Comparison between Pseudocode Usage and Visual Programming with Scratch in Programming Teaching

Críscilla M. C. Rezende
Instituto de Ciências Exatas e Tecnológicas (ICET)
Universidade Federal de Jataí (UFJ)
Jataí-Goiás, Brasil
cris0511.m@gmail.com

Esdras L. Bispo Jr.
Instituto de Ciências Exatas e Tecnológicas (ICET)
Universidade Federal de Jataí (UFJ)
Jataí-Goiás, Brasil
bispojr@ufg.br

***Abstract*— This work presents the results of a research that aimed to evaluate the use of the Scratch visual programming language in the development of computational thinking, in comparison with the use of pseudocode, during the teaching of logic and programming algorithms. The research was delineated with a methodology of action research, which made possible the evaluation of the Scratch language and the pseudocode at the end of the approach of each content. The steps of approaching each content were initiated with the application of the diagnostic evaluation, and finalized with the evaluation of performance. The results indicate that the use of the Scratch language presented better results during the initial stages; as the complexity of the content increased, the use of the pseudocode produced better results. Nevertheless, it is worth noting the good acceptance of the Scratch language for the teaching-learning process of the contents, as well as the contribution that it has had to the learning of logic and programming algorithms.**

Keywords*— *computing education, Computational Thinking, programming logic, Scratch

I. INTRODUÇÃO

O desenvolvimento de habilidades cognitivas é fundamental para a existência do ser humano, uma vez que essas habilidades são as responsáveis pela capacidade de interação com o ambiente, e também pela solução de problemas no cotidiano. Desenvolver essas habilidades está condicionado às situações as quais o indivíduo é exposto, bem como às experiências que ele vive, tendo ou não como cenário um espaço escolar.

Além das habilidades básicas com a matemática, leitura e escrita, compreensão de fenômenos, entre outras, uma habilidade que tem se mostrado fundamental para todos é o pensamento computacional [1], que está relacionado ao saber usar o computador para aumentar a produtividade, inventividade e criatividade [2], e não apenas como ferramenta para solucionar problemas mais rapidamente.

De acordo com Bundy, o "pensamento computacional fornece uma nova linguagem para descrever hipóteses e teoria" [3]; e segundo Wing, "inclui uma série de ferramentas mentais que refletem a vastidão do campo da ciência da computação" [1]. Nesse sentido, o pensamento computacional auxilia na resolução de problemas, à medida que o processo de resolução executado pelo computador é compreendido. Uma vez que o indivíduo compreende esse processo, isso pode contribuir para a escolha da melhor solução, delineada a partir de uma forma de abstrair e modelar o problema, consoante aos processos computacionais. Essa compreensão está relacionada à habilidade de pensar computacionalmente, o que envolve compreender os conceitos de lógica e algoritmos de programação.

Segundo Resnick *et al*, "a programação apoia o pensamento computacional, ajudando a aprender importantes estratégias de resolução de problemas e design" [4]. Entretanto, as dificuldades relacionadas ao ensino de conceitos de programação não são tarefas fáceis de serem solucionadas. O ensino desses conceitos, para alunos que não estão em cursos na área de computação, requer o uso de tecnologias e metodologias de ensino-aprendizagem adequadas, visto que essas, podem auxiliar na superação das dificuldades inerentes ao ensino de programação [5].

O uso de ferramentas que minimizam os detalhes das linguagens de programação, facilitam o aprendizado; com foco nos algoritmos, o aluno pode expressar a sua criatividade em busca da solução do problema, priorizando o desenvolvimento da lógica, em vez de se preocupar com regras de sintaxe da linguagem de programação. Exemplo dessas ferramentas é a linguagem de programação visual, haja vista ao fato desse tipo de linguagem provê uma maior abstração do processo de programação, utilizá-la pode auxiliar no processo de ensino-aprendizagem dos conceitos básicos da programação, e contribuir facilitando a aprendizagem do aluno enquanto o auxilia na resolução de problemas [6].

A linguagem Scratch é um tipo de linguagem de programação visual, cujo código é escrito através de blocos; o usuário organiza os blocos de acordo com a sequência lógica que ele acredita ser o ideal para escrever a sua estória. Dessa forma, o usuário desenvolve a lógica computacional do seu próprio programa, sem se preocupar com detalhes da sintaxe de uma linguagem de programação.

