Um Estudo de Caso Sobre Competências do Pensamento Computacional Estimuladas na Programação em Blocos no Code.Org

Ahemenson Fernandes Cavalcante, Leonardo dos Santos Costa, Ana Liz Souto Oliveira de Araújo

Departamento de Ciências Exatas – Universidade Federal da Paraíba
Rua da Mangueira, s/n, Companhia de Tecidos Rio Tinto
58.297-000 – Rio Tinto – PB - Brasil

{ahemenson, leonardo, analiz}@dcx.ufpb.br

Abstract. *This work presents a qualitative case study about computational thinking skills boosted in Code.org. The results show that the skills related to computational concepts, practices and perspectives modeled on a framework could be identified.*

Resumo. *Este trabalho apresenta um estudo de caso qualitativo sobre as competências do pensamento computacional exploradas em um curso de programação introdutória da plataforma Code.org. Os resultados demonstram quais as competências relacionadas aos conceitos, práticas e perspectivas computacionais modeladas em um framework puderam ser encontradas no curso.*

1. Introdução

A computação, de forma gradual, se faz presente a cada momento do nosso cotidiano. Assim, transformações na forma como usar o computador e as tecnologias envolvidas se tornam inevitáveis. Conforme afirma Wing (2006), futuros sociólogos, economistas, músicos, educadores deverão interagir com profissionais da Computação através de um pensamento interdisciplinar, ou seja, como recurso para solucionar problemas através da construção de novas abordagens computacionais.

O pensamento computacional se caracteriza como um processo com vista à resolução de problemas por meio de conceitos, recursos e ferramentas computacionais. O pensamento computacional baseia-se em fundamentos da computação, envolvendo a resolução de problemas, a capacidade de projetar sistemas e a compreensão do comportamento humano [Wing, 2006]. É uma forma de aplicar conceitos trabalhados na computação, mas que não são exclusivos somente desta área, pois podem ser aplicados na resolução de problemas dos mais variados campos do saber. A aplicabilidade transversal e multidisciplinar ressignifica o "pensar computacionalmente" como uma competência fundamental para todas as pessoas, não apenas para profissionais da computação, despontando como um requisito elementar para a formação básica dos profissionais de todas as áreas nos próximos anos.

O ensino de programação introdutória é apontado como uma abordagem para

DOI: 10.5753/cbie.wcbie.2016.1117

estimular o pensamento computacional no contexto escolar. Nos últimos anos foram criadas plataformas de ensino de programação que adotam uma estratégia composta por traços lúdicos, ambientação amigável e programação através de blocos. Scratch (https://scratch.mit.edu/), App Inventor (http://appinventor.mit.edu/), Alice (http://www.alice.org/index.php), Blockly Games (https://blockly-games.appspot.com) são exemplos dessas plataformas. Um outro projeto que vem se destacando é a plataforma de ensino de programação Code.org (http://www.code.org). O projeto tem como perspectiva trazer os conceitos básicos da Ciência da Computação, principalmente da programação, para o ambiente escolar.

Code.org possui as características de ser um ambiente gratuito, ter suporte ao idioma em português, ser direcionado ao público iniciante e ser disponível online. O diferencial frente as outras plataforma é possuir cursos com conteúdo metodológico pré-definido. Dessa forma, o aluno pode realizar os cursos de forma autônoma, sem depender, obrigatoriamente, da metodologia de aulas proposta por professores. Outro diferencial é possuir um ambiente voltado exclusivamente aos educadores. Esse módulo provê recursos de criação e gerenciamento de turmas e acompanhamento do progresso individual dos alunos nas atividades.

Desde o lançamento em 2013, mais de duzentos milhões de alunos utilizaram o Code.org (http://code.org). Desses alunos, 49% foi composto pelo sexo feminino, fomentando a participação de meninas na área da Ciência da Computação. Outro dado em destaque é a quantidade de professores cadastrados, 321.988 professores, o que novamente evidencia o uso da plataforma em atividades educacionais.

