Skills-CT: Um Modelo para classificação dos Estágios Cognitivos das habilidades do Pensamento Computacional e desenvolvimento da Aprendizagem Criativa

Tatyane S. C. da Silva[1], Patricia C. de A. R. Tedesco (orientadora)[1], Jeane C. B. de Melo (coorientadora)[2]

[1]Centro de Informática –Universidade Federal de Pernambuco (UFPE), Recife,PE–Brasil

[2]Departamento de Computação–Universidade Federal Rural de Pernambuco (UFRPE), Recife,PE–Brasil

{tscs,pcart}@cin.ufpe.br,jeane.bmelo@ufrpe.br

Abstract. *This article presents the Skills-CT Model, relating the Pillars of Computational Thinking (CP) and Cognitive Stages in the problem-solving process and later applied in a Python Programming course. The Skills-CT Model aims to help the teacher to have a complete evaluation, of each one of the competencies of the student's Computational Thinking, subsidizing the teaching-learning process. The results of this study show that the classification of the Cognitive Stages of each CP skill was relevant for creating more specific activities based on the students' learning difficulties and weaknesses.*

Resumo. *Este artigo apresenta o Modelo Skills-CT, relacionando os Pilares do Pensamento Computacional (PC) e Estágios Cognitivos no processo de resolução de problemas e posteriormente aplicado em uma disciplina de Programação com Python. O intuito do Modelo Skills-CT é auxiliar o docente a ter uma avaliação mais completa, para cada uma das competências do Pensamento Computacional do educando, subsidiando o processo de ensino aprendizagem. Os resultados deste estudo mostram que a classificação dos Estágios Cognitivos de cada habilidade do PC foi relevante para a criação de atividades mais específicas com base nas dificuldades e fragilidades de aprendizagem dos alunos.*

1. Introdução

A resolução de problemas é uma habilidade metacognitiva, que indica a forma que uma pessoa aprende e vivencia diferentes aspectos no processo de solução de problemas. No contexto de programação, a resolução de problemas é uma habilidade fundamental. Contudo, muitos estudantes iniciantes de programação possuem dificuldade em formular um problema e expressar sua solução em código [Veerasamy *et al.* 2019].

Na literatura encontramos que, para muitos autores, o objetivo principal de uma disciplina de programação introdutória é desenvolver habilidades de solução de problemas por meio do Pensamento Computacional e dos fundamentos de programação, em vez de dar ênfase no ensino da sintaxe de uma linguagem de programação específica [Medeiros *et al.* 2018].

DOI: 10.5753/cbie_estendido.2022.226348

Recentemente, Lister [2020] propôs uma forma de classificação de habilidades de programação. Segundo Lister [2020], a teoria Neo-Piagetiana poderia ser usada para descrever o desenvolvimento cognitivo de programadores iniciantes em cada um dos estágios de desenvolvimento, a saber: Sensório-Motor, Pré-Operacional, Operacional Concreto e Operacional Formal. No estágio Operacional Formal, o aluno está apto a raciocinar de forma lógica, generalizar ou analisar outras possibilidades de utilização de um conceito aprendido.

Outro fator a ser considerado no processo de ensino-aprendizagem de programação é o Pensamento Criativo. Torna-se necessário engajar as pessoas de modo que possam desenvolver a criatividade, promovendo a integração do fazer e do ser, usando as suas competências com o intuito de transformar o mundo para melhor [Martinez e Tacca, 2009]. Nesse cenário, o Pensamento Computacional e a criatividade são vistos como ferramentas cognitivas que expandem o conhecimento e as habilidades que se pode aplicar na obtenção de uma solução para um dado problema.

Diante desse contexto, visando auxiliar na avaliação das habilidades do Pensamento Computacional de cada estudante e verificar seu nível cognitivo, foi desenvolvido o Modelo *Skills-CT* que relaciona os Pilares do Pensamento Computacional, classifica os Estágios Cognitivos para cada Habilidade do PC para a resolução de problemas em programação, estimulando a aprendizagem criativa dos estudantes.