Adotar essa linguagem para auxiliar no processo de ensino-aprendizagem da lógica de programação, e analisar o resultado proveniente dessa utilização, pode trazer uma avaliação mais ampla sobre as abordagens tradicionais para o ensino de programação quando comparadas com o uso de tecnologias didáticas diferenciadas, nesse caso, a linguagem de programação visual.

Diante do exposto, esse trabalho apresenta os resultados de uma pesquisa que objetivou avaliar a contribuição da linguagem de programação visual Scratch, em comparação ao uso do pseudocódigo, no processo de ensino-aprendizagem de lógica e algoritmos de programação. O cenário de comparação foi o ensino de programação, para alunos do ensino superior do curso de Licenciatura em Física.

Este artigo está organizado como se segue. A Seção II apresenta o referencial teórico acerca dos conceitos abordados no trabalho. A Seção III descreve a metodologia utilizada para o desenvolvimento da pesquisa. A Seção IV discute os resultados obtidos com a realização da pesquisa. E, por fim, na Seção V são feitas as considerações finais sobre a pesquisa.

II. Referencial Teórico

A. *Pensamento Computacional*

A disseminação do uso de computadores em todas as áreas da sociedade mostra que, desde o princípio da computação relacionada à informática, a utilização de computadores traz benefícios consideráveis, atuando como uma extensão de nossas habilidades cognitivas [3]. Entretanto, saber utilizar o computador apenas como uma máquina capaz de processar informações automaticamente, não é suficiente para se aproveitar todo o benefício proveniente da computação.

A computação favorece o desenvolvimento de soluções de problemas que nem sempre estão relacionados ao computador. As habilidades vinculadas à essas soluções, por sua vez, são desenvolvidas a partir da compreensão do processo de resolução de problemas efetuado pelo computador. À essas habilidades atribui-se o nome de pensamento computacional, que de acordo com Wing, "baseia-se no poder e limites de processos computacionais, sejam eles executados por um humano ou por uma máquina" [1]. Ainda segundo o autor, o pensamento computacional envolve tanto a resolução de problemas, quanto a projeção de sistemas, e também a compreensão do comportamento humano. Este envolvimento se dá pela extração de conceitos fundamentais da ciência da computação.

O pensamento computacional é citado como uma habilidade importante para todos [1] [2] [3]; pode ser empregado para solucionar problemas em diferentes âmbitos, e influencia pesquisas em diversas áreas [3], sendo fundamental não apenas na ciência da computação [1].

A habilidade do pensamento computacional é considerada como necessária para o pleno exercício de cidadania no século XXI [2]. Consoante a isso, Bundy afirma que "se você quer entender o século XXI, então você deve primeiro entender a computação" [3].

O desenvolvimento do pensamento computacional é favorecido pela utilização de computadores, em especial, por ferramentas computacionais que aproximem à compreensão humana, a forma como o computador executa a solução de problemas. Ferramentas que são capazes de propiciar uma alta abstração dos detalhes da codificação de programas, tendem a ser mais atrativas, pois estas minimizam o desgaste com o aprendizado em relação a informações minuciosas, como a sintaxe de uma linguagem de programação. De acordo com França e Tedesco, merecem destaques as ferramentas Scratch, App Inventor, AgentSheets e RoboMind [7], todos voltados para a programação visual.

B. *Scratch*

O Scratch é uma linguagem de programação visual, desenvolvida e mantida pelo grupo Lifelong Kindergarten do Instituto de Tecnologia de Massachusetts (MIT). Criada em 2007, foi inicialmente idealizada para atender um público com idade entre 8 e 16 anos. Porém, é utilizada atualmente por pessoas em qualquer faixa etária.

Conforme informações disponibilizadas na página oficial da linguagem [1], esta ferramenta auxilia no pensamento criativo, raciocínio sistematizado e trabalho colaborativo; estas características fazem parte da descrição do pensamento computacional [2].

Baseada na linguagem Logo, e fundamentada nos conceitos construcionistas dessa linguagem, Scratch permite ao usuário desenvolver habilidades como o pensamento criativo, análise crítica, e aprendizagem contínua.