Apesar do grande número de usuários, são poucas as pesquisas nacionais sobre as vantagens do ensino de programação ofertada pela plataforma [Dantas e Costa, 2013]. Sobretudo, são escassos os trabalhos que apontam evidências sobre o estímulo do pensamento computacional por meio dos cursos da plataforma Code.org [Kalelioğlu, 2015] . Assim, desejamos responder a seguinte questão de pesquisa: quais competências do pensamento computacional são estimuladas por meio de cursos de introdução à programação ofertados pela plataforma Code.org?

Nesse contexto, este trabalho exploratório visa identificar e analisar competências do pensamento computacional estimuladas através de um curso de programação oferecido pela plataforma Code.org. Para atingir esse objetivo, selecionamos um *framework* que avalia competências do pensamento computacional no contexto de ambientes de programação em blocos e identificamos e analisamos seus elementos em um curso.

O presente trabalho está organizado da seguinte maneira: a Seção 2 apresenta brevemente competências do pensamento computacional; a Seção 3 descreve o *framework* que avalia competências do pensamento computacional no contexto de ambientes de programação em blocos; a Seção 4 cita trabalhos relacionados; a Seção 5 descreve os procedimentos adotados; a Seção 6 analisa os resultados; a Seção 7 conclui o trabalho e apresenta trabalhos futuros.

2. Competências do Pensamento Computacional

Competência é a capacidade de utilizar mais de um recurso para resolver algo de forma

inovadora, criativa e no momento necessário. Já a habilidade pode ser entendida como uma capacidade individual de resolver problemas utilizando conhecimentos previamente estabelecidos, na qual, a partir deles, podemos realizar induções e deduções com o objetivo de resolvê-los [Bonotto e Felicette, 2014]. Assim, competência consiste em um conceito mais abrangente no qual há um conjunto de habilidades e atitudes vinculadas na realização de uma ação. Por isso, usaremos o termo competências do pensamento computacional no restante do texto.

A *Computer Science Teacher Association*[1] (CSTA) e a *International Society for Technology in Education* (ISTE) organizaram uma definição operacional que auxilia na identificação de competências do pensamento computacional. Desta forma, o pensamento computacional poderá ser desenvolvido a partir de atividades de: Formular problemas de forma que nos permita usar um computador e outras ferramentas para ajudar a resolvê-los; Organizar e analisar dados logicamente; Representar dados através de abstrações tais como modelos e simulações; Propor soluções através do pensamento algorítmico; Identificar, analisar e implementar as soluções combinando recursos de forma eficiente e eficaz; Generalizar e transferir um processo de solução de um problema para outros.

Essa definição operacional é suportada e aprimorada por um conjunto de disposições ou atitudes que são essenciais para o pensamento computacional. Tais atitudes, conforme , incluem[1]: Confiança em lidar com a complexidade; Persistência ao trabalhar com problemas difíceis; Tolerância em lidar com ambiguidade; Capacidade de lidar com problemas em aberto; Capacidade de se comunicar e trabalhar em grupo para atingir um objetivo ou solução em comum.

3. Framework para identificar competências do pensamento computacional na programação introdutória

O *framework* proposto por Brennan e Resnick (2012) permite a compreensão das competências do pensamento computacional que podem ser exploradas em ambiente de programação introdutória como o Scratch. O *framework* está dividido em três elementos: conceitos computacionais, práticas computacionais e perspectivas computacionais. A Figura 1 resume as competências exploradas em cada parte.

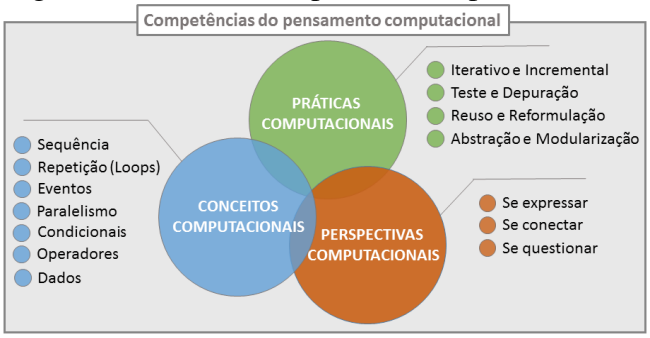

Figura 1. Framework de Brennan e Resnick (2012)