Adicionalmente, o *Skills-CT* contempla a detecção do Pilar do PC no qual o discente apresenta dificuldade, influenciando diretamente na resolução de problemas algoritmicamente. O modelo foi avaliado em uma pesquisa-ação realizada nos módulos de Lógica de Programação e Python de um curso de formação profissional em Desenvolvimento de *Software*.

Com o objetivo de abordar o problema de aplicação dos conceitos de programação na resolução de problemas, lançando mão dos Pilares, das habilidades do Pensamento Computacional e dos Estágios Cognitivos da Teoria Neo-Piagetiana, tem-se, a seguir, uma questão que norteará esta pesquisa: "*Como a relação entre os Estágios Cognitivos Neo-Piagetianos e o Pensamento Computacional pode auxiliar no diagnóstico da aprendizagem do aluno de programação e estimular a aprendizagem criativa?*"

2. Fundamentação Teórica

As definições que embasam este estudo são apresentadas nesta seção. Os domínios cobertos incluem a definição e os Pilares do Pensamento Computacional e os Estágios Cognitivos da teoria Neo-Piagetiana. A relação destes aportes teóricos fundamentará o modelo desenvolvido e sua aplicação, os quais serão apresentados na seção 4.

2.1 Pensamento Computacional

O Pensamento Computacional pode ser definido como uma forma de analisar e resolver problemas, de realizar o *design* de sistemas, bem como compreender o comportamento humano, com base em conceitos fundamentais da Computação [Wing 2006].

O Pensamento Computacional possui quatro pilares que norteiam o processo de resolução de problemas. O primeiro pilar é a Decomposição, sendo definida como a

divisão de um problema complexo em partes menores e mais simples de resolver, aumentando a atenção aos detalhes. O segundo pilar é o Reconhecimento de Padrões, caracterizado pela identificação de semelhanças em diferentes processos para resolvê-los de forma mais eficiente e rápida. A Abstração é o terceiro Pilar, onde se busca isolar aspectos importantes de um todo e considerá-los individualmente. Por fim, o último Pilar são os Algoritmos que compreende o processo de criação de um conjunto de regras para resolver o problema [BBC 2017]. Nesta pesquisa utilizaremos os quatro pilares do Pensamento Computacional proposto pela BBC [2017] para resolução de problemas.

2.2 Teoria Neo-Piagetiana

Segundo Lister [2020], baseando-se na teoria Neo-Piagetiana, existem quatro estágios do desenvolvimento cognitivo do programador iniciante. Na pesquisa aqui apresentada, trabalhamos com as três primeiras etapas.

O primeiro estágio Neo-Piagetiano é denominado Sensório-Motor. Nesse estágio, os iniciantes em programação têm pouca capacidade de produzir resultados consistentes de maneira confiável por meio do rastreamento de código. Eles ainda têm dificuldade em diferenciar entre palavras reservadas, variáveis, listas e sintaxe em geral.

No estágio Pré-Operacional , os programadores iniciantes conseguem rastrear o código de forma confiável, mas eles não abstraem rotineiramente o código para ver um cálculo significativo que este executa por esse código. Eles têm dificuldade em raciocinar sobre como analisar a primeira e a última iterações e ignoram as iterações intermediárias em um *loop*.

Os programadores iniciantes que se encontram no estágio Operacional Concreto, por sua vez, podem raciocinar sobre abstrações do seu código. Neste estágio, os programadores podem relacionar as partes do código e entender como todos eles se inter-relacionam para formar o quadro geral. Eles não se concentram apenas nos estados inicial e final dos *loops*, podendo raciocinar sobre a mudança de variáveis nas iterações de comandos de repetição.

