INTEGRAÇÃO DO PENSAMENTO COMPUTACIONAL NO CURRÍCULO DA EDUCAÇÃO BÁSICA: DIFERENTES ESTRATÉGIAS USADAS E QUESTÕES DE FORMAÇÃO DE PROFESSORES E AVALIAÇÃO DO ALUNO

VALENTE, José Armando*

RESUMO

A maneira como as tecnologias digitais estão sendo trabalhadas nas escolas, em praticamente todos os países, não tem contribuído para o desenvolvimento do pensamento computacional. Essas atividades estão restritas ao uso do que foi chamado de software de escritório, como o processador de texto, a planilha e, com isso, não exploram conceitos da Ciência da Computação, permitindo usar o computador como um instrumento de pensar com e pensar sobre o pensar. Isso tem levado alguns países a alterarem o currículo da Educação Básica. O presente artigo tem como objetivo analisar as diferentes estratégias de implantação do pensamento computacional no currículo da educação básica, bem como as questões relacionadas à formação de professores e à avaliação. Essa análise permitiu classificar as estratégias em três grandes categorias: a inclusão de temas da Ciência da Computação; a inserção de disciplinas que exploram conceitos do pensamento computacional por meio do uso de tecnologias em diferentes atividades, como jogos e robótica; e a exploração dos conceitos do pensamento computacional de maneira transversal aos assuntos curriculares. Além disso, entender como está sendo a formação de professores para o desenvolvimento dessas atividades e como avaliar o desenvolvimento do pensamento computacional no aluno.

Palavras-chave: Tecnologias educacionais. Ciência da Computação. Estratégias de ensino e aprendizagem. Fazer computacional.

* Livre Docente pela Universidade Estadual de Campinas (UNICAMP). Mestre e Doutor pelo Massachusetts Institute of Technology (MIT). Professor Titular do Departamento de Multimeios, Mídia e Comunicação, Instituto de Artes, e Pesquisador do Núcleo de Informática Aplicada à Educação (NIED) da UNICAMP. Professor Colaborador do Programa de Pós-Graduação em Educação: Currículo, da Pontifícia Universidade Católica de São Paulo (PUC-SP). Coordenador do Grupo Gestor de Tecnologias Educacionais (GGTE) da UNICAMP. Email: jvalente@unicamp.br

PUC-SP
e-Curriculum
Programa de Pós-Graduação em Educação: Currículo

INTEGRATION OF COMPUTATIONAL THINKING IN THE K-12 CURRICULUM: DIFFERENT STRATEGIES USED AND QUESTIONS RELATED TO TEACHER TRAINING AND STUDENT'S ASSESSMENT

VALENTE, José Armando*

ABSTRACT

The way in which digital technologies are used by schools in practically every country of the world, has rarely contributed to students' development of computational thinking. Activities were almost always restricted to the use of what was called the office software, such as word processor or spreadsheets and, therefore, did not explore computer science concepts, concepts which allow the students to use the computer as a tool to think with and to think about thinking. This has led some countries to change K-12 curricula. This article aims to analyze the different strategies used to exploit computational thinking in K-12 curriculum as well as questions related to teacher training and evaluation. This analysis allowed to classify these strategies into three broad categories: inclusion of Computer Science activities; inclusion of curriculum subjects that exploit computational thinking through the use of technologies for the development of activities, such as games, robotics; and the exploration of the concepts of computational thinking transversally with the curriculum subjects. Also it is possible to understand the professional development required for teachers to develop these activities and how to evaluate students' development of computational thinking.

Keywords: *Educational technologies. Computer Science. Teaching and learning strategies. Computational doing.*

* *Habilitation from the State University of Campinas (UNICAMP). MSc and PhD from the Massachusetts Institute of Technology(MIT). Full Professor in the Multimedia, Media and Communication Department, Arts Institute, and Researcher of the Nucleus of Informatics Applied to Education (NIED) at UNICAMP. Collaborating Professor in the Graduate Program in Education: Curriculum at Pontifical Catholic University of São Paulo (PUC-SP). Coordinator of the Management Group on Educational Technologies (GGTE) at UNICAMP.jvalente@unicamp.br*

1 INTRODUÇÃO

A programação de computadores foi, no início dos anos 1980, uma das principais atividades relativas à aplicação da informática na educação. Isso foi possível graças ao uso da linguagem Logo, criada por Seymour Papert (1980).

Com o advento dos computadores pessoais (*personal computer* - *PC*) no final dos anos 1980 e do *software* de escritório (editor de texto e de imagem, planilha eletrônica, sistemas de autoria), a programação foi praticamente esquecida na Educação Básica. Embora esses novos recursos tenham contribuído para ampliar o leque de possibilidades de uso das tecnologias digitais na educação, elas não foram trabalhadas no sentido de estimular o desenvolvimento do pensamento lógico dos aprendizes, nem contribuíram para a compreensão das especificidades do funcionamento dessas tecnologias e dos conceitos computacionais trabalhados por meio do uso desses *software*. No entanto, esses conhecimentos passaram a ser fundamentais no contexto da cultura digital, e para viver e atuar na sociedade do conhecimento.