A primeira grande dimensão estabelecida no framework refere-se aos conceitos computacionais empregados na programação. **Sequência** expressa uma série de

1 https://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

procedimentos a serem realizados os quais são acionados passo a passo, uma linha por vez. **Loops** são estruturas de repetição que permitem que um conjunto de passos seja realizado repetidamente. **Eventos** são mecanismos programados para serem acionados a partir da ocorrência de uma situação especifica, assim quando um evento é despertado certas instruções de código serão disparadas. **Paralelismo** é empregado para executar sequência de instruções ao mesmo tempo, em paralelo, minimizando o tempo dispendido na realização de uma tarefa. **Condicionais** permitem estabelecer condições para execução de ações. **Operadores** pode ser compostos por estruturas da matemática, da lógica booleana e expressões em caracteres. **Dados** refere-se ao fato de armazenar, recuperar e atualizar valores que serão contidos em variáveis.

As práticas computacionais estão concentradas nos processos de pensar e de aprender durante as atividades de programação. **Iterativo e incremental** representa o processo sequencial no plano de planejamento e de execução para codificação. **Teste e depuração** envolvem etapas de análise e reflexão detalhada do código, para descobrir a raiz de um problema. **Reuso e reformulação** estão relacionados ao fato de utilizar trechos próprios de códigos desenvolvidos em trabalhos anteriores ou fazer uso de trechos de terceiros para ajustes e adaptações. **Abstração e modularização** estão relacionados à desconsiderar informações irrelevantes para focar apenas nas informações importantes e à dividir o problema em pequenas partes para auxiliar a propor a solução.

As perspectivas computacionais estão relacionadas as atitudes desempenhadas pelos alunos durante a programação. Essas atitudes são relativas ao modo de se expressar, se conectar com outros alunos e com a comunidade, se questionar e de agir colaborativamente. Mais detalhes podem ser visto em [Brennan e Resnick, 2012].

4. Trabalhos relacionados

Na literatura nacional, encontramos apenas um trabalho que usou a plataforma Code.org como objeto de estudo. Dantas e Costa (2013) relataram os benefícios das oficinas do Code.org à formação dos estudantes. Os autores evidenciaram a importância da plataforma por auxiliar no desenvolvimento de habilidades como: raciocínio lógico, trabalho em equipe, capacidade de resolver problemas e estímulo a criatividade.

Kalelioğlu (2015) investigou os efeitos que oficinas code.org provocaram na habilidade de resolução de problemas em estudantes com 10 anos de idade. Os resultados apontaram que as meninas tiveram maior impacto positivo na resolução de problema em comparação ao meninos. Já Israel *et al* (2015) investigou como professores da educação básica com pouca experiência em computação poderiam integrar o pensamento computacional em sua prática pedagógica. Os autores utilizaram oficinas do Code.org e outras ferramentas como instrumento para introduzir conceitos de programação e pensamento computacional aos professores.

5. Procedimentos e métodos

Iniciamos o estudo com uma pesquisa bibliográfica no intuito de identificar as habilidades e competências do pensamento computacional estimuladas pelos ambientes

de programação introdutória em blocos. Em seguida, pesquisamos modelos de avaliação de competências do pensamento computacional no contexto de programação introdutória em blocos.

Dentre os *frameworks* para avaliação do pensamento computacional, o de Brennan e Resnick (2012) apresentou-se como o mais abrangente para avaliação de competências do pensamento computacional em programação em blocos. Embora ele tenha sido concebido sob a perspectiva do ambiente Scratch, avaliamos sua aplicabilidade e limitações no contexto de um curso da plataforma Code.org. Além disso, não encontramos na literatura um framework adequado ao contexto dos cursos do Code.org.