A intenção da linguagem Scratch é oferecer ao aluno um ambiente em que ele mesmo possa programar sua estória [8]. Isso exige o desenvolvimento de sua criatividade no processo de resolução de problemas. Esse desenvolvimento está aliado à lógica de programação de uma maneira mais intuitiva e agradável, pois esta linguagem propicia o encapsulamento de detalhes sobre a sintaxe de linguagens de programação comuns.

A programação com o Scratch é feita em blocos. O usuário organiza estes blocos de acordo com o seu raciocínio, a fim de prover uma sequência lógica na estória.

A linguagem Scratch é disponibilizada gratuitamente, e pode ser utilizada como ferramenta pedagógica em vários níveis da educação [9]. Possui uma vasta documentação de apoio e também mantém comunidades de educadores como a ScratchEd, desenvolvida pela Escola de Graduação de Educação de Havard. O objetivo do ScratchEd é promover um compartilhamento colaborativo das histórias desenvolvidas, no intuito de difundir novas possibilidades de interação com a ferramenta.

De acordo com Marcelino *et al*, a linguagem Scratch é um dos produtos mais utilizados para promover o desenvolvimento do pensamento computacional [10]. O uso dessa linguagem, para o ensino de programação, é empregado em diferentes níveis da Educação, apresentando resultados satisfatórios quanto ao desenvolvimento de habilidades pelos alunos [10] [11] [12] [13]. Esses resultados corroboram para mostrar a contribuição que essa linguagem pode oferecer no processo de ensino-aprendizagem de lógica e algoritmos de programação.

III. Metodologia

Essa seção descreve os procedimentos técnicos adotados para a pesquisa, recursos tecnológicos e ambiente de aula, bem como a população participante.

A. *Pesquisa-Ação*

A pesquisa-ação, na educação, é um tipo de pesquisa que tem por objetivo estabelecer mudanças imediatas em uma

[1] https://scratch.mit.edu

configuração educacional [14]. Ela tem um poder imenso de gerar rápidas mudanças, pois a pesquisa é conduzida pelos educadores em seus próprios espaços de trabalho. Assim, a pesquisa-ação fornece condições tanto para o aprimoramento das competências do docente, quanto para uma formulação científica das realidades por ele vividas.

Existem diferentes tipos de pesquisa-ação [15]. Podemos citar três tipos a seguir. O primeiro, a pesquisa-ação técnica, é guiado pelo interesse em melhorar o controle sobre os resultados. O segundo, a pesquisa-ação crítica, é guiado pelo interesse de emancipar pessoas e grupos da irracionalidade, injustiça ou sofrimento. E, por último, a pesquisa-ação prática, adotada neste trabalho, é guiado pelo interesse em educar ou esclarecer profissionais para que ajam mais sábia e prudentemente.

A pesquisa-ação é de natureza cíclica e bastante flexível, sendo, às vezes, difícil descrevê-la como uma sequência de passos a seguir [16]. Entretanto, com o intuito de estruturar mais formalmente as etapas da pesquisa, alguns passos podem ser adotados. Estes passos podem facilitar as atividades do pesquisador-docente, permitindo que este atenda satisfatoriamente aos princípios da metodologia.

Os passos adotados neste trabalho, para a execução da pesquisa-ação, são os sugeridos por McKay e Marshall [17], conforme apresentado na Fig. 1.

Fig.1. Ciclo de desenvolvimento da pesquisa-ação. Fonte: Adaptado de McKay e Marshall [17].

A pesquisa é estruturada em 1 ciclo com 8 passos. No passo 1 ocorre a concepção da ideia inicial; no passo 2 é feita a fundamentação teórica acerca dessa ideia. No passo 3, são planejadas as atividades a serem desenvolvidas nos passos 4, 5 e 6, havendo possíveis revisões desse planejamento ao final de cada passo. No passo 4 são executadas as atividades para a implementação da ação; no passo 5 são executadas as atividades para o monitoramento. No passo 6 a evolução da ação é avaliada. A ordem de execução dos passos 4, 5 e 6 depende da sequência proposta para as atividades, desempenhadas nesses passos. Se for necessária a alteração das atividades planejadas, essa é feita como passo 7. O passo 8 indica a finalização do plano de ação.

