https://doi.org/10.34179/revisem.v5i2.14421

A LINGUAGEM DE PROGRAMAÇÃO SCRATCH NA FORMAÇÃO DO PROFESSOR: UMA ABORDAGEM BASEADA NO TPACK

SCRATCH PROGRAMMING LANGUAGE IN TEACHER TRAINING: A TPACK-BASED APPROACH

Ana Karina de Oliveira Rocha
Universidade Federal de Sergipe – UFS
karinina@gmail.com

Maria Elisabette Brisola Brito Prado
Universidade Anhanguera de São Pulo – UNIAN
bette.prado@gmail.com

José Armando Valente
Universidade Estadual de Campinas – UNICAMP
jvalente@unicamp.br

Resumo

Este artigo tem como objetivo analisar as possibilidades do uso da atividade de programação Scratch na formação continuada do professor de Matemática na perspectiva da integração de conhecimentos, segundo modelo teórico do TPACK. A metodologia de natureza qualitativa norteou a coleta de dados e as ações desenvolvidas durante o curso de formação sobre Linguagem de Programação Scratch no contexto de Matemática, com a participação de um grupo de dez professores que atuam na Educação Básica na rede pública de ensino da cidade de São Paulo. No curso, os professores vivenciaram um modo de aprender baseado na perspectiva construcionista e essa experiência inspirou o grupo a desenvolver o software educacional sobre generalização de padrões de sequências numéricas para ser utilizado com seus alunos. Durante a elaboração do software, os professores foram descrevendo seus algoritmos, aplicando os comandos da linguagem, articulando conceitos matemáticos e computacionais. Nesse processo, o conhecimento do conteúdo matemático foi sendo realimentado pelo conhecimento da linguagem de programação do Scratch e vice-versa. Além disso, essa experiência fez com que os professores refletissem a respeito da maneira de intervir em uma situação de erro do aluno, enquanto construíam o software educativo. Esse fato mostra que a tecnologia no âmbito da educação e pautada pelos princípios construcionistas ganha uma nova dimensão cognitiva vinculada a processos reflexivos que podem viabilizar a reconstrução de conhecimentos da tecnologia, pedagogia e do conteúdo curricular de forma integrada, resultando em um novo conhecimento sobre e para prática do professor.

Palavras-chave: Educação Matemática. Construcionismo. Reconstrução de Conhecimento.

Abstract

This article aims to analyze the possibilities of using Scratch programming activity in mathematics teacher continuing education from the perspective of knowledge integration, according to the TPACK theoretical model. Qualitative methodology guided the data collection and the actions developed during the training course on Scratch Programming Language in the context of Mathematics. A group of ten teachers working in K-12 Education from a public school system in the city of São Paulo participated in the course. In the course, the teachers experienced a way of learning based on the constructionist approach. This experience inspired the group to develop an educational software on the generalization of numerical sequence patterns to be used with their students. During the development of the software, teachers were describing their algorithms using the language commands, articulating mathematical and computational concepts. In this process, the mathematical content knowledge was supported by the knowledge of the Scratch programming language and vice versa. In addition, this experience enabled teachers to reflect on how to intervene in the student's error, while building the educational software. This fact shows that when the technology is used in an educational context and based on constructionist principles it gains a new cognitive dimension linked to reflective processes that can provide the reconstruction of knowledge about technology, pedagogy and curricular content in an integrated manner, resulting in new knowledge about and for teacher practice.
Keywords: Mathematical Education. Constructionism. Knowledge Reconstruction.

INTRODUÇÃO

Este artigo se trata de um recorte da pesquisa de doutorado da primeira autora que buscou compreender como ocorreu o processo de integração entre os conhecimentos matemáticos, pedagógicos e tecnológicos por um grupo de professores da Educação Básica durante um curso de formação continuada.

Intencionalmente, esse curso se propôs a usar a linguagem Scratch como objeto de estudo e de reflexão na formação continuada de professores de Matemática. Isso ocorreu pelo fato de oportunizar ao grupo de professores vivenciar um novo modo de aprender, utilizando a atividade de programação baseada na abordagem construcionista. Segundo Papert e Harel (1991), o construcionismo enfatiza a criação de situações de aprendizagem que favoreçam tanto o *aprender-com* como o aprender-*sobre-o-pensar*. É a ideia do *hands-on and head-in*, ou seja, da relação entre o fazer (mão na massa) e o compreender (mente envolvida), no processo de construir algo com significado para o aprendiz. Para Valente (2005), essa relação entre o fazer e o compreender pode se dar durante a atividade de programação por meio de um movimento cíclico de ações-pensamentos: *descrição-execução-reflexão depuração e (nova) descrição* que se estabelece na interação do usuário, no caso o professor, com o computador durante a programação. O desencadeamento da *reflexão* e da *depuração* é que propicia o movimento cíclico de ações-pensamentos na forma de uma espiral ascendente da evolução do conhecimento.

Os princípios construcionistas, embora tenham surgido na atividade de programação, devem nortear as práticas pedagógicas independentemente do recurso utilizado. No caso da formação de professores, o fazer e o compreender tornam-se fundamentais para que o processo de apropriação da tecnologia possa ir além do domínio operacional.