5.1. Delimitação do escopo da pesquisa

Os cursos do Code.org são constituídos de uma série de etapas *online* (executadas na plataforma) e *offline* (atividades desplugadas). Atualmente estão disponíveis quatro cursos principais na página inicial da plataforma. O curso 1 é voltado aos alunos em fase da alfabetização. O curso 2 é dirigido aos alunos que já possuem a capacidade de interpretação de textos e estão cursando o ensino fundamental ou médio. Nos cursos 3 e 4 são aprofundados os conteúdos abordados no curso 2, sendo que o nível de complexidade dos problemas aumenta à medida em que o aluno avança.

Neste estudo de caso exploratório nos restringimos às etapas do curso 2. A justificativa para a escolha é a sua abrangência de conceitos de programação trabalhados e ter sido projetado para um público mais amplo composto por estudantes do ensino fundamental e médio sem experiência prévia em programação. Desconsideramos as etapas realizadas *offline* (atividades desplugadas) pelo fato de não utilizarem a plataforma para sua execução.

No curso 2 há 19 etapas, sendo 12 constituídas de histórias, caracterizadas pelos desafios propostos aos alunos. Essas etapas são constituídas de elementos familiares ao seu público-alvo. Na primeira etapa, o cenário é do jogo *Angry Bird*, no qual o personagem do pássaro vermelho precisa destruir o porco. Na etapa 6 estão presentes os cenários e personagens do jogo mobile *Plants vs. Zombies*, no qual o aluno deve orientar o zombie a percorrer o caminho até encontrar e devorar a planta. Outro personagem que aparece é o pássaro do jogo *Flappy Bird*, cujo os desafios propostos são bastantes similares ao do jogo. Há também personagens especialmente criados para motivar os estudantes na solução dos desafios, como as etapas em que é preciso fazer a abelha Bee colher o néctar das flores do cenário e produzir o seu mel.

6. Análise das competências do pensamento computacional no Code.org

Discutimos nesta seção as competências do pensamento computacional no curso 2 da plataforma Code.org sob a ótica do framework de Brennan e Resnick (2012).

6.1. Conceitos computacionais no Code.org

Em todas as etapas da plataforma a habilidade de elaborar uma **sequencia** de passos para a resolução de uma tarefa é requerida aos alunos. A Figura 2(a) ilustra a etapa do Angry Bird. Ao longo de todas as demais etapas, o modo de resolução segue o

mesmo princípio, variando *(i)* no aumento da distância percorrida entre os personagens, *(ii)* na maior presença de obstáculos e *(iii)* no acesso a novos blocos representativos (estruturas de programação).

A partir da etapa 6 (Labirinto: Ciclos) o conceito de **repetição** é explorado. Os desafios são construídos de maneira a fazer o aluno identificar, no problema proposto, a existência de ações repetitivas. Nesse contexto, o aluno é estimulado a encontrar o padrão de atividades que se repetem do personagem. O emprego desse conceito está relacionando a prática de reconhecimento de padrões (não citado no framework).

O uso de **eventos** como mecanismo auxiliar na resolução de problemas está presente a partir da etapa 16 (Flappy Bird). Neste momento do curso, o aluno deverá construir a solução empregando estruturas conceituais que denotam a ocorrência de eventos e as suas respectivas ações. A ocorrência do evento põe em ação seus blocos associados, os quais podem variar em quantidade e ações. A Figura 2(b) apresenta ações relacionadas a ocorrência de eventos na etapa 16 Flappy Bird.

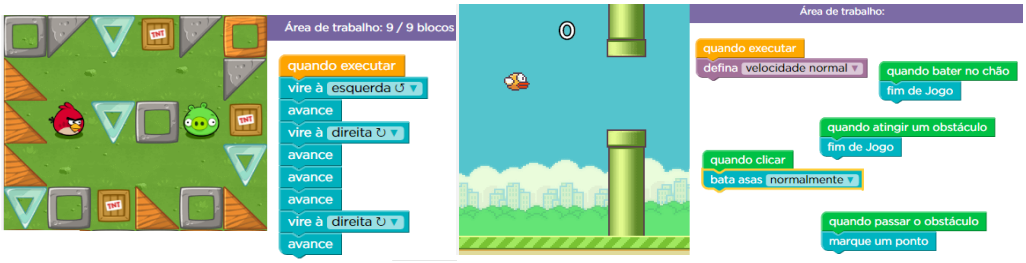

Figura 2. (a) Sequência na etapa Angry Bird; (b) Eventos na etapa Flappy Bird

A utilização em conjunto de eventos pode ser considerada aqui como utilização do conceito de **paralelismo**, pois o aluno percebe a necessidade de estabelecer mecanismo múltiplos para solucionar o problema numa dinâmica em paralelo.