A estrutura da pesquisa, quanto ao procedimento técnico, é delineada por uma pesquisa-ação prática; quanto ao objetivo, se classifica como experimental. A abordagem da pesquisa para o levantamento de dados é qualitativa e quantitativa. O instrumento utilizado para a coleta de dados é classificado como questionário, que foi adotado por facilitar o anonimato do participante, minimizando possíveis desconfortos quanto ao processo de aferição do conhecimento adquirido.

B. Recursos Tecnológicos

Para o desenvolvimento da pesquisa fez-se uso dos seguintes recursos tecnológicos:

- Quadro-negro e giz;
- Projetor multimídia;
- Computadores Desktop na proporção de 1 por aluno;
- Conexão com Internet;
- Scratch como linguagem visual, utilizando a versão online;
- Programa Portugol Online para programar com pseudocódigo.

As aulas foram ministradas em laboratório de computação, dispondo de todos os recursos necessários para a condução da pesquisa.

C. Descrição da População

Participaram da pesquisa alunos regularmente matriculados na disciplina de Introdução à Computação, ministrada para o curso de Licenciatura em Física, durante o segundo semestre letivo da Universidade Federal de Goiás Regional Jataí, no ano de 2017. Inicialmente todos os alunos matriculados participaram da pesquisa. Entretanto, durante o semestre houve evasão de alguns alunos do curso, e assim, deixaram de participar da pesquisa. A quantidade de participantes ao final da pesquisa totalizou 7, sendo 1 representante do sexo feminino e 6 do sexo masculino; 6 com idade entre 19 e 26 anos e 1 com idade entre 33 e 40 anos. Todos cursavam a disciplina pela primeira vez, e segundo eles, não possuiam conhecimento com o conteúdo.

IV. Plano de Ação da Pesquisa

Para a condução da pesquisa foi elaborado um plano de ação contemplando os passos propostos por McKay e Marshall (ver Fig. 1). Esses passos estão descritos como segue.

A. Idéia Inicial

O desenvolvimento do pensamento computacional pode ser auxiliado com a utilização da linguagem de programação visual Scratch, durante o processo de ensino-aprendizagem de lógica e algoritmos de programação de computadores.

B. Reconhecimento

O pensamento computacional, enquanto habilidade fundamental para todos [1] [2] [3], pode auxiliar no processo de resolução de problemas do mundo real. Desenvolver essa habilidade implica em compreender os processos de resolução de problemas executados pelo computador, o que envolve compreender os conceitos de lógica e algoritmos de programação.

Ensinar programação não é tarefa fácil e requer didática adequada, pois muitos são os problemas relacionados às dificuldades no aprendizado dessa matéria, como por exemplo: falta de conhecimento prévio dos pré-requisitos em matemática, dissociação entre os exercícios propostos com o mundo real, dificuldades na compreensão do assunto e pouca disponibilidade extraclasse [18]. De acordo com Júnior, a adoção de metodologias pedagógicas apropriadas auxilia na superação desses problemas [5].

A utilização das linguagens de programação visual minimiza as dificuldades com a compreensão das linguagens de programação comuns. A linguagem Scratch por exemplo, utiliza a descrição do código em blocos, propiciando um envolvimento mais lúdico com a programação, contribuindo dessa forma, para o desenvolvimento de conhecimento sobre conceitos básicos relacionados à computação.

C. Planejamento das Atividades

As atividades planejadas no passo 3 estão descritas abaixo:

a) aplicação da avaliação diagnóstica: a avaliação diagnóstica é aplicada antes do início de cada aula de um novo conteúdo;

b) apresentação do conteúdo: o conteúdo é apresentado aos alunos utilizando quadro-negro, giz, computador e projetor multimídia;

c) programação utilizando Scratch: o conteúdo apresentado é incorporado à um programa desenvolvido pelos alunos, utilizando a linguagem Scratch;

d) aplicação da avaliação de desempenho com Scratch: a avaliação é aplicada após os alunos encerrarem o exercício proposto utilizando a linguagem Scratch;

e) aferição do progresso com Scratch a partir da comparação entre as avaliações diagnóstica e de desempenho: as notas obtidas pelos alunos com as avaliações diagnóstica e de desempenho são calculadas, e a partir da subtração entre elas, têm-se o progresso da programação com Scratch;