Além disso, tais princípios diferem-se daqueles que norteiam, muitas vezes, as propostas de formação de professores para o uso pedagógico das tecnologias, as quais enfatizam uma abordagem teórica e/ou prática em propiciar o domínio operacional dos recursos tecnológicos. Alguns pesquisadores, como Almeida e Valente (2011), Prado e Lobo da Costa (2016), apontam que a formação do professor precisa proporcionar a esses profissionais da prática a reconstrução de conhecimentos na perspectiva da integração da tecnologia ao currículo da escola. Sob esse enfoque, Almeida e Valente (2011, p. 50) salientam que a formação do professor para a aplicação das tecnologias digitais de informação e comunicação (TDIC) como recurso pedagógico envolve:

> [...] muito mais do que provê-lo com o conhecimento técnico sobre as TDIC. Ela [a formação] deve criar condições para o professor construir conhecimento sobre os aspectos computacionais; compreender as perspectivas educacionais subjacentes aos *softwares* em uso, isto é, as noções de ensino, aprendizagem e conhecimento implícitas no *software*; e entender por que e como integrar o computador com o currículo e como concretizar esse processo na sua prática pedagógica.

Atentando-se aos aspectos abordados por esses autores, fica evidenciado que para o professor desenvolver uma prática integrando as tecnologias digitais ao currículo é preciso construir um novo tipo de conhecimento. Nesse sentido, Mishra e Koehler (2006) criaram um modelo que integra três diferentes áreas do conhecimento: tecnologia, pedagogia e o conteúdo específico a ser ensinado. Esse modelo foi denominado pelos autores de *Technological Pedagogical Content Knowledge* (TPACK) e representa uma estrutura teórica para auxiliar na compreensão da natureza dos conhecimentos necessários a serem mobilizados pelos professores na docência com o uso das TDIC.

Considerando a importância desse modelo teórico do TPACK, este artigo apresenta as possibilidades do uso da atividade de programação Scratch na formação continuada do professor e analisa o processo de integração de conhecimentos necessários para a prática docente.

LINGUAGEM DE PROGRAMAÇÃO SCRATCH

A linguagem Scratch foi desenvolvida em 2007 pelo *Lifelong Kidendarten Group*, grupo de pesquisa liderado por Mitchel Resnick e que faz parte do *Media Lab do Massachusetts Institute of Technology* (MIT), com o objetivo de propiciar a participação de pessoas das mais variadas idades, formações e nacionalidades na criação de *softwares* como: jogos, animações e simulações (RESNICK *et al*., 2009). De acordo com o referido autor, o Scratch oferece suporte ao pensamento computacional, proporcionando às pessoas a aprendizagem acerca de resoluções de problemas na Matemática e o desenvolvimento de estratégias de *design* que podem ser aproveitadas em outros domínios do conhecimento.

A gramática da linguagem Scratch baseia-se em uma coleção de blocos de comandos organizados em várias categorias (com cores distintas), que podem ser encaixados e encadeados de forma a produzir ações desejadas. Os blocos são constituídos por comandos, conforme suas funções, por exemplo, comandos de movimentos, de controle, sensores, operadores lógicos, variáveis, entre outros. A Figura 1 exemplifica um bloco de comandos que verifica se dois valores são iguais ou diferentes:

Figura 1 – Exemplo de comandos da linguagem Scratch

Fonte: Acervo da pesquisa

Os comandos da Figura 2 à esquerda podem ser empregados em atividades envolvendo conteúdo da geometria e trigonometria, pois possibilitam a utilização de conceitos como ângulo e direção, cujas medidas estão definidas em graus. Os comandos da Figura 2 à direita podem ser usados em conteúdo da Matemática que explora o eixo cartesiano, por exemplo, a construção do gráfico de funções de 1.º e 2.º graus, além de viabilizar a construção de figuras geométricas na tela, com o auxílio da posição das coordenadas (x, y).

Figura 2 esquerda – Comandos para uso de ângulos; **Figura 2 direita** – Comandos para movimento no plano cartesiano

Fonte: Acervo da pesquisa

Há comandos que são utilizados na atribuição de valores para as variáveis, conforme mostra o exemplo na Figura 3 à esquerda, e outro recurso do Scratch, ilustrado pelo conjunto de comandos na Figura 3 à direita, serve para armazenar valores, mantendo a ordem linear entre eles, constituindo uma estrutura de dados, denominada Lista.