Figura 3. Uso de estruturas condicionais e de repetição

A **estrutura condicional** é exposta a partir da 13° etapa (Abelhas Condicionais) na Figura 3. O estudante é provocado a refletir acerca de circunstâncias envolvidas na resolução do problema. As etapas anteriores se caracterizavam pela construção linear do algoritmo, desconsiderando reações adversas. A partir de agora, considera-se situações as quais não temos controle e não podemos prever o momento do seu acontecimento, mas podemos estabelecer condições que tratam a ocorrência dessas situações, possibilitando chegar aos resultados pretendidos.

Operadores e dados estão limitados a exibição de caracteres, não sendo encontrados estrutura matemática e lógica booleana. A presença de variáveis pôde ser identificada em quase todas as etapas. Porém, a plataforma não fornece explicação ao aluno sobre seu significado e utilização, diferentemente do que ocorre com outros conceitos computacionais encontrados. Além disso, a presença deles se caracteriza por não conceder ao aluno total liberdade de uso, cabendo apenas a utilização de opções pré-determinadas, conforme ilustrado na Figura 4.

Figura 4. Uso de dados

6.2. Práticas computacionais no Code.org

As práticas computacionais **iterativo e incremental** se fazem presentes por meio da abordagem de resolução dos desafios. Quando o aluno se depara com um novo desafio, ainda em nível cognitivo, ele busca pensar a cada passo, a cada posição, qual bloco utilizar. Desta forma, o aluno constrói a solução, seguindo passos sucessivos que o fazem se aproximar gradativamente da solução idealizada. À medida em que realiza a seleção e o encaixe dos blocos e constata que ainda não concluiu o desafio, ou que há erros na resolução, novamente inicia-se o processo. As aplicações dessas práticas seguem durante todo o curso. Sendo assim, o exercício contínuo fortalece essas habilidades no aluno que, em outros contextos, poderá adotar abordagens semelhantes para a resolução de problemas.

As práticas de **teste e depuração** na resolução é uma atividade presente em todas as etapas do curso. Quando a implementação do código está concluída, o aluno verifica a solução através do botão executar. Ao clicá-lo, a animação do código é exibida. Caso a solução esteja correta, o aluno conquista o desafio e passa para o próximo. Caso contrário, é exibida mensagem de erro, como da Figura 5a. Assim, cabe ao aluno identificar o erro e corrigi-lo. Para estimular a atividade de depuração, a etapa 11 (Artista) é dedicada exclusivamente a depurar código e reparar os erros (Figura 5b).

Reuso e reformulação no sentido em que o framework trata não foram identificados no curso 2. Não é possível reutilizar trechos de códigos anteriores para aplicá-los em uma outra etapa. A cada etapa, os blocos têm que ser inseridos um a um.

Entretanto, apesar da plataforma Code.org não permitir copiar e colar códigos de desafios anteriores, em determinadas etapas, os desafios se repetem permitindo o **reuso** de soluções. Isso possibilita ao aluno reutilizar a mesma abordagem mental proposta no desafio anterior. Em outras palavras, o aluno recorre a informações sobre os passos

necessários que permitiram a solução do desafio anterior. A partir deles, o aluno planeja a solução do novo desafio. Portanto, para desafios semelhantes reutiliza-se estratégias semelhantes para solução.