f) programação utilizando pseudocódigo: o conteúdo apresentado é incorporado à um programa desenvolvido pelos alunos, utilizando pseudocódigo;

g) aplicação da avaliação de desempenho com pseudocódigo: a avaliação é aplicada após os alunos encerrarem o exercício proposto utilizando pseudocódigo;

h) aferição do progresso com pseudocódigo a partir da comparação entre as avaliações diagnóstica e de desempenho: as notas obtidas pelos alunos com as avaliações diagnóstica e de desempenho são calculadas, e a partir da subtração entre elas, têm-se o progresso da programação com pseudocódigo;

i) comparação entre os resultados do progresso com Scratch e pseudocódigo: as notas obtidas no progresso com cada uma das ferramentas são comparadas, no intuito de identificar qual das abordagens apresenta maior contribuição para o processo de ensino-aprendizagem de lógica e algoritmos de programação.

As atividades anteriormente citadas são executadas nos passos 4, 5 e 6, e são descritos na sequência.

D. Implementação

As atividades executadas para implementar a ação são: aplicação da avaliação diagnóstica, apresentação do conteúdo, programação utilizando Scratch e programação utilizando pseudocódigo.

As avaliações, tanto diagnóstica como de desempenho, são idênticas e possuem 5 questões. As questões apresentadas nas avaliações são (onde se lê conteúdo, é substituído pelo conteúdo apresentado na aula):

- Descreva o significado da palavra conteúdo:
- O que significa conteúdo para você?
- O que significa conteúdo em uma linguagem de programação de computadores?
- Exemplifique o uso de um conteúdo em um contexto que não esteja relacionado à programação de computadores:
- Exemplifique o uso de um conteúdo em um código de programação para computadores:

Cada resposta certa equivale a 2 pontos, totalizando no máximo 10 pontos em cada avaliação.

E. Monitoramento

Para monitorar a ação, são executadas as seguintes atividades: aplicação da avaliação de desempenho com Scratch e aplicação da avaliação de desempenho com pseudocódigo. Essas avaliações têm como objetivo constatar o nível de domínio de conteúdo que cada aluno apresenta, sem, no entanto, provocar a manifestação de preocupação no aluno quanto à pontuação adquirida acerca de seu conhecimento, pois não tem caráter punitivo. As avaliações diagnóstica e de desempenho serão analisadas no intuito de prover dados para verificar a evolução do efeito da ação

F. Evolução do Efeito da Ação

A evolução do efeito da ação é medida com a execução das atividades: aferição do progresso com Scratch a partir da comparação entre as avaliações diagnóstica e de desempenho, aferição do progresso com pseudocódigo a partir da comparação entre as avaliações diagnóstica e de desempenho, e, com a comparação entre os resultados do progresso com Scratch e pseudocódigo.

G. Alteração das Atividades

O passo 7 sugere a alteração das atividades planejadas no passo 3, no intuito de alterar o plano inicial da ação para modificar a ferramenta que será trabalhada inicialmente. A ferramenta que apresentar menor variação positiva, entre os resultados das avaliações de desempenho e diagnóstica, é trabalhada no início da etapa do novo conteúdo. O objetivo dessa alteração é propiciar condições de comparação mais justas entre as ferramentas, sem que a utilização da primeira colabore com, ou prejudique o progresso da segunda, uma vez que o aluno terá maior domínio do conteúdo quando for utilizar a segunda ferramenta.

H. Saída da Execução do Plano

A execução do plano da ação será finalizada quando: for encerrada a pesquisa com a aplicação de 4 conteúdos, configurando 4 etapas; ou, se for verificado que a média das

notas do progresso de aprendizagem dos alunos não atingiu nota 6. Uma outra alternativa que deve ser considerada para a saída da execução do plano, é o caso de algum participante se queixar da metodologia empregada, alegando insatisfação ou desconforto com a mesma.

V. Resultados e Discussão

Os passos 4 e 5 do ciclo da pesquisa foram executados em laboratórios de informática durante as aulas da disciplina de Introdução à Computação. O conjunto dos passos 4, 5 e 6 foi igualmente executado para cada conteúdo; esse conjunto será tratado como etapa. Os conteúdos apresentados na etapa 1, 2, 3 e 4 foram, respectivamente: conceito de variáveis; estrutura sequencial, estrutura condicional e estrutura de repetição. Cada conteúdo foi ministrado durante 4 aulas, de 50 minutos cada. Durante todo o processo da pesquisa, o aluno foi identificado por um único código, conhecido apenas por ele.