3. Trabalhos Relacionados

O trabalho de Araújo *et al.* [2020] propõe armazenar, de forma estruturada, o percurso dos estudantes durante o desenvolvimento dos projetos no *Scratch*, objetivando investigar o processo de aprendizagem das habilidades do PC. Ao final do estudo, Araújo *et al.* [2020], indicam que tal proposta seria viável para representação dessas habilidades, tendo resultados equivalentes a outras ferramentas.

O estudo de Troiano *et al.* [2019] objetivou investigar até que ponto as métricas do PC podem ser usadas para analisar o desenvolvimento dos estudantes, além de fornecer *insights* sobre como os alunos desenvolvem suas habilidades ao longo dos projetos. A análise verificou quais dimensões se desenvolvem mais cedo e com mais proficiência, e quais se apresentam como mais difíceis de desenvolver.

Alguns pontos em comum nos artigos sobre classificação dos Estágios Cognitivos e análise de código foram verificados, tais como: a falta de clareza na relação entre a resolução de problemas e os pilares do Pensamento Computacional, não identificando em que nível de desenvolvimento do PC os estudantes se encontram e

mapear ou classificar quais as habilidades do Pensamento Computacional os estudos auxiliam no desenvolvimento.

4. Modelo Conceitual Skills-CT

Para a criação do modelo *Skills-CT* proposto neste trabalho, foram estudadas pesquisas relacionadas aos campos de conhecimento abordados nas seções anteriores. A partir dos resultados da revisão, o modelo denominado *Skills-CT* foi criado e, posteriormente, aplicado através de pesquisa-ação, ou seja, um processo no qual os participantes examinam sua própria prática docente de forma sistemática e cuidadosa, usando as técnicas de pesquisa [Ferrance 2000].

O Modelo *Skills-CT* objetiva auxiliar o docente a avaliar pilares e habilidades de Pensamento Computacional e Estágios Cognitivos de forma personalizada, estabelecendo suas classificações. A partir dos resultados, a ideia é propor atividades e preparar aulas de acordo com os níveis e habilidades identificados, visando diminuir a dificuldade de aplicar os conhecimentos na Resolução de Problemas, além de promover o incremento do Estágio Cognitivo do estudante. O Modelo *Skills-CT* (Figura 1) propõe três relações: a dos Pilares do PC com as habilidades, os Estágios Cognitivos com os Níveis das Habilidades do PC e dos Pilares do Pensamento Computacional com o CPS (*Creative Problem Solving*).

A relação entre o Pilar Decomposição (Figura 1) e as habilidades de Abstração e Decomposição de Problemas é estabelecida tendo em vista que, para essa habilidade, deve-se dividir a programação em funções com ou sem retorno e utilização de funções específicas da linguagem de programação.

O Reconhecimento de Padrões, por sua vez, está relacionado ao Paralelismo. Essa relação ocorre no contexto de uma linguagem de programação quando uma função é passada como parâmetro e/ou quando realiza o chamado de outra função (Figura 1).