Figura 3 esquerda – Comandos para atribuição de variáveis; **Figura 3 direita** – Comandos da estrutura de dados Lista

Fonte: Acervo da pesquisa

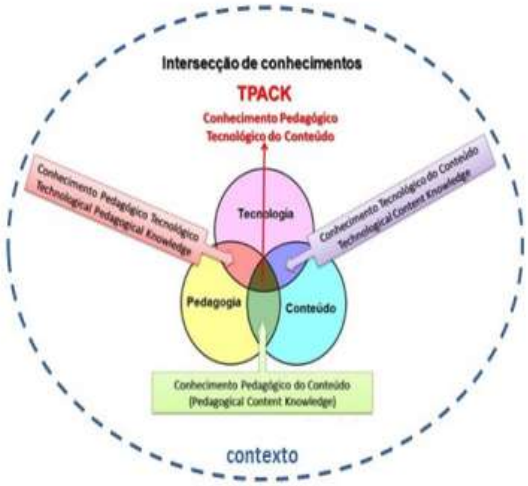
Além dos comandos existentes, a linguagem tem operadores aritméticos, tais como adição, subtração, multiplicação e divisão; operadores de comparação: menor que, igualdade, maior que; operadores lógicos: <E>, <OU>, <NÃO>; e as funções matemáticas predefinidas que executam operações como: raiz quadrada, logaritmo, arredondamento de um número, entre outros. Na linguagem Scratch, os operadores lógicos são utilizados em blocos de comandos do tipo: "Se então (execute A) senão (execute B)", enquanto A e B são procedimentos ou comandos definidos por quem está programando com a linguagem.

Esses são alguns dos comandos da linguagem Scratch que mostram as possibilidades de recursos relacionados a conceitos matemáticos que podem ser utilizados em diversas atividades envolvendo resolução de problemas e criação de programas de diferentes níveis de complexidade.

MODELO TPACK NA FORMAÇÃO DO PROFESSOR

Na criação do modelo do TPACK, os autores Mishra e Koehler (2006) tomaram como ponto de partida a teoria da base de conhecimento dos professores para a docência, de Shulman (1986), que se constitui pela integração dos conhecimentos pedagógico e do conteúdo específico, dando origem na sua intersecção ao conhecimento pedagógico do conteúdo. A essa estrutura foi integrada pelos autores do TPACK o conhecimento tecnológico, conforme mostra a Figura 4:

Figura 4 – Conhecimento Pedagógico Tecnológico do Conteúdo



Fonte: Adaptação do original – The TPACK framework and its knowledge Components (KOEHLER; MISHRA, 2009, p. 63).

A constituição do TPACK, como indicado na Figura 4, ocorre a partir de três intersecções envolvendo os conhecimentos pedagógicos, tecnológicos e do conteúdo específico. A intersecção entre os conhecimentos da pedagogia e do conteúdo resulta no Conhecimento Pedagógico do Conteúdo (CPC), herdado das ideias de Shulman (1986, 1987), e se refere ao conhecimento do professor de quais abordagens pedagógicas são adequadas para o ensino de um determinado conteúdo. A intersecção entre pedagogia e tecnologia gera o Conhecimento Pedagógico Tecnológico (CPT), que significa conhecer as potencialidades dos diferentes recursos tecnológicos e suas implicações nos processos de ensino e de aprendizagem; e da intersecção entre conteúdo e tecnologia deriva o Conhecimento Tecnológico do Conteúdo (CTC), o qual implica conhecer o conteúdo de modo a utilizá-lo em diferentes formas de representação viabilizadas pelos recursos tecnológicos. Esses três tipos de conhecimentos – (CPC), (CPT) e (CTC) – também se interseccionam, originando o Conhecimento Tecnológico e Pedagógico do Conteúdo (em inglês, *Technological Pedagogical Content Knowledge* – TPACK), situado em um determinado contexto.

Esse modelo teórico TPACK, nas últimas décadas, vem sendo estudado no contexto da formação continuada de professores de Matemática por vários pesquisadores, tais como Vieira (2013), Rocha (2015), Cibotto e Oliveira (2017), Prado e Lobo da Costa (2016), Colling e Richit (2020), entre outros, que salientam a complexidade do processo de formação e, ao mesmo tempo, anunciam que este pode ser um caminho possível para propiciar o desenvolvimento profissional docente, considerando as características da sociedade tecnológica.

CONTEXTO DO ESTUDO

A metodologia de caráter qualitativo norteou a coleta de dados e as ações realizadas durante uma formação continuada desenvolvida no laboratório da Diretoria de Ensino – Norte 2, localizada na cidade de São Paulo. Participaram do curso dez professores que atuavam na Educação Básica na rede pública de ensino da cidade de São Paulo. Por meio de uma parceria entre o Programa de Pós-Graduação em Educação Matemática da Universidade Anhanguera de São Paulo e a Diretoria de Ensino – Norte 2 da Secretaria de Educação do Estado de São Paulo, foi oferecido um curso de formação continuada sobre Linguagem de Programação Scratch no contexto de Matemática.

A formação ocorreu em dez encontros de três horas cada, perfazendo 30 horas de atividades. Além dos encontros presenciais, os participantes tiveram acesso ao ambiente virtual no Moodle, que foi customizado para ser um espaço complementar de aprendizagem. Nesse espaço virtual, eram disponibilizados os materiais de apoio, tutoriais e textos, além das ferramentas de comunicação para viabilizar as interações entre todos os envolvidos, favorecendo a troca de ideias, o esclarecimento de dúvidas e o compartilhamento de reflexões.