As Tabelas 1 e 2 mostram as notas obtidas pelos alunos nas avaliações da etapa 1 e etapa 2, respectivamente:

TABLE I. Notas Obtidas Pelos Participantes Na Etapa 1

Código do aluno	**Avaliações na etapa 1**		
	Diagnóstica	***Desempenho Scratch***	***Desempenho Pseudocódigo***
92	5	7	8
666	6	7	6
1313	6	8	6
1387	4	6	5
1981	5	7	6
4646	0	0	4
6969	4	8	6

TABLE II. Notas Obtidas Pelos Participantes Na Etapa 2

Código do aluno	**Avaliações na etapa 2**		
	Diagnóstica	***Desempenho Pseudocódigo***	***Desempenho Scratch***
92	5	6	7
666	4	4	5
1313	2	2	6
1387	4	7	5
1981	0	2	2
4646	4	4	4
6969	4	7	4

Após a execução da etapa 1, os resultados das avaliações foram analisados, a fim de identificar a necessidade de uma intervenção no planejamento da ação. De acordo com os resultados, a ferramenta que apresentou menor variação positiva, entre os resultados das avaliações de desempenho e diagnóstica, foi utilizada para iniciar a próxima etapa, alterando o planejamento inicial. Essa forma de intervenção possibilitou aferir com maior cuidado o desempenho de cada ferramenta, sem que a utilização consecutiva de uma favorecesse a outra. De acordo com os resultados da Tabela 1 e Tabela 2, é possível verificar qual a contribuição de cada ferramenta, em cada uma das etapas. A Tabela 3 mostra o progresso com as ferramentas, evidenciado pelas variações entre a avaliação diagnóstica e as avaliações de desempenho.

TABLE III. Progresso Scratch X Pseudocódigo

Código do aluno	**Scratch X Pseudocódigo**			
	Scratch etapa 1	***Pseudocódigo etapa 1***	***Scratch etapa 2***	***Pseudocódigo etapa 2***
92	2	3	2	1
666	1	0	1	0
1313	2	0	4	0
1387	2	1	1	3
1981	2	1	2	2
4646	0	4	0	0
6969	4	2	0	3

A etapa 1 iniciou com a utilização da linguagem Scratch. Entretanto, foi verificado que o desempenho com pseudocódigo foi pior. Houve a intervenção e modificação do plano inicial, fazendo com que a etapa 2 fosse iniciada com a utilização de pseudocódigo. Mesmo após a intervenção no plano inicial, os resultados apontaram que, geralmente, as notas de desempenho com a linguagem Scratch eram superiores às notas obtidas com a utilização de pseudocódigo.

A etapa 3 também foi iniciada com a utilização de pseudocódigo, pois o progresso obtido com pseudocódigo na etapa 2, assim como na etapa 1, foi inferior ao obtido com Scratch na maioria das notas. A Tabela 4 mostra as notas obtidas com as avaliações na terceira etapa:

TABLE IV. Notas Obtidas Pelos Participantes Na Etapa 3

Código do aluno	**Avaliações na etapa 3**		
	Diagnóstica	***Desempenho Pseudocódigo***	***Desempenho Scratch***
92	2	8	5
666	0	2	2
1313	6	8	8
1387	5	6	5
1981	2	7	2
4646	4	4	4
6969	5	9	8

Como pode ser visto na Tabela 4, o desempenho com pseudocódigo foi melhor que o desempenho com Scratch para o conteúdo ministrado na etapa 3. Nessa etapa foi abordado o conteúdo sobre estrutura condicional. A mudança da ferramenta que apresenta menor contribuição, novamente interfere na execução do plano; assim, a etapa 4 foi iniciada com a utilização da linguagem Scratch.

De acordo com a sequência de utilização de ferramentas, temos que:

- Etapa 1: iniciou com Scratch;
- Etapa 2: iniciou com pseudocódigo;
- Etapa 3: iniciou com pseudocódigo;
- Etapa 4: iniciou com Scratch.