É inegável que a presença das tecnologias digitais de informação e comunicação (TDIC) tem provocado transformações importantes na organização econômica, social e cultural. Isso pode ser observado desde a maneira como interagimos socialmente, como acessamos a informação, como procedemos nas transações comerciais, e nas interações sociais. Diversos segmentos da sociedade já estão inseridos na cultura digital.

Porém, como já afirmou Buckingham (2007), na maior parte do tempo estamos utilizando um potencial muito limitado dessas tecnologias, uma vez que nos restringimos ao uso de "*software* de escritório" como os processadores de texto e as planilhas. Ainda estamos na fase de entender e explorar essas tecnologias como se fossem sofisticadas máquinas de escrever, de acessar a informação e de se comunicar.

Certamente as TDIC podem oferecer muito mais. Para tanto, será necessário aprofundarmos nossa concepção sobre essas tecnologias, entendendo como elas funcionam, como elas podem ser adaptadas aos diferentes contextos e situações de nosso dia-a-dia e, com isso, podermos usufruir dos verdadeiros benefícios da cultura digital. Essa preocupação tem levado elaboradores de políticas educacionais a enfatizar a importância da programação e de conceitos oriundos da Ciência da Computação para todos, como foi proposto pela Casa

Branca (EUA) lançando em 2013 a página *Computer Science is for Everyone!* (WHITE HOUSE, 2013); ou pesquisadores como Rushkoff (2011) que propõe em seu livro que programemos ou sejamos programados.

A ênfase nos conceitos da Ciência da Computação tem sido justificada com base no argumento que atividades realizadas no âmbito dessa ciência desenvolvem habilidades do pensamento crítico e computacional, e permitem entender como criar com as tecnologias digitais, e não simplesmente utilizá-las como máquinas de escritório. Esses conhecimentos são considerados fundamentais para preparar as pessoas para o século 21, independentemente da área de estudo ou ocupação que irão desenvolver (CODE, 2016).

Essa argumentação tem provocado mudanças no currículo de diversos países, nos quais a programação ou a Ciência da Computação está sendo introduzida, inclusive nos primeiros anos da Educação Básica. Por exemplo, na Europa, a Comissão Europeia publicou o relatório *European Schoolnet* (2014) baseado no estudo da situação atual em 20 países europeus, sendo que em 13 desses países a programação já faz parte de disciplinas obrigatórias do ensino infantil ao nono ano (K-9, equivalente ao nosso Ensino Fundamental), como na Estônia e Grécia. A Inglaterra alterou a disciplina obrigatória de Informática (denominada *ICT*), que explorava as ferramentas de escritório, substituindo-a pela *Computing*, estruturada no tripé: Ciência da Computação, Tecnologia da Informação e Letramento Digital (UK DEPARTMENT FOR EDUCATION, 2013).

Na maior parte das propostas implantadas ou dos estudos realizados a ideia é reavivar a programação por meio de atividades como *coding computer science ou computer programming*, objetivando a criação de condições para o desenvolvimento do pensamento computacional. Outros países têm uma preocupação muito mais ampla do que simplesmente aprender a programar e estão buscando novas maneiras de explorar os conceitos computacionais no sentido de criar condições para o desenvolvimento do pensamento computacional.

As pesquisas relativas ao pensamento computacional encontradas na literatura podem ser divididas em praticamente três grandes blocos: a natureza do pensamento computacional e como ele pode ser avaliado (como identificar o pensamento computacional no aprendiz); a formação de educadores para desenvolverem atividades que exploram os conceitos do pensamento computacional, especialmente integrados às atividades curriculares; e a implantação na escola de atividades que exploram o pensamento computacional e os

benefícios que essas atividades produzem. Obviamente, essa classificação tem um caráter puramente didático, uma vez que os conteúdos desses três blocos estão intimamente relacionados.

A análise dos documentos, artigos e das políticas sobre implantação de tecnologias na educação adotada por diferentes países permite identificar diferentes estratégias, que podem ser classificadas em termos de três grandes categorias: a inclusão de assuntos da Ciência da Computação, especialmente a programação, sendo subdividida em duas outras subcategorias: a) programação fora da sala de aula e b) a inserção de disciplinas no currículo que usam tecnologias para explorar temas relativos à *computer literacy* ou letramento digital; a inclusão de disciplinas no currículo que exploram conceitos do pensamento computacional por meio do desenvolvimento de diferentes atividades, como jogos, robótica; e a exploração dos conceitos do pensamento computacional de maneira transversal, no desenvolvimento de atividades, usando as tecnologias em diferentes disciplinas do currículo. Além da inclusão de atividades computacionais no currículo, diversos trabalhos abordam a questão da preparação de professores para desenvolver essas atividades, e como avaliar o aluno com relação ao desenvolvimento do pensamento computacional.