Para a coleta de dados foram utilizados como instrumentos: questionário, registros escritos e gravados em áudio dos encontros de formação e protocolos de atividades realizadas pelos professores participantes. Esses professores, denominados S01, S02.... S10, tinham familiaridade com vários recursos tecnológicos e conheciam alguns *softwares* específicos de Matemática, no entanto não tinham nenhuma experiência com atividade de programação.

Considerando o perfil dos participantes, os primeiros encontros formativos orientaram o grupo de professores a desenvolver pequenos programas para explorar os comandos da linguagem Scratch e, em seguida, a proposta envolveu a criação de um *software* educacional compreendendo o conceito de generalização na Matemática.

APRENDIZAGEM DO SCRATCH SEGUNDO O MODELO TPACK

Tendo em vista o fato de o grupo de professores nunca ter realizado atividade de programação, tampouco conhecer a linguagem Scratch, foi necessário primeiramente explorar os comandos e o ambiente de programação por meio da elaboração de pequenos programas, principalmente aqueles que envolviam interação do programa com o usuário.

O programa "Número ímpar ou par?", por exemplo, mostra o processo utilizado por alguns professores em sua elaboração, envolvendo a descrição do algoritmo em linguagem natural, a construção de um fluxograma e o procedimento via linguagem computacional.

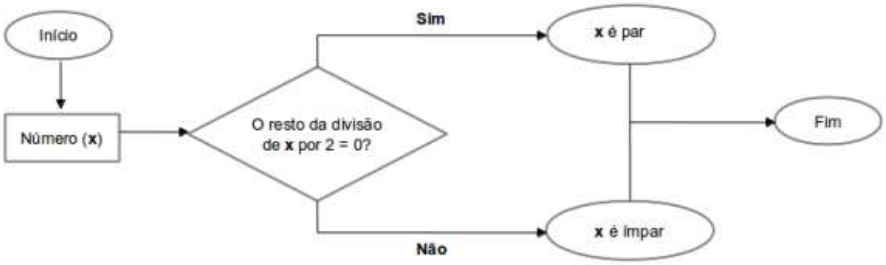
> Programa "Número ímpar ou par?"
> Construir um programa que, dado um número x do conjunto dos números naturais fornecido pelo usuário, diga se ele é par ou ímpar.

Resolução: Algoritmo em Linguagem Natural, representado pelo fluxograma da Figura 5.

Sendo x ϵ N* um número fornecido pelo usuário, calcule o resto da divisão do número x por 2. Se o resto da divisão do número x por 2 for igual a zero, o número x é par. Senão, o número x é ímpar.

Resolução: Algoritmo representado no fluxograma

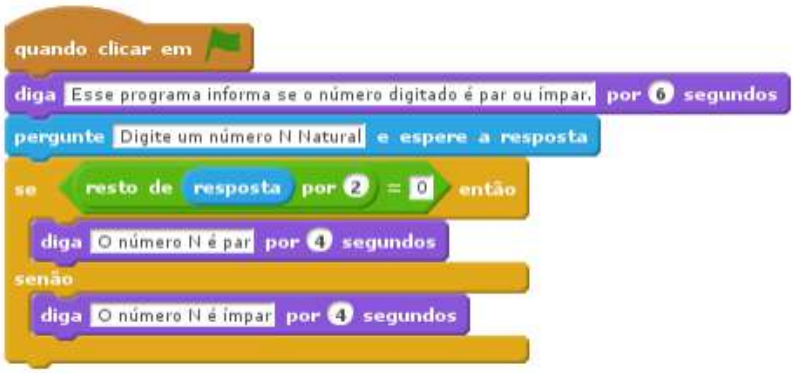
Figura 5 – Fluxograma "Número ímpar ou par?"



Fonte: Rocha (2015, p. 106)

Resolução: Algoritmo na linguagem computacional, usando os blocos de comandos do Scratch, indicado na Figura 6.

Figura 6 – Procedimento do programa "Número ímpar ou par?"



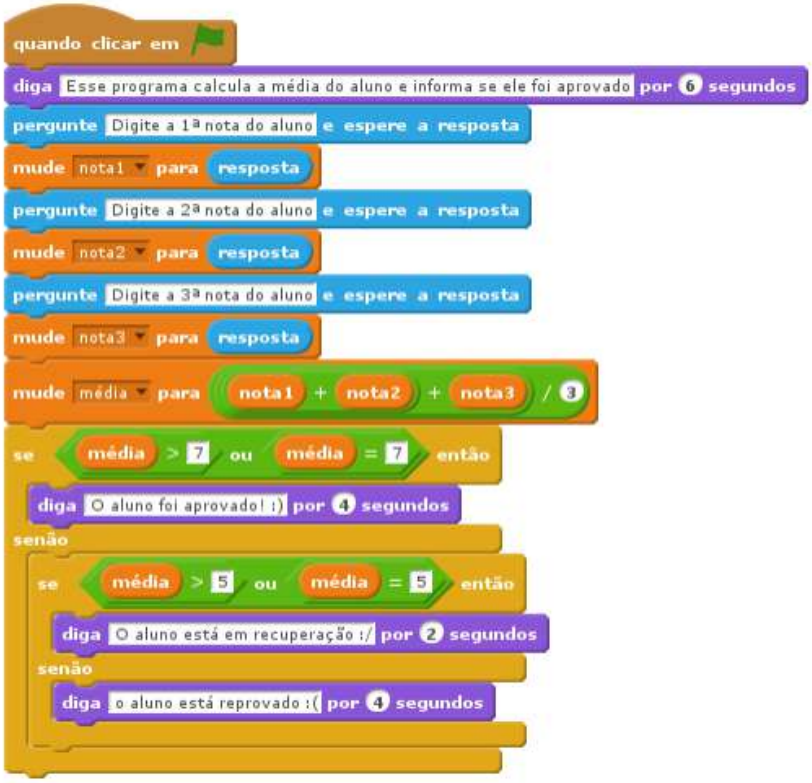
Fonte: Rocha (2015, p. 195).