A Tabela 5 mostra as notas obtidas com as avaliações na quarta etapa:

TABLE V. NOTAS OBTIDAS PELOS PARTICIPANTES NA ETAPA 4

Código do aluno	Avaliações na etapa 4		
	Diagnóstica	***Desempenho Scratch***	***Desempenho Pseudocódigo***
92	0	5	9
666	0	3	7
1313	0	4	6
1387	0	3	8
1981	0	5	8
4646	0	3	6
6969	0	4	7

Na etapa 4, foi verificado que o desempenho com pseudocódigo se manteve superior ao de Scratch. A Tabela 6 mostra o progresso com as ferramentas nas etapas 3 e 4:

TABLE VI. PROGRESSO SCRATCH X PSEUDOCÓDIGO

Código do aluno	Scratch X Pseudocódigo			
	Scratch etapa 3	***Pseudocódigo etapa 3***	***Scratch etapa 4***	***Pseudocódigo etapa 4***
92	3	6	5	9
666	2	2	3	7
1313	2	2	4	6
1387	0	1	3	8
1981	0	5	5	8
4646	0	0	3	6
6969	3	4	4	7

Conforme pode ser observado, as etapas 1 e 2 tiveram uma maior contribuição no desempenho das notas, com a linguagem Scratch; já as etapas 3 e 4 ocorre o inverso, as maiores contribuições estão relacionadas ao pseudocódigo. Essa mudança no avanço, pode estar relacionada ao fato dos conteúdos iniciais serem menos complexos, o que pode ser melhor representado com blocos; já os conteúdos posteriores, ou foram mais facilmente compreendidos com o pseudocódigo, pelo fato do aluno já estar assimilando-o mais à uma linguagem de programação comum, ou simplesmente esses conteúdos são mais difíceis de serem modelados com blocos devido a quantidade de peças que são necessárias para criar o cenário.

Outra alternativa para essa mudança na direção das notas, pode estar relacionada à manifestação do desenvolvimento do pensamento no aluno. Assim, a partir da compreensão dos conceitos mais básicos sobre programação, o aluno já consegue abstrair melhor o processo de modelagem do problema, tendo dificuldades para descrevê-lo em blocos, visto que é uma representação mais lúdica para a programação.

É necessário salientar, que nem todas as notas que aparecem como 0, significam exatamente que o aluno respondeu ao questionário e não acertou qualquer uma das questões. Essa nota também foi atribuída para os alunos que não estiveram presentes no dia da aula; uma vez que os alunos não tinham seus questionários relacionados a si, não era possível controlar quais participavam ou não de determinada etapa.

Contudo, quando os alunos foram questionados sobre a preferência de utilização de ferramenta, 5 deles disseram preferir programar com Scratch; 2 com pseudocódigo. A alegação da preferência pode estar relacionada à facilidade com que a maioria conseguiu memorizar os comandos, seja pelas cores dos blocos, seja pelos atores que promovem uma maior interação entre o programa e o programador, o entretenimento promovido pela linguagem, ou ainda, outro fator não citado pelos alunos.

Ao final da pesquisa, todos os alunos responderam uma autoavaliação para que fosse possível analisar a satisfação desses alunos em participar da pesquisa, bem como o engajamento deles durante todo o processo.

Todos os participantes afirmaram estar aprendendo com a utilização das duas ferramentas, embora a maioria desejasse a utilização da linguagem Scratch.

VI. CONSIDERAÇÕES FINAIS

De acordo com os resultados apresentados neste trabalho, é possível verificar que a linguagem de programação visual Scratch contribui de forma positiva no processo de ensino-aprendizagem de lógica e algoritmos de programação. Embora a sua contribuição não seja superior à de pseudocódigo em todas as etapas, para conteúdos menos complexos, Scratch mostrou ser uma alternativa bastante interessante para se aplicar no processo de ensino-aprendizagem, tanto pelo olhar do docente quanto pelo do aluno. Outro ponto que deve ser considerado é o fato de nem sempre as notas apresentarem variação positiva entre as avaliações, embora fosse desejável, mostrando que o aprendizado é constantemente construído. É importante ressaltar que a adequação dos recursos às etapas de ensino, de forma conveniente para o processo de ensino-aprendizagem, propicia um melhor aproveitamento dos mesmos.