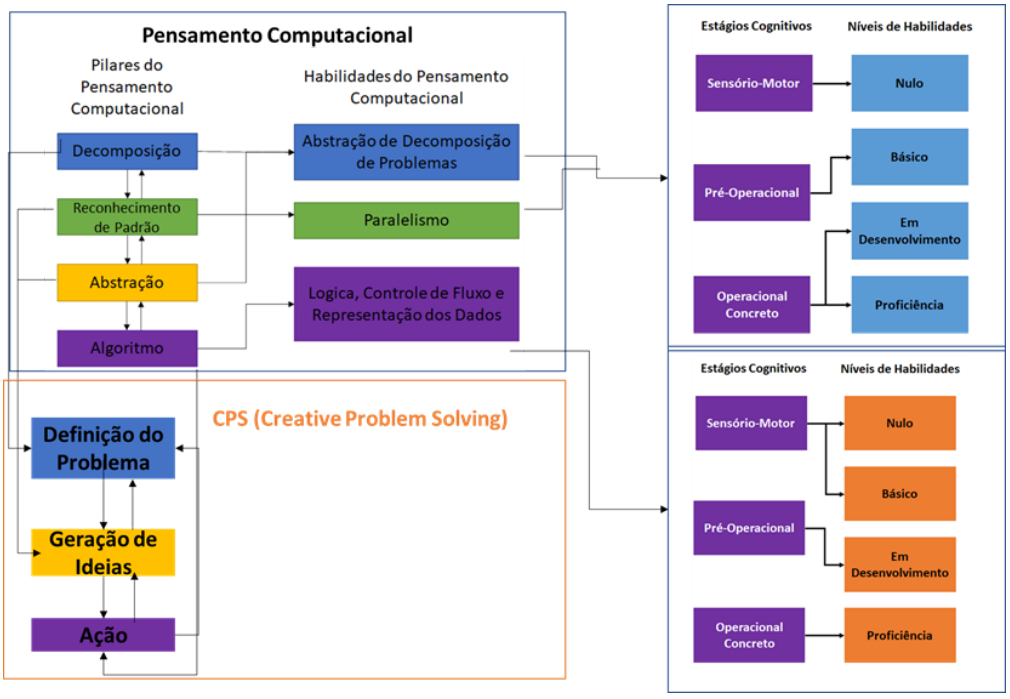

Figura 1. Modelo Conceitual SKills-CT. Fonte: Autora.

O último dos pilares, Algoritmo, está relacionado à formalização da solução. Portanto, as habilidades do Pensamento Computacional que estão vinculadas a este Pilar são: Lógica, Controle de Fluxo e Representação de Dados. Tendo em vista sua relação direta ao desenvolvimento de algoritmos que envolvem o uso de condicionais, *loops*, variáveis, *arrays* e listas.

A relação dos Estágios Cognitivos com os níveis das Habilidades do Pensamento Computacional podem ser observadas nos Quadros 1 e 2. Para este trabalho foram consideradas apenas as habilidades apresentadas no Quadro 1 para o uso do *Scratch* e no Quadro 2 para uso de linguagem de programação procedural. Esta escolha ocorreu por estarem relacionadas aos pilares do Pensamento Computacional.

Quadro 1. Habilidades do Pensamento Computacional para o Scratch. Fonte: Moreno-León, Robles e Román-González (2015)

Conceito de Pensamento Computacional	**Níveis de Habilidades**			
	(0) Nulo	(1) Básico	(2) Em desenvolvimento	(3) Proficiência
Abstração e Decomposição de Problemas	-	Mais de um *script* e mais de um *sprite*	Definição de blocos	Uso de clones
Paralelismo	-	Dois *scripts* na bandeira verde	Dois *scripts* na tecla pressionada, dois scripts no sprite e ação de clique no mesmo *sprite*	Dois *scripts* quando recebo a mensagem, criar clone, dois *scripts* quando o cenário muda
Lógica	-	Se (*if*)	Se, senão (*if else*)	Operações Lógicas
Controle de Fluxo	-	Sequência de blocos	Repetir, Sempre	Repetir até
Representação de Dados	-	Modificadores de propriedades	Operações sobre variáveis	Operações em lista

Quadro 2. Habilidades do Pensamento Computacional para linguagem de programação procedural. Fonte: Autores.

Conceito de Pensamento Computacional	**Níveis de Habilidades**			
	(0) Nulo	(1) Básico	(2) Em desenvolvimento	(3) Proficiência
Abstração e Decomposição de Problemas	Não possui função	Função sem retorno	Função com retorno	Uso de um recurso da linguagem de programação
Paralelismo	Não possui função	Uma função chamando outra	Usar retorno de uma função como parâmetro de outra função	Usa de um recurso da linguagem de programação
Lógica	Não possui *if*	*if*	*if else*	Operações lógicas
Controle de Fluxo	Não possui estrutura de repetição	Não possui estrutura de repetição	*For*	*While*

Representação de Dados	-	Não possui variáveis	Operações sobre variáveis	Operações em lista ou array

Adicionalmente, o Modelo *Skills-CT* apresenta a relação entre os pilares do Pensamento Computacional e CPS (*Creative Problem Solving*) que destina-se a auxiliar na solução de problemas para facilitar o aprendizado da programação de forma criativa [Silva *et al.* 2020].