Ao executar o programa "Número ímpar ou par?", aparecerão na tela a mensagem "Esse programa informa se o número digitado é par ou ímpar" e, em seguida, a pergunta "Digite um número N Natural". O programa aguardará o fornecimento de um número qualquer do conjunto dos números naturais a ser dado pelo usuário. Assim que digitado o número, ele será armazenado na variável "resposta". Em seguida, o programa testará se o valor dado pelo usuário é ímpar ou par, usando os comandos operadores: resto de [ ] por [ ] = [ ] e o comando de controle: Se-Senão. Caso o teste seja verdadeiro, será mostrada a mensagem "O número N é Par"; se não for verdadeiro, será exibida a mensagem "O número N é ímpar".

Esse tipo de programa que interage com usuário abordando o conteúdo matemático motivou o grupo de professores a criar outros programas, tais como Números Primos, Teorema de Pitágoras, Soma da Progressão Aritmética, além de diversas figuras geométricas. Durante o processo de elaboração desses programas, ampliou-se o repertório de possibilidades dos recursos computacionais do Scratch e, consequentemente, o grupo se familiarizou com a atividade de programação, de forma que os professores passaram a criar diretamente os procedimentos via linguagem computacional, como ilustra o Programa "Calculando a média", cujo procedimento é mostrado na Figura 7:

> Programa "Calculando a média"
> Construir um programa para calcular a média de um aluno, a partir de três notas fornecidas pelo usuário. O programa deve considerar que a média necessária para o aluno ser aprovado é, no mínimo, 7.0, e reprovado ou em recuperação, a partir do cálculo da média.

Figura 7 – Procedimento do programa "Calculando a média"



Fonte: Rocha (2015, p. 197)

Ao executar o programa “Calculando a média”, aparecerão na tela a mensagem “Esse programa calcula a média do aluno e informa se ele foi aprovado” e, em seguida, a pergunta “Digite a primeira nota do aluno”. O programa ficará na espera do fornecimento do valor da 1.ª nota a ser dado pelo usuário. Assim que digitado o valor, ele será armazenado na variável “nota1” e, na sequência, repetirá a mensagem da pergunta e aguardará o fornecimento dos valores da 2.ª e da 3.ª notas, as quais serão armazenadas nas variáveis “nota2” e “nota3”, respectivamente. O resultado da operação [ ] + [ ] + [ ] / [ ] ficará armazenado na variável “média”, para ser testado (utilizando dois pares da condicional); se o valor da operação for maior que [ ] ou igual [ ] a [ ], será mostrada a mensagem “O aluno foi aprovado”. Se for diferente, será realizado um novo teste utilizando um parâmetro diferente do anterior, e, se o valor da operação for maior que [ ] ou igual [ ], será exibida a mensagem “O aluno está em recuperação”. Caso contrário, será apresentada a mensagem “O aluno está reprovado”.

Portanto, ao criarem pequenos programas, os professores conheceram conceitos e comandos da linguagem em atividades matemáticas pontuais. No programa “Número ímpar ou par?”, os professores vivenciaram o processo de ensinar o computador a identificar se um determinado número é ímpar ou par, e, no programa “Calculando a

média", ensinaram o computador a fazer o cálculo da média das notas e, ainda, informar os possíveis resultados, aprovado, em recuperação e reprovado. Esse tipo de atividade provocou no grupo de professores o sentimento de *empoderamento,* ou seja, o reconhecimento de que para a máquina aprender ele precisa ter o conhecimento para ensiná-la – via programação.

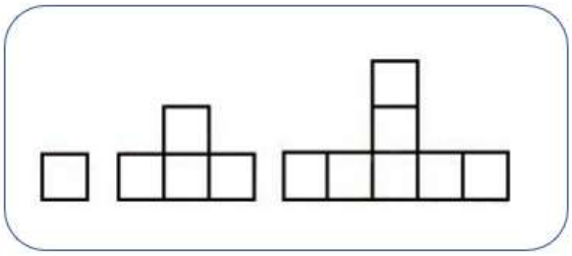
> [...] construções desenvolvidas pelos alunos mostra que o Scratch possibilita a manipulação de conceitos e símbolos de acordo com regras lógicas formais (cálculo proposicional), embora estas normalmente estejam implícitas para o programador. Portanto, a linguagem de programação Scratch, embora seja semelhante à linguagem natural, tem uma base matemática clara (VECCHIA; MALTEMPI; BORBA, 2015, p. 58 – tradução nossa).[1]