AGRADECIMENTOS

Os autores agradecem ao apoio da Universidade Federal de Jataí, aos participantes da pesquisa e aos membros do Jataí ACM SIGCSE *Chapter*.

REFERÊNCIAS

[1] J. Wing, “Pensamento Computacional – Um conjunto de atitudes e habilidades que todos, não só cientistas da computação, ficaram ansiosos para aprender e usar,” Revista Brasileira de Ensino de Ciência e Tecnologia, vol. 9, n. 2, 2016.

[2] P. Blikstein, “O pensamento computacional e a reinvenção do computador na educação”. 2008. [Online]. Disponível em: http://www.blikstein.com/paulo/documents/online/ol_pensamento_computacional.html.

[3] A. Bundy, “Computational Thinking Is Pervasive,” Journal of Scientific and Practical Computing, vol. 1, n. 2, pp. 67-69, 2007.

[4] M. Resnick, et al, “Scratch: programming for all,” Communications of the ACM 52.11, pp. 60-67, November 2009.

[5] J. C. R. P. Júnior and C. E. Rapkiewicz, “O processo de ensino-aprendizagem de fundamentos de Programação: uma visão crítica da pesquisa no Brasil,” Anais do XII Workshop sobre Educação em Computação (SBC). 2004.

[6] T. R. Silva, R. Lopes, T. J. Medeiros, E. Aranha and H. Medeiros, “Ensino-aprendizagem de programação: uma revisão sistemática da literatura,” Revista Brasileira de Informática na Educação, vol. 23, n. 1, 2015.

[7] R. França and P. Tedesco, “Desafios e oportunidades ao ensino do pensamento computacional na educação básica no Brasil,” In Anais dos Workshops do Congresso Brasileiro de Informática na Educação vol 4, n. 1, p. 1464, 2015.

[8] Lifelong Kindergarten, research group at the MIT Media Lab. [Online]. Disponível em: https://www.media.mit.edu/projects/scratch/overview/.

[9] G. P. Santos and R. S. Bezerra, “Desenvolvendo o pensamento computacional utilizando Scratch e lógica matemática,” VI Jornada de Atualização em Informática na Educação (JAIE 2017). Congresso Brasileiro de Informática na Educação (CBIE 2017). pp. 66-99, 2017.

[10] M. J. Marcelino, T. Pessoa, C. Vieira, T. Salvador and A. J. Mendes, “Learning Computational Thinking and Scratch at Distance,” Computers in Human Behavior, vol. 80, pp. 470–477, 2018.

[11] E. Ribas, G. D. Bianco and R. A. Lahm, “Programação visual para introdução ao ensino de programação na Educação Superior: uma análise prática,” RENOTE – Revista Novas Tecnologias na Educação, vol. 14 n. 2, 2016.

[12] J. M. Sáez-López, M. Román-González and E. Vázquez-Cano, “Visual programming languages integrated across the curriculum in elementary school: A two year case study using “Scratch” in five schools,” Computers & Education, vol. 97, pp. 129-141, 2016.

[13] M. L. Q. Bressan and M. A. Amaral, “Avaliando a Contribuição do Scratch para a Aprendizagem pela Solução de Problemas e o Desenvolvimento Criativo,” Intersaberes, vol. 10, n. 21, pp. 509–527, 2015.

[14] M. G. Lodico, D. T. Spaulding and K. H. Voegtle, “Action research”, Methods in educational research: From theory to practice, Capítulo 12, pp. 311–359, John Wiley & Sons, 2010.

[15] S. Kemmis, Action research as a practice-based practice. Educational Action Research, vol. 17, n. 3, pp. 463–474, 2009.

[16] M. Thiollent, Metodologia da pesquisa-ação, Cortez Editora, 2011.

[17] J. McKay and P. Marshall, “The dual imperatives of action research”, Information Technology & People, vol. 14, n. 1, pp. 46-59, 2001.

[18] L. Giraffa, L. Muller and M. C. Moraes. “Ensinando Programação apoiada por um ambiente virtual e exercícios associados a cotidiano dos alunos: compartilhando alternativas e lições aprendidas,” Anais dos Workshops do Congresso Brasileiro de Informática na Educação. vol. 4, n. 1, p. 1330, 2015.