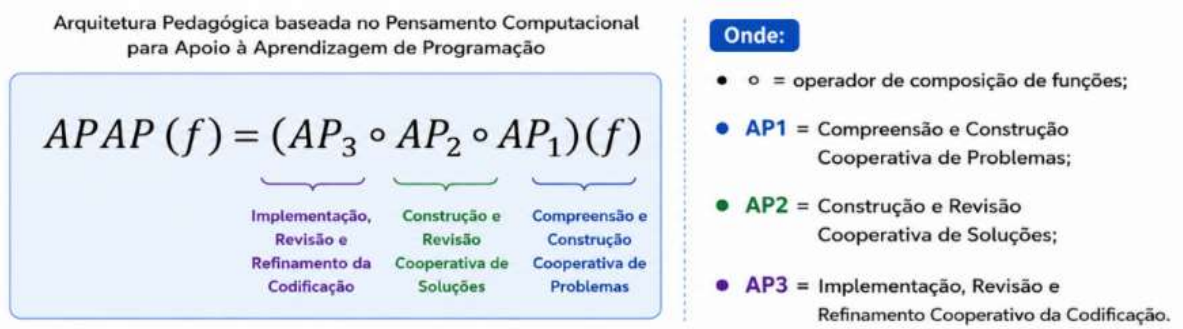

Figura 1: Composição da APAP

Cada AP é uma função que facilita a reflexão, interação e cooperação. Essa abordagem configura a APAP como uma função composta e permeável, na qual os estudantes são incentivados a refletir, interagir e cooperar, com a possibilidade de revisitar atividades pedagógicas anteriores sempre que necessário. A seguir, descreveremos mais detalhadamente cada uma das APs simples que compõem a APAP.

AP1 - O professor apresenta problemas ($Pa(f)$) que levam em conta as habilidades do PC e que contenham situações da vida cotidiana. Os estudantes utilizam estratégias pessoais para escolher ($Pe(f)$), no mínimo, um dos problemas apresentados e, em seguida, individualmente, propor outros problemas ($Pn(f)$), que serão

compartilhados com o grupo para que façam escolha coletiva ($Pc(f)$)de no mínimo um dos problemas, conforme descrito na função a seguir, conforme Figura 2.

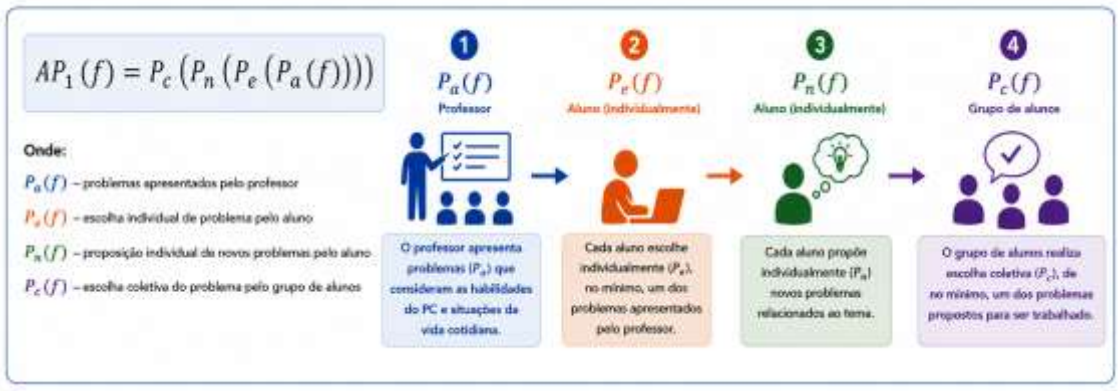

Figura 2: AP1 - Compreensão do Problema e Construção Cooperativa de Problemas

Esse processo incentiva a reflexão, a interação e a cooperação, permitindo que os estudantes se engajem de forma significativa e autônoma no aprendizado.