O CPS foi criado por Osborn [2008] e consiste em um paradigma metodológico composto por métodos e técnicas para analisar, identificar e solucionar problemas [Osborn 2008]. Este método é formado por três etapas: a Investigação, a Descoberta de Ideias e a Descoberta de Soluções [Osborn 2008].

A fase de Decomposição que compõem os pilares do PC está relacionada à fase de definição do problema CPS (Figura 1), pois esta técnica ajuda a dividir o problema e observá-lo de diferentes perspectivas. O estágio de Reconhecimento de Padrões visa reconhecer os padrões para criar algo inovador e diz respeito à fase de geração do CPS, que é a etapa selecionada para essa função. A fase de Abstração está relacionada à fase de geração de ideias, uma vez que visa treinar no indivíduo a habilidade de observar conceitos apenas de forma genérica na busca pelos mais relevantes. Por fim, o pilar Algoritmo está relacionado ao código ou qualquer representação algorítmica da solução e está localizado na fase Ação do modelo CPS, por ser a fase de desenvolvimento da solução [Silva *et al.* 2020].

4.1. O Estudo

O estudo foi realizado com uma turma de onze estudantes de um curso profissional *on-line* de Desenvolvimento de *Software*. Destes estudantes, sete eram do gênero masculino (A1, A2, A3, A5, A8, A10, A11) e quatro do gênero feminino (A4, A6, A7, A9). Apenas cinco alunos (A1, A2, A5, A6, A11) tinham conhecimento prévio de introdução à programação. A pesquisadora deste estudo foi a professora da turma.

Os módulos utilizados neste estudo foram o de Lógica de Programação e o de Python. As aulas do curso ocorreram duas vezes por semana, com duração de duas horas cada. A ferramenta usada nas aulas on-line foram o *Meet*, o *Classroom* para envio de material e atividades e o *GitHub* onde os alunos submetiam seus códigos.

Para aplicação do modelo, o Guia de Ensino de Programação (GEP) e suas etapas foi usado para planejar as aulas, as ferramentas utilizadas para classificar as habilidades do PC, as linguagens de programação visual e de script dos módulos de Lógica de Programação e *Python*.

As etapas do GEP são: 1. Linguagens de Programação: o docente pode escolher 3 linguagens de programação 1 visual, uma do idioma e de sintaxe simples. Contudo, fica a critério do docente seguir os passos passando pelas três linguagens ou ir diretamente para uma linguagem de *script*; 2. Ferramentas: Escolher ferramentas para analisar os códigos e os estágios cognitivos; 3. Estágios Cognitivos: elaborar exercícios, utilizando a relação dos pilares do PC com o CPS, para avaliar os estágios cognitivos dos estudantes; 3. Métodos visual para gerenciar tarefas: usar quadro para acompanhar o processo de resolução de problemas dos estudantes e preparar aulas semanais; 4. Questionário de Autoeficácia: elaborar e aplicar questionário para autoavaliação do

estudante; 5. Avaliação: realizar avaliação dos estágios cognitivos, do quadro de tarefas e questionário de autoeficácia.

A Jornada de Aprendizagem de Programação (JAP) desenvolvida com objetivo de criar uma experiência de aprendizagem focada na resolução criativa de problemas, análise da autoavaliação realizada pelos alunos e, finalmente, o questionário de autoeficácia.

4.2. Resultados

A análise foi realizada com base nos códigos dos alunos, nos Estágios Cognitivos e nos Pilares do PC. Para análise utilizou-se o Dr. *Scratch* e o classificador desenvolvido pela autora para avaliar os códigos em Portugol e *Python*. As Figuras 2 e 3 apresentam os resultados das avaliações das soluções dos alunos em *Scratch,* Portugol e Python. A análise foi realizada com 11 alunos e podem ser observadas com mais detalhes na Tese.