O objetivo deste artigo é descrever essas diferentes estratégias e como elas podem contribuir para que os aprendizes desenvolvam conceitos que estão sendo relacionados ao pensamento computacional, a formação de professores e a avaliação dos alunos. Inicialmente, são apresentados os tópicos sobre o que está sendo entendido por pensamento computacional, em seguida, as diferentes atividades que podem contribuir para o desenvolvimento do pensamento computacional, a integração do pensamento computacional no currículo da Educação Básica, e finalmente as questões relacionadas à formação de professores e à avaliação dos alunos.

2 O QUE É O PENSAMENTO COMPUTACIONAL

A ideia que a programação de computadores ajuda a pensar melhor não é nova. Desde que a linguagem Logo foi criada em meados dos anos 1960, Papert já mencionava a importância dessa atividade para o processo de construção de conhecimento e para o desenvolvimento do pensamento. Em 1971, ele argumentou que a computação pode ter "um impacto profundo por concretizar e elucidar muitos conceitos anteriormente sutis em

psicologia, linguística, biologia, e os fundamentos da lógica e da matemática" (PAPERT, 1971, p. 2). Isso é possível pelo fato de proporcionar a uma criança a capacidade "de articular o trabalho de sua própria mente e, particularmente, a interação entre ela e a realidade no decurso da aprendizagem e do pensamento" (p. 3).

A possibilidade de a computação auxiliar a criança a pensar melhor ficou clara no livro *Mindstorms* (PAPERT, 1980) no qual Papert propõe que a programação Logo pode estimular o que ele chamou de "*Powerful ideas*" e "*Procedural knowledge*". Para esse pesquisador, os computadores deveriam ser utilizados para que as pessoas pudessem "pensar com" as máquinas e "pensar sobre" o próprio pensar.

O termo "pensamento computacional" ou *computational thinking* veio à tona com o artigo de Jeannette M. Wing, em 2006, no qual ela afirma que o "pensamento computacional se baseia no poder e nos limites de processos de computação, quer eles sejam executados por um ser humano ou por uma máquina" (WING, 2006, p. 33). A autora afirma que pensamento computacional é uma habilidade fundamental para todos, não apenas para cientistas da computação. Segundo essa autora, à leitura, escrita e aritmética, é preciso acrescentar o pensamento computacional à capacidade analítica de cada criança (WING, 2006). Em 2011, Wing forneceu uma definição mais concreta, porém sem acrescentar substância à definição anterior (WING, 2011).

Encontrar uma definição para o pensamento computacional que todos concordam tem sido difícil para a comunidade da Ciência da Computação e mesmo para pesquisadores e organizações interessadas nesse tema. Os resultados de dois *workshops* sobre o âmbito e a natureza do pensamento computacional, patrocinados pela *National Academy of Sciences* dos Estados Unidos da América, respectivamente em 2009 e 2011, envolvendo pesquisadores de diversas áreas, concluíram que:

> Os debates realizados no *workshop* de fevereiro 2009 não chegaram a um acordo geral entre os participantes sobre o conteúdo preciso de pensamento computacional, e muito menos a sua estrutura. No entanto, a falta de desacordo explícito sobre seus membros poderia ser entendida como refletindo uma intuição compartilhada entre os participantes do *workshop* que o pensamento computacional, como um modo de pensamento, tem o seu próprio caráter distintivo (USA NATIONAL RESEARCH COUNCIL, 2010, p. 65).

> Embora os participantes do *workshop* não tenham concordado explicitamente sobre a definição de pensamento computacional, os exemplos que eles forneceram durante o *workshop* são valiosos como indicadores das maneiras como as pessoas veem a intersecção da computação, conhecimento disciplinar e algoritmos (USA NATIONAL RESEARCH COUNCIL, 2011, p. 5).

A tentativa de identificar conceitos e operacionalizar o pensamento computacional foi realizada por duas organizações, a *International Society for Technology in Education* (ISTE) e a *American Computer Science Teachers Association* (CSTA), que trabalharam com pesquisadores da Ciência da Computação e das áreas de Humanas e propuseram uma definição para o pensamento computacional que pudesse nortear as atividades realizadas na Educação Básica (K-12). Eles identificaram nove conceitos: coleta de dados, análise de dados, representação de dados, decomposição de problema, abstração, algoritmos, automação, paralelização e simulação. Enfatizaram que as habilidades relativas a esses conceitos não estão limitadas aos sujeitos da Ciência da Computação ou das áreas de Ciências, Tecnologia, Engenharia e Matemática (STEM), mas podem ser praticadas e desenvolvidas no âmbito de todas as disciplinas.