Durante o processo de aprender-fazendo programas com o Scratch, os professores reconheceram que a atividade de programar exige outra forma de representar o conhecimento matemático, requer ir além do conhecimento puramente baseado em aplicação de fórmulas ou de técnicas padronizadas. Os professores utilizaram não apenas o conhecimento matemático, mas também a maneira pela qual esse conteúdo poderia ser representado com o uso da tecnologia, numa situação em que ensinavam a máquina a aprender, demonstrando com isso o conhecimento construído, resultante da intersecção entre o conhecimento matemático e o computacional, denominado por Mishra e Koehler (2006), do *Conhecimento Tecnológico do Conteúdo* (CTC).

Assim, após essa vivência, o próximo passo do grupo de professores foi projetar e implementar um *software* educacional envolvendo o conteúdo de generalização de padrões com sequências numéricas, voltado para a realidade de seus alunos da educação básica, como exemplifica a imagem da Figura 8, utilizada pelo professor (S07) como ponto de partida para a elaboração do *software* educacional.

[1] "[...] the constructions developed by the students shows that Scratch enables the manipulation of concepts and symbols according to logical formal rules (propositional calculus), though these usually are implicit to the programmer. Therefore, the Scratch programming language, although it is similar to natural language, has a clear mathematical basis."

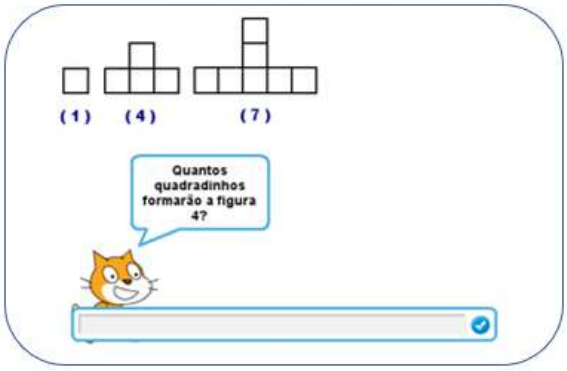
Figura 8 – Imagem da sequência utilizada no *software* educacional do professor (S07)



Fonte: Rocha (2015, p. 114)

O professor (S07) inicialmente elencou algumas fases para serem programadas, por exemplo, uma tela para apresentar o problema, uma situação para desencadear a interação entre o *software* e o aluno, o tratamento da resposta dada pelo aluno e o retorno da resposta. Para o *software* educacional interagir com os alunos, o professor criou um personagem, um tipo de avatar, usando a imagem de um *sprite* com a função de exibir na tela do computador uma pergunta sobre o conteúdo da Figura 8, conforme mostra a Figura 9:

Figura 9 – Tela do *software* educacional do professor (S07)



Fonte: Rocha (2015, p. 115)

Diante dessa tela, o aluno deveria responder (digitando) à questão feita pelo *sprite* e aguardar um retorno. Para tanto, o professor teve que ensinar o *software* a verificar se a resposta do aluno estava correta ou não, assim como a fornecer um *feedback*. A princípio, parecia ser algo simples de ser feito, pois o professor tinha como base as experiências anteriores, por exemplo, os programas "Número ímpar ou par?" e "Calculando a média". No entanto, essa situação era diferente; tratava-se de um *software* educacional para ser utilizado pelo aluno com intenção de propiciar sua aprendizagem.

Os professores do grupo demonstraram certa preocupação em como ensinar o computador a avaliar a resposta dada e a dar um *feedback* para o aluno. A discussão entre eles girou em torno de como poderia ser programado esse *feedback*, caso o aluno não acertasse a resposta. O grande desafio era como ensinar o *software* a tratar o erro do aluno.

Nesse momento, os professores sugeriram algumas soluções de *feedback* por meio de mensagens, tais como:

> Quando o aluno acerta, o *sprite* pode dizer: "parabéns"; quando erra: "ainda não" [Registro de áudio de S02].
> "Se o aluno der a resposta correta, o *software* pode dar uma mensagem: Muito bem, você acertou! ou Parabéns, sua resposta está correta; e, se errada: Que pena, você errou! ou Tente novamente!" [Registro de áudio de S05].
> Nessa situação [em que o aluno erra a resposta], eu fico perdida sobre o que colocar" [Registro de áudio de S04].
> "Então, eu coloquei essa mensagem 'chame o professor', pensando justamente nessa parte de tratar o erro do aluno'" [Registro de áudio de S07].

A Figura 10 mostra a programação de uma das fases do *software* com esse tipo de *feedback*.

Figura 10 – Procedimento de uma das fases do *Software* Educacional

Fonte: Rocha (2015, p. 200)

Essa situação deixou os professores incomodados, pois, inicialmente, eles não conseguiram pensar em como tratar o erro do aluno para incluir na programação do *software*. Mesmo aqueles professores que não concordavam em dar um *feedback* do tipo "parabéns você acertou", ou "tente novamente", quando o aluno errasse, a sugestão dada foi "chame o professor".