O Estudante 1 (Figura 2A) tem um Estágio Cognitivo em desenvolvimento em quase todas as habilidades de Pensamento Computaciona (*Scratch*), apresentando mais dificuldade em habilidades que requerem o uso de *loop* ou o uso de variáveis em *loop*. O Aluno 1 (Figura 2B) tem o Estágio Cognitivo Sensório-Motor (Portugol*)* presente em três habilidades do PC: Lógica, Paralelismo e Abstração e Decomposição de Problemas. Com isto, pode-se inferir que o estudante tem dificuldade nos três primeiros Pilares do PC, que estão mais relacionados em preparar a resolução de problemas.

Figura 2.Estágios Cognitivos de habilidades do PC – Aluno 1.

O Aluno 1 (Figura 2C) tem um Estágio Cognitivo Operacional Concreto (*Python)* presente em quatro habilidades de Pensamento Computacional, com maior ênfase na Representação de Dados, que possui em sua totalidade o nível Operacional Concreto.

O Aluno 11 (Figura 3A) tem um Estágio Cognitivo Sensório-Motor (*Scratch*) mais forte no Paralelismo, mostrando que tem dificuldade em aplicar conhecimentos para criar um código que possa ser executado ao mesmo tempo. O Aluno 11 (Figura 3B) possui os Estágios Cognitivos Sensório-Motor (Portugol) mais acentuados em

Lógica, Paralelismo e Abstração, indicando que se encontra no nível iniciante. Nas habilidades de Controle de Fluxo, está em desenvolvimento, uma vez que foi classificado no Estágio Cognitivo Pré-Operacional. O Aluno 11 (Figura 3C) apresenta nível de habilidades do PC em Proficiência, mas em Lógica ainda está no Sensório-Motor, revelando que não usa operadores lógicos em suas soluções (*Python*).

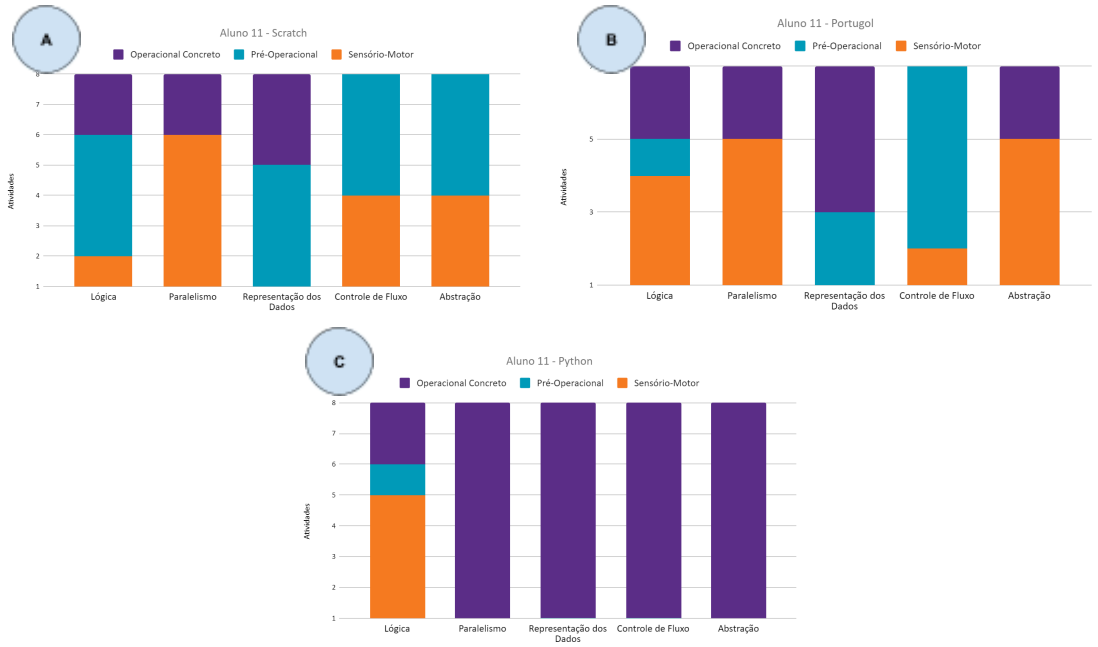

Figura 3.Estágios Cognitivos de habilidades do PC – Aluno 11.