AP2 - Os estudantes constroem soluções individuais ($Sc(f)$), no formato de textos e screencasts, para os problemas escolhidos na AP1. Eles realizam revisões por pares dessas soluções de forma cooperativa ($Rp(f)$) e, por fim, apresentam suas soluções para o grupo ($Ag(f)$), descrito a seguir, conforme Figura 3.

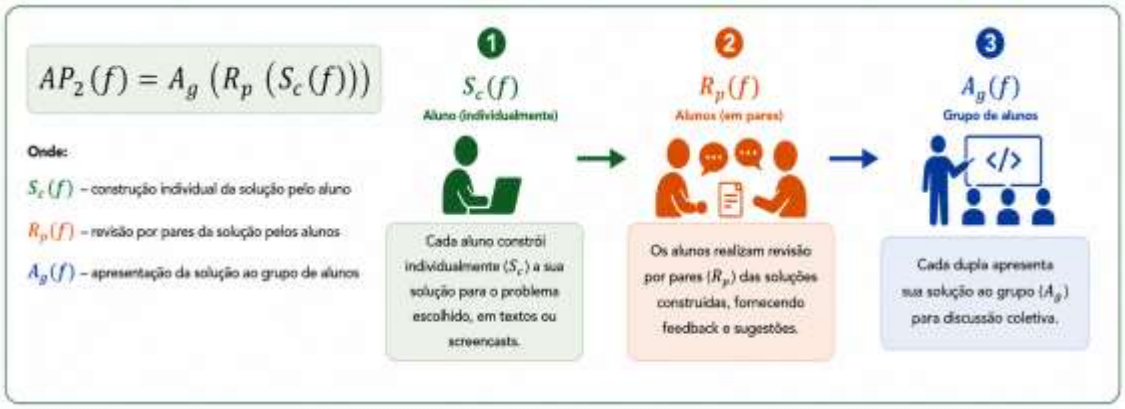

Figura 3: AP2 – Construção e Revisão Cooperativa de Soluções

Favorecendo a cooperação e a troca de ideias, permitindo que as soluções recebam críticas e sugestões. Em seguida, as soluções revisadas e aprimoradas são apresentadas ao grupo. Onde os estudantes se reúnem para discutir as propostas revisadas, combinando ideias, identificando pontos fortes e fracos, e desenvolvendo uma ou mais soluções para o grupo, garantindo que as melhores ideias de cada estudante sejam integradas em uma solução coesa e bem fundamentada. O resultado é uma solução que reflete a cooperação dos estudantes.

AP3 - Consiste em um sistema de codificação da resolução do problema realizada pelos pares para favorecer a interação e a construção de possíveis soluções para erros sintáticos (de sintaxe) e semânticos. Para favorecer a reflexão, cada estudante registra suas sugestões construindo um screencast e um documento compartilhado

($Rs(f)$. Por fim, ao receber feedback sobre seu código, o estudante construirá uma nova codificação ($Fc(f)$) e a apresentará à turma para que todos possam conhecer sua solução e, se necessário, oferecer novas alterações ($As(f)$), conforme Figura 4.