Vale ressaltar que a formação continuada se pautou pelos princípios construcionistas. Portanto, desde o início, a intenção discutida com o grupo de professores era criar um *software* educacional que pudesse propiciar aprendizagem do aluno, ou seja, era ensinar o computador (programando) a ensinar o aluno a aprender. Esse modo de conceber o ensino é totalmente diferente de o professor, diante do erro do aluno, dizer "tente novamente".

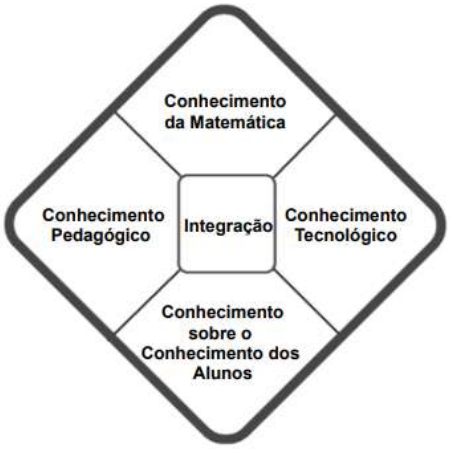
Essa situação foi bastante desafiadora para os professores, pois os deixou instigados a pensar sobre os processos de ensino e de aprendizagem. Interessante que o processo de construção do *software* educativo constituiu para o grupo um contexto rico de reflexão sobre a prática docente (ZEICHNER, 2008). Nesse momento, a intervenção da formadora-pesquisadora foi essencial para que o grupo pudesse explicitar as estratégias pedagógicas utilizadas no cotidiano das aulas de Matemática. O compartilhamento das práticas acompanhadas por reflexões e discussões evidenciou para o grupo de professores a necessidade de entender como aluno aprende.

De fato, a compreensão do professor sobre a dificuldade de aprendizagem do aluno, suas lacunas e equívocos conceituais vai ao encontro dos estudos de Ball, Thames e Phelps (2008), que ressaltam a importância de esse profissional apresentar competências e habilidades para lidar com o saber dos alunos e o saber da Matemática.

> Ainda nesta perspectiva, esses autores relatam que reconhecer uma resposta errada faz parte do conhecimento comum; em contrapartida, desenvolver familiaridade com erros comuns e decidir quais dos vários erros os alunos são mais propensos a fazer são exemplos de conhecimento de conteúdo e estudantes (CARVALHO; ANTONOW; SILVA, 2017, p. 83).

Isso significa que a prática docente, assim como a construção do *software* educacional, necessitam estar ancoradas em um conjunto integrado de conhecimentos, como o matemático, o tecnológico, o pedagógico e o do estudante, como ilustra a Figura 11:

Figura 11 – Integração dos conhecimentos para construção do *software* educacional



Fonte: Rocha (2015, p. 132)

Na Figura 11, o *conhecimento da Matemática* engloba o conhecimento do conteúdo especializado, o qual, na perspectiva de Ball, Thames e Phelps (2008), refere-se ao conteúdo matemático e às habilidades utilizadas unicamente para o ensino; o *conhecimento pedagógico* foca os objetivos para fins educacionais; o *conhecimento tecnológico* diz respeito ao conhecimento dos recursos tecnológicos que, nesse caso, é a linguagem de programação Scratch; e o *conhecimento sobre o conhecimento dos alunos* que está associado, diretamente, à compreensão do professor sobre o que o aluno sabe ou precisa saber para poder prever as reações do aluno diante do *software*.

No curso, os professores vivenciaram um modo de aprender baseado na perspectiva construcionista e essa experiência inspirou o grupo a desenvolver o *software* educacional. Durante a elaboração do *software*, os professores levantaram uma série de conjecturas sobre situações-problema que pudessem evidenciar as regularidades de sequências numéricas e, aos poucos, foram descrevendo seus algoritmos usando os comandos da linguagem, articulando conceitos matemáticos e computacionais. Nesse processo, o conhecimento do conteúdo matemático foi sendo realimentado pelo conhecimento da linguagem de programação do Scratch e vice-versa, retratando, por conseguinte, que o grupo de professores conseguiu reconstruir o conhecimento tecnológico do conteúdo.

A experiência do grupo de professores em programar um *software* demonstrou uma particularidade interessante pelo fato de ter desencadeado momentos reflexivos sobre a maneira de intervir em uma situação de erro do aluno. Isso aconteceu quando os professores antecipavam as possíveis reações dos alunos, à medida que construíam o *software* educativo. Essa antecipação das reações dos alunos fez com que os professores

revisitassem suas práticas no cotidiano da sala de aula. Nesse momento, a reflexão sobre a própria prática fez emergir no grupo a tomada de consciência acerca da necessidade de que o ato de ensinar requer também inteirar-se a respeito do conhecimento dos alunos, seja tanto para programar o *software* como na prática em sala de aula. Interessante perceber que o conhecimento tecnológico, especialmente a programação do *software* educacional, foi sendo realimentado pelo conhecimento pedagógico, acrescido pela percepção do conhecimento dos alunos e vice-versa.