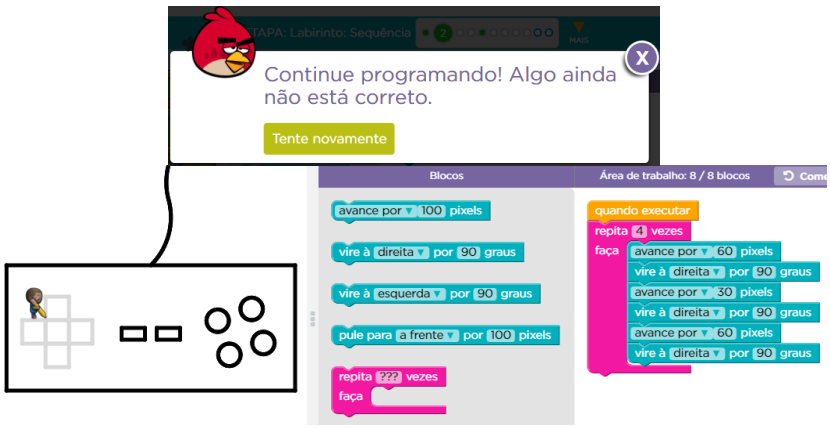

Figura 5. (a) Mensagem de erro na implementação do aluno (b) O objetivo na etapa 17 é depurar os códigos apresentados

A prática de **reformulação** está presente no momento em que o aluno se depara com desafios que assemelham-se à problemas anteriores, possibilitando o reuso de estratégias, mas que é necessário adaptações para o problema atual. Uma vez que em desafios consecutivos, o aluno se depara com problemas semelhantes, ele precisa abstrair e reorganizar, ou seja, o aluno precisa mensurar o que continua semelhante da estratégia anterior e o que é inédito na etapa atual, e então implementar a solução.

As práticas de **abstração** e **modularização** estão presentes quando o aluno, em suas abordagens, concentra-se em partes essenciais do desafio ou divide-o em partes menores. Primeiramente por meio da observação geral do problema do desafio, o aluno cria pequenas metas que somadas permitem alcançar o resultado pretendido. Dessa forma, ele vai construindo seu algoritmo atento somente ao alcance da meta específica. Uma vez tendo sido alcançada, toma a próxima como meta principal, e assim sucessivamente até alcançar a conclusão definitiva. Portanto, a abstração e modularização são práticas recorrentes para superar os desafios ao longo do curso.

6.3. As perspectivas computacionais no Code.org

As perspectivas computacionais na dimensão proposta no framework não foram verificadas neste estudo. Para analisar as atitudes relativas ao modo de se expressar, se conectar com outros alunos e com a comunidade, se questionar e o modo de agir colaborativamente seria necessário realizar análises que vão além do estudo teórico da plataforma Code.org. Isso implicaria em observações, entrevistas ou outros métodos de coletar dados junto aos alunos durante a execução do curso, o que está fora do escopo atual da pesquisa.

6.4. Outras competências

Observamos que em todos os desafios o aluno é sempre estimulado a construir

suas soluções de maneira eficiente. **Eficiência** nesse contexto está relacionado ao uso da quantidade mínima de blocos para resolução do problema. A Figura 6 (parte superior) mostra que o aluno é informado quanto a eficiência de sua abordagem (limite de blocos para atingir a solução do desafio). No lado (a) da Figura 6 temos um exemplo de uso eficiente. Já no lado (b) temos um código não eficiente, onde há uma indicação de superação do limite de blocos. Na parte (c) mostra a notificação solicitando a utilização de menos blocos. Isso não impede que o aluno prossiga para os próximos desafios, mas notifica-o que ele poderia ter usado menos blocos. Essas notificações estimulam funções cognitivas associadas ao planejamento e execução de estratégia de solução, podendo levar o aluno a raciocinar de forma a prever as consequências de sua estratégia.