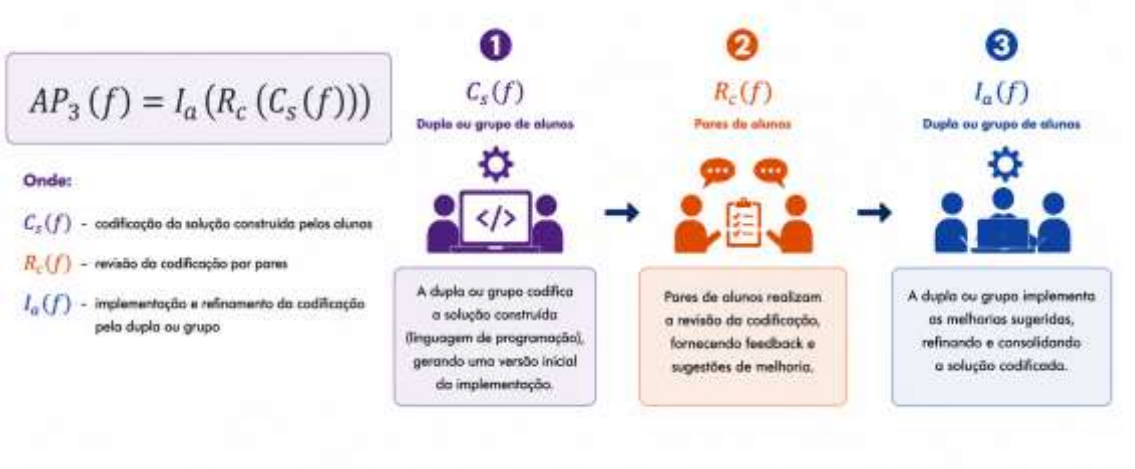

Figura 4: AP3- implementação, revisão e refinamento da codificação.

Esse processo de revisão estimula a interação e a reflexão, permitindo a identificação e correção de erros. Baseando-se no feedback recebido, os estudantes aprimoram sua codificação e preparam uma nova versão da solução para apresentação. Finalmente, a solução revisada é apresentada à turma, possibilitando uma discussão coletiva onde todos conhecem a solução final e podem sugerir novas melhorias, promovendo um processo contínuo de aprimoramento cooperativo.

A APAP se constitui de uma AP complexa e dinâmica, na qual os estudantes são incentivados a refletir, interagir e cooperar. Isso possibilita o retorno às microarquiteturas anteriores, permitindo a interação entre elas e a transição entre atividades conforme necessário.

4. Experimento de Ensino

Foi realizado um experimento de ensino para validação da APAP. O estudo foi desenvolvido com estudantes do 2.º semestre do Curso Técnico Integrado em Desenvolvimento de Sistemas do Instituto Federal de Mato Grosso do Sul (IFMS) – Campus Coxim, mediante aprovação do Comitê de Ética em Pesquisa da Universidade Federal de Mato Grosso do Sul (CEP/UFMS), Parecer nº 6.842.168. A participação foi voluntária, mediante os termos de consentimento e assentimento aplicáveis, sendo garantidos o anonimato dos participantes e a utilização dos dados exclusivamente para fins científicos.

Inicialmente, foi realizado um convite às turmas do 2.º semestre para participarem voluntariamente do curso de complementação curricular on-line "Programação: Aprendendo a Resolver Problemas"**.** Nove estudantes se inscreveram, sendo oito do sexo masculino e uma do sexo feminino. Este experimento foi mediado integralmente pelas tecnologias. Para iniciarmos, foi disponibilizado um link do *Google Meet* por e-mail aos estudantes para o primeiro encontro síncrono, e todos os estudantes inscritos no curso compareceram, demonstrando um grande interesse. Durante esse encontro, apresentamos a estrutura e o funcionamento do curso.

Estabelecemos que teríamos dois encontros síncronos por semana, cada um com duração máxima de duas horas, e atividades assíncronas. Também expomos as principais ferramentas que utilizaríamos ao longo do curso, como o *Moodle* para a plataforma de aprendizado, o *Google Meet* para videoconferências, o *Google Docs* para compartilhamento de documentos, o *WhatsApp* para comunicação e o *Loom* para recursos visuais.

4.1 Desenvolvimento do Experimento

Durante seis encontros síncronos e atividades assíncronas, os estudantes participaram da microarquitetura pedagógica denominada “Compreensão do Problema e Repositório de Problemas Cooperativo (AP1)”. Essa abordagem incluiu a utilização de fóruns de discussão, repositórios cooperativos e vídeos de apoio que exploraram os conceitos de “Pensamento Computacional” e “Pensamento Crítico”, além da apresentação e elaboração de problemas [Silva et al. 2023].