O processo de integração de conhecimentos na perspectiva do TPACK não é simples de ocorrer. No entanto, existem estratégias que podem contribuir nesse processo, as quais devem contemplar momentos de apropriações, reflexões e vivências baseadas em situações que priorizem o aprender-fazendo, criando algo que seja significativo, tal como aconteceu com a atividade de elaboração do *software* educacional com o uso da linguagem Scratch.

CONSIDERAÇÕES FINAIS

Este estudo mostra que a atividade realizada pelos professores na construção de um *software* educacional com o uso da linguagem Scratch durante a formação revelou um elemento caracterizado por uma situação-problema que permitiu ao grupo aprofundar a reflexão acerca do conhecimento profissional docente na perspectiva do TPACK. Esse fato corrobora que a tecnologia utilizada no âmbito da educação e pautada pelos princípios construcionistas ganha um nova dimensão cognitiva vinculada a processos reflexivos que podem, como ocorreu na formação, propiciar a reconstrução de conhecimentos sobre a tecnologia, a pedagogia e o conteúdo curricular de forma integrada, o que resulta em um novo conhecimento sobre e para a prática do professor.

REFERÊNCIAS

ALMEIDA, M. E. B.; VALENTE, J. A. *Tecnologias e currículo*: trajetórias convergentes ou divergentes? São Paulo: Paulus, 2011.

BALL, D. L.; THAMES, M. H.; PHELPS, G. Content knowledge for teaching: what makes it special? *Journal of Teacher Education*, v. 59, n. 5, p. 389-407, Nov. 2008.

CARVALHO, M. P.; ANTONOW, L. M.; SILVA, J. F. O que pensam os futuros professores de Matemática sobre o programa de bolsas de iniciação à docência. *Com a Palavra o Professor*, Vitória da Conquista (BA), v. 2, n. 2, p. 77-91, jan./abr. 2017.

CIBOTTO, R. A. G.; OLIVEIRA, R. M. M. TPACK Conhecimento tecnológico e pedagógico do conteúdo: uma revisão teórica. *Imagens da Educação*, v. 7, n. 2, p. 11-23, 2017.

COLLING, J.; RICHIT, A. Aspectos transversais da articulação dos conhecimentos profissionais na formação inicial de professores de Matemática. *Jornal Internacional de Estudo em Educação Matemática – JIEEM*, v. 13, n. 1, p. 17-25, 2020.

KOEHLER, M. J.; MISHRA, P. What is technological pedagogical content knowledge? *Contemporary Issues in Technology and Teacher Education*, v. 9, n. 1, p. 60-70, 2009.

MISHRA, P.; KOEHLER, M. J. Technological pedagogical content knowledge: a framework for teacher knowledge. *Teachers College Record*, v. 108, n. 6, p. 1017-1054, 2006.

PAPERT, S.; HAREL, I. *Constructionism*. Norwood: Ablex Publishing, 1991.

PRADO, M. E. B. B.; LOBO DA COSTA, N. M. O papel da atividade de programação no processo de construção de conhecimentos para a docência. *Revista e-Curriculum*, PUC-São Paulo, v. 14, n. 3, p. 898-918, jul./set. 2016.

RESNICK, M. *et al*. Scratch: programming for all. *Communications of the ACM*, New York, v. 52, n. 11, p. 60-67, 2009.

ROCHA, A. K. de O. *A programação de computadores como meio para integrar diferentes conhecimentos*: uma experiência com professores de Matemática. 2015. Tese (Doutorado em Educação Matemática) – Universidade Anhanguera de São Paulo – UNIAN, São Paulo, 2015.

SHULMAN, L. S. Those who understand: knowledge growth in teaching. *Educational Researcher*, v. 15, n. 2, p. 4-14, 1986.

SHULMAN, L. S. Knowledge and teaching: foundations of the new reform. *Harvard Educational Review*, v. 57, n. 1, p. 1-22, 1987.

VALENTE, J. A. *A espiral da espiral de aprendizagem*: o processo de compreensão do papel das tecnologias de informação e comunicação na educação. 2005. Tese (Livre-docência) – Universidade Estadual de Campinas, Instituto de Artes, Campinas, 2005.

VECCHIA, R. D.; MALTEMPI, M. V.; BORBA, M. C. The construction of electronic games as an environment for Mathematics education. *In*: LOWRIE, T.; JORGENSEN (ZEVENBERGEN), R. (ed.). *Mathematics Education in the Digital Era*. New York: Springer, 2015, p. 55-70.

VIEIRA, E. R. *Grupo de estudos de professores e a apropriação de tecnologia digital no ensino de Geometria*: caminhos para o conhecimento profissional. 2013. Tese (Doutorado em Educação Matemática) – Universidade Anhanguera de São Paulo – UNIAN, São Paulo, 2013.

ZEICHNER, K. M. Uma análise crítica sobre a "reflexão" como conceito estruturante na formação docente. *Educação e Sociedade*, Campinas, v. 29, n. 103, p. 535-554, 2008.

Submetido em 19 de outubro de 2019.
Aprovado em 26 de janeiro de 2020.