4.3 Discussão

Sobre a classificação dos Estágios Cognitivos no *Scratch*, a pesquisa mostrou que mesmo estudantes com experiência anterior em programação possuem Estágios Cognitivos ainda não desenvolvidos em diversas habilidades de PC. Isto pode ser uma indicação de que o ensino de habilidades de Pensamento Computacional deve ser priorizado em relação ao ensino focado na sintaxe de uma linguagem de programação.

Em relação aos pilares do PC, pode-se inferir que os estudantes apresentaram o Estágio Cognitivo Sensório-Motor com maior frequência nas habilidades de PC e mais dificuldade nos primeiros Pilares do PC.

As habilidades do PC que apresentaram os níveis Sensório-Motor e Pré-Operacional com maior frequência em todas as linguagens foram o Paralelismo e a Abstração da Decomposição do Problema. Embora o estudo não tenha como objetivo comparar os alunos entre si, mas, sim, um aluno com sua evolução, esse foi um fato que se repetiu em quase todas as análises dos discentes. Depreende-se que a razão é porque o estudante do Sensório-Motor tem dificuldade em acompanhar a execução do código e o Pré-Operacional ainda tem que rastrear uma parte do código linha a linha e não por execução, bem como ainda tem dificuldade em entender a recursão.

No que concerne aos Estágios Cognitivos apresentados no módulo de Python, as quais os problemas foram elaborados e resolvidos pelos alunos, percebeu-se o Estágio Cognitivo Operário Concreto mais presente. Entende-se que isto é resultado da aprendizagem criativa, uma vez que o discente usou os processos de personalização das

informações, de comparação com os assuntos de programação aprendidos a partir de interpretação e pontos de vistas próprios e de geração de novas ideias ao criar suas próprias questões que resolviam um problema do seu contexto social e das suas vivências.

Em relação ao questionário de autoeficácia, na maioria das questões os alunos responderam que estão confiantes para realizar as atividades. Assim, denota-se que o Scratch tem o maior índice de confiança e que é uma ferramenta fácil e intuitiva de usar.

5. Considerações Finais

O estudo aqui apresentado foi discorrido a partir da seguinte pergunta: como a relação entre os Estágios Cognitivos Neo-Piagetianos e o Pensamento Computacional pode auxiliar no diagnóstico da aprendizagem do aluno de programação e estimular a aprendizagem criativa? Para tanto, foi proposto um Modelo *Skills-CT* que visa avaliar as habilidades de Pensamento Computacional e os Estágios Cognitivos dos alunos em cada uma das habilidades de PC para auxiliar o docente em uma avaliação mais profunda dos desenvolvimentos de tais habilidades. Adicionalmente, *o Skills-CT* ajuda no desenvolvimento da aprendizagem criativa por meio da criação das atividades que possuem de forma implícita o CPS.

Considerando a perspectiva do desenvolvimento do Pensamento Computacional através da Programação, Resolução de Problemas e Aprendizagem Criativa, a pesquisa contribuiu para avaliar e propor atividades para auxiliar os estudantes a galgar os Estágios Cognitivos no desenvolvimento das habilidades do PC.

A seguir são listadas as principais contribuições do trabalho desenvolvido: (i) criação, utilização e avaliação do Modelo *Skills-CT*; (ii) Definição e utilização do Guia para o Ensino de Programação (GEP); (iii) Promover a Aprendizagem Criativa por meio do Guia para o Ensino de Programação; (iv) Agilizar o acompanhamento, por parte do docente, do desenvolvimento das habilidades do Pensamento Computacional, bem como os Estágios Cognitivos dessas habilidades.