Iniciamos a AP1 com os vídeos de apoio sobre “O que é Pensamento Computacional?” e “O que é Pensamento Crítico?”, seguido de um fórum sobre o que os estudantes entenderam de “Como resolver problemas usando o Pensamento Computacional?”. Em seguida, foram apresentados problemas, escolhidos pelo mediador, do teste Bebras Australia Computational Thinking Challenge, traduzidos para a língua portuguesa, dos anos de 2018, 2020 e 2022 [Silva et al. 2023].

A escolha dos problemas do Desafio Bebras se constitui em uma importante ferramenta, tendo em vista que os problemas foram selecionados de um evento internacional, utilizando habilidades do PC.

De imediato, foi solicitado aos estudantes que, individualmente, escolhessem por ordem de prioridade um dos seis problemas do teste Bebras Austrália Computational Thinking Challenge propostos pelo mediador/professor, conforme o link: https://shorter.me/BLZGI.

Cada estudante revisou, no mínimo, os problemas de dois colegas e, por sua vez, teve seus próprios problemas revisados por outros dois estudantes. Após essa etapa inicial de revisão, cada estudante refletiu individualmente sobre as revisões que fez e recebeu, considerando o feedback e analisando como poderia melhorar suas próprias soluções. Em seguida, os estudantes foram agrupados em pequenos grupos para revisar, no mínimo, os problemas de outros dois colegas e, por sua vez, teve seus próprios problemas revisados por outros dois colegas.

Por fim, cada membro do grupo explicou a escolha dos seus problemas. Esse processo de discussão permitiu que todos os estudantes explorassem diferentes perspectivas e abordagens para os problemas consolidando um problema para todos, e seguindo para a próxima AP, conforme Figura 5.

A partir do sétimo encontro até o décimo quarto encontro síncrono, os estudantes participaram da Arquitetura Pedagógica AP2. Essa abordagem incluiu a construção de soluções em formatos de textos e screencasts, a revisão cooperativa por pares das soluções e a apresentação das soluções para o grupo. Utilizando fóruns de discussão, revendo os repositórios cooperativos e vídeos de apoio que exploraram os conceitos de "Pensamento Computacional" e “Pensamento Crítico” [Silva et al. 2024].

Iniciou-se a AP2 com a atividade de construção da solução em linguagem natural, de modo individual, para o problema escolhido, juntamente com a explicação dessa solução no formato de vídeo, demonstrando como ela foi desenvolvida.

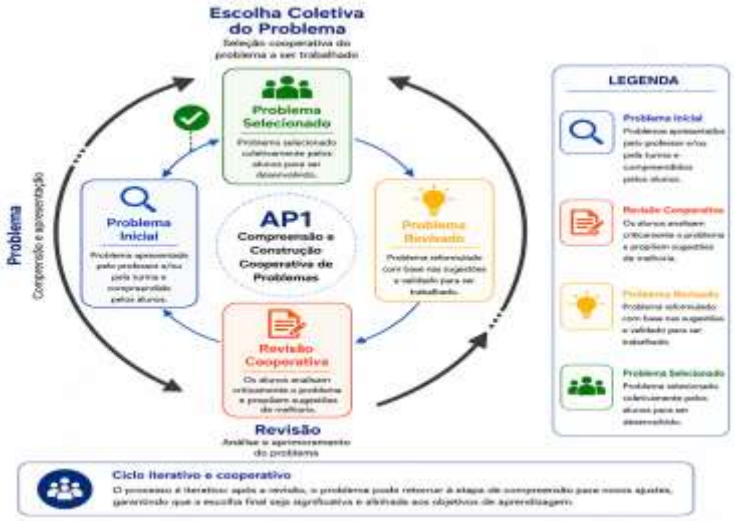

Figura 5: Ciclo de aprendizagem cooperativa AP1

Em seguida, o mediador realizou a distribuição de revisões entre os pares, levando em conta os diferentes estágios de desenvolvimento cognitivo dos participantes, para favorecer a interação entre revisores e revisados, com o intuito de ajudar no desenvolvimento cognitivo, conforme Figura 6.