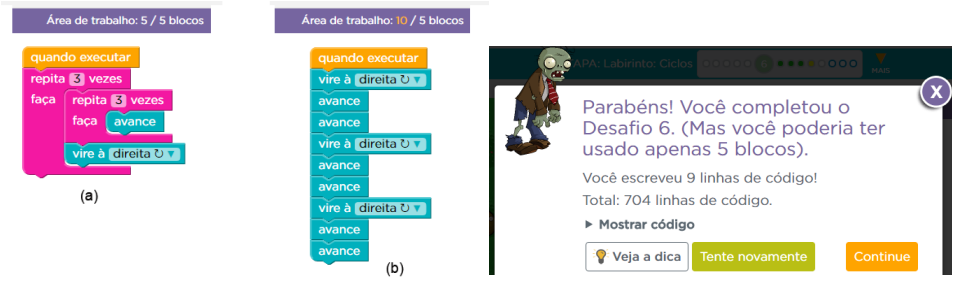

Figura 6. Indicação dos limites de blocos de cada desafio

O **reconhecimento de padrões** é uma prática não descrita no framework, mas que identificamos como prática relacionada ao emprego do conceito de repetição. O reconhecimento da existência de passos repetitivos é o ponto de partida para a utilização do conceito de repetição na implementação. A Figura 7 mostra dois exemplos em etapas distintas de desafios no qual primeiramente é preciso identificar o padrão de repetição dos passos do personagem e posteriormente adotar estruturas de repetição no código.

Figura 7. O reconhecimento de padrões precede o uso do bloco de repetição

7. Conclusão

Este trabalho apresentou um estudo exploratório sobre as competências do pensamento computacional estimuladas em um curso de programação introdutória da plataforma Code.org. Os resultados demonstram que as competências modeladas no *framework* de Brennan e Resnick (2012) puderam ser aplicadas no curso, com possibilidade de expansão nos quesitos de eficiência e reconhecimento de padrões.

Os conceitos computacionais puderam ser identificados ao longo das etapas do curso. Todavia, não localizamos operadores de estruturas matemáticas e de lógica booleana. As práticas computacionais foram identificadas, dando destaque ao reuso e

modularização. Essas práticas estão relacionadas à utilização de estratégias de resolução semelhantes para problemas semelhantes, adaptando-as quando necessário. Já as perspectivas computacionais não foram observadas, pois necessita de dados coletados durante a execução do curso, fugindo do escopo inicial da pesquisa.

Destacamos que o estímulo à eficiência (utilizar a menor quantidade de blocos necessários para solução) é uma prática presente no curso. Isso contribui para que o aluno planeje melhor a construção de sua abordagem de resolução do problema. O reconhecimento de padrões é outra prática estimulada no curso durante a introdução do conceito de repetição. Ambas as práticas não são destacadas no *framework*. Assim, elas poderão ser incorporadas em outro modelo de avaliação de pensamento computacional em cursos de programação introdutória baseados em blocos.

Em trabalhos futuros, planejamos analisar outros cursos da plataforma, sob outras perspectivas do pensamento computacional e utilizando outros *frameworks* como modelo. Em outra vertente, também planejamos aplicar e coletar dados de alunos durante a execução do curso, em busca de evidências empíricas das perspectivas computacionais, bem como identificar outras habilidades específicas que são estimuladas no desenvolvimento do curso.

8. Agradecimentos

Os autores agradecem à CAPES pela concessão de bolsas do Programa Institucional de Bolsa de Iniciação à Docência - PIBID no período de realização deste trabalho.

Referências

Bonotto, G.; Felicetti, V. L. (2014). Habilidades e competências na prática docente: perspectivas a partir de situações-problema. In: Educação Por Escrito, 5(1), 17-29.

Dantas, R. F.; Costa, F. E. A. (2013). CODE: O ensino de linguagens de programação educativas como ferramentas de ensino/aprendizagem. In: Simpósio Hipertexto e Tecnologias na Educação, vol. 5, Recife.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In: Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada, p. 1-25.

ISTE – International Society for Technology in Education; (2011) National Science Foundation. Computational thinking: leadership toolkit. First Edition.

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. Computers & Education, 82, 263-279.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. Org. Computers in Human Behavior, 52, 200-210.

Wing. (2006). Computational Thinking. Communications of the ACM,v.49, n.3, p. 33-35.