5.1 Limitações

As limitações desta pesquisa serão apresentadas a seguir: (i) A pesquisa utilizou o Modelo *Skills-CT*, contudo é necessária uma validação deste Modelo por meio de sua utilização em outros estudos e turmas; (ii) A atividade de elaboração de questões por parte dos alunos foi aplicada apenas no módulo de *Python* e impactou positivamente nos resultados, mas não foi aplicada para as outras linguagens, *Scratch* e Portugol, não permitindo, por conseguinte, verificar seu impacto nos Estágios Cognitivos dos alunos; (iii) Pesquisa realizada em um pequeno grupo de alunos.

5.2 Trabalhos Futuros

A presente pesquisa alcançou os objetivos traçados inicialmente e apresentou êxito ao usar o *Modelo Skills-CT* e o Guia para o Ensino de Programação. Os resultados obtidos conduziram a possíveis trabalhos futuros, entre eles: (i) Validar o Modelo *Skills-CT*; (ii) Replicar o estudo em outras turmas de disciplinas de Introdução à Programação e usar a análise dos códigos ao longo das soluções e não apenas no final do projeto ou atividade.

6. Agradecimentos

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro para a realização desta pesquisa.

Referências

Araújo, N. de; Primo, T. T.; Pernas, A. M. OntoScratch: ontologias para a avaliação do ensino de Pensamento Computacional através do Scratch. In: Anais do XXXI Simpósio Brasileiro de Informática na Educação. SBC, 2020. p. 1823-1832.

BBC, C. T.(2017) Introduction to computational thinking. Retrieved June 2, 2021, from BBC website: https://www.bbc.com/education/guides/zp92mp3/revision/1.

Ferrance, E. Action research. Providence, RI: LAB, Northeast and Island Regional Education Laboratory at Brown University, 2000.

Lister, R. On the cognitive development of the novice programmer: and the development of a computing education researcher. In: Proceedings of the 9th Computer Science Education Research Conference (CSERC '20). Association for Computing Machinery, New York, NY, USA, Article 2, p. 1-15, 2020.

Martinez, A. M.; Tacca, M. C. V. R. A complexidade da aprendizagem. Campinas: Editora Alínea, 2009.

Medeiros, R. P.; Ramalho, G. L.; Falcão, T. P. A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. IEEE TRANSACTIONS ON EDUCATION, p. 14, 2018.

Moreno-León, J.; Robles, G. (2015) Dr. Scratch: A web tool to automatically evaluate Scratch projects. In: Proceedings of the workshop in primary and secondary computing education.. p. 132-133.

Osborn, A. Your Creative Power. London and New York: Read Books, 2008.

Silva, T. S. C.; Melo, J. C. B.; Tedesco, P. C. A. R. Creative Learning in Problem Solving and Development of Computational Thinking. In: International Conference on Computer Supported Education, Springer, Cham, p. 199-215, 2020.

Troiano, G. M., Snodgrass, S., Argımak, E., Robles, G., Smith, G., Cassidy, M., ... & Harteveld, C. (2019). Is my game OK Dr. Scratch? Exploring programming and computational thinking development via metrics in student-designed serious games for STEM. In Proceedings of the 18th ACM international conference on interaction design and children (pp. 208-219).

Veerasamy, Ashok Kumar et al. Relationship between perceived problem‐solving skills and academic performance of novice learners in introductory programming courses. Journal of Computer Assisted Learning, v. 35, n. 2, p. 246-255, 2019.

Wing, J. M. Computational thinking. Commun. ACM, The Royal Society, v. 49, n. 3, p. 33 – 35, 10 (2006). ISSN 1364-503X. Retrieved June 2, 2021, from ACM website: http://doi.acm.org/10.1145/1118178.1118215.