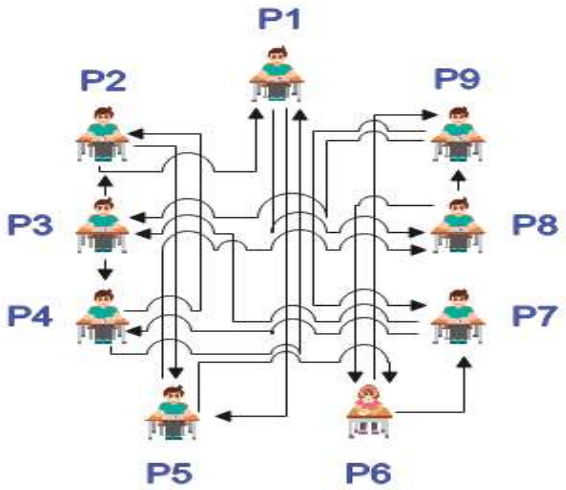

Figura 6: Distribuição de revisores, para 9 participantes e 2 revisões da AP2

Depois de refletir e discutir em pares, os estudantes foram agrupados em pequenos grupos para discutir suas soluções, e considerar o feedback recebido. Cada membro do grupo explicou suas soluções, defendeu suas escolhas e debateu as críticas e sugestões feitas pelos revisores.

Esse processo de discussão permitiu que todos pudessem experimentar diferentes perspectivas e abordagens para os problemas. Finalmente, trabalhando juntos consolidaram uma solução única e abrangente para o problema, combinando as melhores ideias e abordagens de todos [Silva et al. 2024].

Nos últimos sete encontros síncronos, os estudantes participaram da Arquitetura Pedagógica AP3, voltada à implementação computacional das soluções construídas anteriormente. Para isso, foram utilizados o *Google Colab* e a linguagem Python, escolhidos por serem ferramentas gratuitas e acessíveis. Inicialmente, cada estudante codificou individualmente a solução do problema, registrando dúvidas, dificuldades, erros e acertos por meio de fóruns, comentários no código e screencasts.

Posteriormente, os códigos e screencasts foram distribuídos entre os pares, seguindo a mesma estratégia de revisão utilizada na AP2. Essa etapa favoreceu a interação entre os estudantes na identificação e correção de erros sintáticos e semânticos, além da proposição de melhorias nas soluções desenvolvidas. As sugestões foram registradas diretamente nos códigos revisados, promovendo reflexão e reconstrução das soluções.

Ao analisar os códigos dos colegas, os estudantes tiveram contato com diferentes estratégias de resolução de problemas, confrontando suas próprias compreensões e ampliando seus conhecimentos. Nesse contexto, tanto os acertos quanto os erros contribuíram para a aprendizagem. Após receberem feedback dos pares, os participantes revisaram suas implementações, produziram novos screencasts com sugestões de melhoria e realizaram uma nova versão da codificação. Por fim, as soluções finais foram apresentadas à turma, permitindo novas discussões e sugestões, consolidando um processo de aprendizagem cooperativa, reflexiva e contínua.

4.2 Resultados e discussões do experimento de ensino

Este estudo analisou 40 horas de encontros síncronos, distribuídos em vinte e seis encontros ao longo de dois meses e meio, além de um repositório cooperativo de problemas, três fóruns de discussão, dezoito screencasts, documentos compartilhados e produções dos estudantes, incluindo soluções em linguagem natural, soluções em grupo e códigos com suas respectivas revisões.

Os resultados indicam que a tríade de Piaget (INTRA, INTER e TRANS) esteve presente durante o desenvolvimento das atividades. Na etapa INTRA, os estudantes construíram soluções individuais utilizando seus conhecimentos prévios e habilidades de pensamento computacional. Na etapa INTER, compartilharam soluções, receberam feedback e discutiram diferentes perspectivas durante os encontros síncronos e fóruns de discussão. Já na etapa TRANS, ampliaram suas compreensões ao participar das revisões por pares e da construção coletiva do conhecimento.

A utilização do repositório cooperativo de problemas e das discussões em grupo favoreceu a aprendizagem cooperativa, permitindo a troca de ideias e a exploração de diferentes abordagens para a resolução dos problemas. As revisões por pares desempenharam papel fundamental nesse processo, promovendo feedback construtivo, reflexão metacognitiva e aprimoramento contínuo das soluções propostas.

Na etapa de codificação, os estudantes desenvolveram inicialmente suas soluções de forma individual, e, posteriormente, compartilharam seus códigos para revisão cooperativa. As interações entre pares contribuíram para a identificação de erros, sugestões de melhorias e otimizações. Dificuldades relacionadas à sintaxe da linguagem de programação foram superadas por meio das discussões coletivas, do compartilhamento de códigos e do feedback oferecido pelos colegas e pelo professor.

Esse processo fortaleceu a colaboração, a reflexão e o desenvolvimento das habilidades de programação dos participantes.

Após as revisões por pares, o estudante "L" apresentou sua solução, com os comentários dos seus pares no código, incorporando as reflexões adquiridas ao revisar outros códigos e o feedback recebido, com base nesses achados, podemos inferir pontos importantes sobre as atividades práticas AP1, AP2 já publicadas e validadas, e apresentamos a AP3, que dará continuidade ao ciclo de aprendizagem cooperativa iniciado no experimento. E AP3, especialmente no contexto do desenvolvimento de habilidades de pensamento computacional e programação, ao identificar erros ou lacunas nas codificações dos colegas, os estudantes incentivam a reflexão, sobre as etapas do problema e os diferentes tipos de erros. Esse processo de ajuda mútua entre pares resulta em efeitos positivos nos processos de aprendizagem de cada estudante.

Durante a implementação da APAP, também foram observadas algumas dificuldades. Inicialmente, alguns estudantes apresentaram insegurança na realização das revisões por pares e na elaboração de feedbacks mais críticos e construtivos. Além disso, diferenças nos conhecimentos prévios de programação exigiram maior mediação do professor em determinadas atividades. Apesar dessas limitações, observou-se que a interação entre os estudantes contribuiu para reduzir essas dificuldades ao longo do experimento.

5. Considerações Finais

Este trabalho apresentou a Arquitetura Pedagógica baseada no Pensamento Computacional para apoio à Aprendizagem de Programação (APAP). Os resultados indicam que as práticas cooperativas e o Pensamento Computacional contribuíram para o desenvolvimento das habilidades de programação dos estudantes.

A AP1 favoreceu a compreensão e construção cooperativa de problemas por meio de um repositório compartilhado, estimulando reflexão e colaboração. A AP2 promoveu a construção e revisão cooperativa de soluções em linguagem natural, utilizando screencasts, revisão por pares e feedback contínuo. A AP3 destacou a importância da codificação prática e da revisão cooperativa de código, incentivando a identificação e correção de erros, a reflexão sobre estratégias de solução e a aprendizagem cooperativa.

Como limitação, este estudo foi conduzido com apenas nove estudantes voluntários, caracterizando um experimento exploratório. Como trabalhos futuros, pretende-se aplicar a APAP em turmas maiores e em diferentes instituições, além de investigar a integração da Inteligência Artificial como apoio à aprendizagem cooperativa e ao desenvolvimento das habilidades de Pensamento Computacional.

6. Declaração sobre uso de Inteligência Artificial

Os autores declaram o uso de ferramentas de inteligência artificial exclusivamente como apoio à revisão gramatical e ao aprimoramento da redação do manuscrito, bem como para melhorar a qualidade visual de algumas figuras e a apresentação de fórmulas matemáticas. O conteúdo científico, incluindo a concepção do estudo, a metodologia, o desenvolvimento, as análises, a interpretação dos resultados e as conclusões, foi integralmente elaborado e validado pelos autores.

Referências

Barr, V. and Stephenson, C. 2011. Bringing Computational Thinking to K-12: What Is Involved and What Is the Role of the Computer Science Education Community? ACM Inroads, 2(1), 48–54.

Carvalho, M. J. S., Nevado, R. A. and Menezes, C. S. 2005. Arquiteturas Pedagógicas para Educação a Distância: Concepções e Suporte Telemático. In Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE), 1(1), 351–360.Dagienė, V., Futschek, G., & Stupurienė, G. (2019). Creativity in solving short tasks for learning computational thinking. Constructivist Foundations, 14(3), 382-396.

Garcia, M. B. 2021. Cooperative Learning in Computer Programming: A Quasi-Experimental Evaluation of Jigsaw Teaching Strategy with Novice Programmers. Education and Information Technologies, 26(4), 4839–4856.

Giannakoulas, A. and Xinogalos, S. 2018. A Pilot Study on the Effectiveness and Acceptance of an Educational Game for Teaching Programming Concepts to Primary School Students. Education and Information Technologies, 23, 2029–2052.

López-Pernas, S., Gordillo, A., Barra, E. and Quemada, J. 2019. Examining the Use of an Educational Escape Room for Teaching Programming in a Higher Education Setting. IEEE Access, 7, 31723–31737.

Mazur, E. and Somers, M. D. 1999. Peer instruction: A user's manual.

Menezes, C. S., Castro Junior, A. D. and Aragón, R. 2021. Arquiteturas Pedagógicas para Aprendizagem em Rede. Série de Livros-Texto da CEIE-SBC. 1. ed. Porto Alegre: Editora da SBC, p. 1–27. Disponível em: https://ieducacao.ceie-br.org/arquiteturas-pedagogicas/. Acesso em: 5 jun. 2024.

Morales-Urrutia, E. K., Ocaña, J. M., Pérez-Marín, D. and Pizarro, C. 2021. Can Mindfulness Help Primary Education Students to Learn How to Program With an Emotional Learning Companion? IEEE Access, 9, 6642–6660.

Motil, J. and Epstein, D. 2000. JJ: A Language Designed for Beginners (Less Is More). Disponível em: https://www.ecs.csun.edu/~jmotil/TeachingWithJJ.pdf. Acesso em: 20 jul. 2026.

Piaget, J. 2013. A Psicologia da Inteligência. Editora Vozes, Petrópolis.

Piaget, J. 2016. L'Epistemologia Genetica. Edizioni Studium, Roma.

Piaget, J. and García, R. 1982. Psicogénesis e Historia de la Ciencia. Siglo XXI Editores, México.

Proulx, V. K. 2000. Programming Patterns and Design Patterns in the Introductory Computer Science Course. ACM SIGCSE Bulletin, 32(1), 80–84.

Román-González, M., Pérez-González, J. C. and Jiménez-Fernández, C. 2017. Which Cognitive Abilities Underlie Computational Thinking? Criterion Validity of the Computational Thinking Test. Computers in Human Behavior, 72, 678–691.

Silva, F. X., Menezes, C. S. and Castro Junior, A. N. 2023. Arquitetura Pedagógica Baseada no Pensamento Computacional para a Compreensão do Problema &

Portfólio de Aprendizagem. *Revista Novas Tecnologias na Educação (RENOTE)*, **21**(2).

Silva, F. X., Menezes, C. S. and Castro Junior, A. N. 2024. Arquitetura Pedagógica Baseada no Pensamento Computacional para a Resolução Cooperativa de Problemas. *Revista Novas Tecnologias na Educação (RENOTE)*, 22(1).

Selby, C. and Woollard, J. 2013. Computational Thinking: The Developing Definition. University of Southampton, Southampton.

Wing, J. M. 2006. Computational Thinking. Communications of the ACM, 49(3), 33–35.

Zha, S., Jin, Y., Moore, P. and Gaston, J. 2020. Hopscotch into Coding: Introducing Pre-Service Teachers Computational Thinking. TechTrends, 64(1), 17–28.