O grupo ISTE/CSTA desenvolveu também uma definição operacional para o pensamento computacional como um processo de resolução de problema, com as seguintes características: formulação de problemas de uma forma que permita usar um computador e outras ferramentas para ajudar a resolvê-los; organização lógica e análise de dados; representação de dados através de abstrações como modelos e simulações; automação de soluções através do pensamento algorítmico (a série de passos ordenados); identificação, análise e implementação de soluções possíveis com o objetivo de alcançar a mais eficiente e efetiva combinação de etapas e recursos; e generalização e transferência desse processo de resolução de problemas para uma ampla variedade de problemas.

Finalmente, observam que essas habilidades são “apoiadas e reforçadas por uma série de disposições ou atitudes que são dimensões essenciais do pensamento computacional" tais como “confiança em lidar com a complexidade, persistência em trabalhar com problemas difíceis, tolerância para a ambiguidade e capacidade de lidar com problemas abertos" (ISTE/CSTA, 2011, p. 7).

Diversos autores têm procurado identificar os componentes do pensamento computacional. Por exemplo, Zapata-Ros (2015) define 14 componentes, como análise

ascendente, heurística, pensamento divergente, criatividade, resolução de problema, pensamento abstrato, interação, recursividade, métodos colaborativos, metacognição. A questão com a identificação desses componentes é que parte deles está relacionada com o processo de pensar e de resolver problemas usando ou não as tecnologias. No entanto, quando utilizamos as tecnologias digitais elas adicionam possibilidades que permitem abordar problemas e situações que não poderiam ser enfrentados sem elas. Além disso, como é enfatizado na literatura, o trabalho com essas tecnologias tem um caráter distinto e, nesse sentido, é importante distinguir o que é parte do pensamento em geral e o que é específico sobre o pensamento computacional.

Por outro lado, mesmo no trabalho usando as tecnologias é importante distinguir o que elas possibilitam em termos do desenvolvimento de conceitos relativos ao problema sendo resolvido e o desenvolvimento de conceitos relativos ao pensamento computacional. Por exemplo, a simulação de uma reação química usando um *software* possibilita o desenvolvimento de conceitos relativos à química, bem como, dependendo do que é feito com esse *software* e a maneira como o aprendiz explora as características das tecnologias, essas atividades podem estar contribuindo para o desenvolvimento de conceitos relativos ao pensamento computacional.

O trabalho realizado no Núcleo de Informática Aplicada à Educação (NIED) da UNICAMP com a utilização da linguagem Logo permitiu entender que o processo de criação de um programa para a resolução de um problema acontece por intermédio de um ciclo de ações descrição-execução-reflexão-depuração (VALENTE, 1993; 1999).

O desenvolvimento de um programa se inicia com uma ideia de como resolver o problema. Essa ideia é passada para o computador na forma de uma sequência de comandos do Logo, ou seja, essa ação implica na **descrição** da solução do problema usando comandos do Logo. O computador, por sua vez, promove a **execução** desses comandos, produzindo um resultado. O aluno, baseado no resultado obtido pode realizar a ação de **reflexão** sobre o que ele obteve e o que intencionava, acarretando diversos níveis de abstração: abstração empírica, abstração pseudo-empírica e abstração reflexionante (PIAGET, 1995; MANTOAN, 1994). Essa reflexão pode acarretar uma das seguintes ações alternativas: ou o aluno não modifica o programa porque as suas ideias iniciais sobre a resolução daquele problema correspondem aos resultados apresentados pelo computador e, portanto, o problema está resolvido; ou depura o programa quando o resultado é diferente da sua intenção original. A **depuração** pode ser em

termos de alguma convenção da linguagem de programação, sobre um conceito envolvido no problema em questão, ou ainda sobre estratégias sobre como usar o conceito ou sobre como explorar os recursos tecnológicos. A depuração implica uma nova descrição e, assim, sucessivamente, ou seja, descrição-execução-reflexão-depuração-novadescrição.

As ações do ciclo não foram caracterizadas como conceitos do "pensamento computacional", porém elas têm sido úteis para explicitar as atividades que o aprendiz realiza na interação com as tecnologias digitais e ajudam a entender como a interação com as tecnologias digitais contribuem para o desenvolvimento do pensamento computacional.

Primeiro, o ciclo tem sido útil para explicar o processo de construção de conhecimento que acontece na interação com o computador. As ações podem ser cíclicas e repetitivas, mas a cada realização de um ciclo, as construções são sempre crescentes. Mesmo errando e não atingindo um resultado de sucesso, o aprendiz está obtendo informações que são úteis na construção do seu conhecimento. Na verdade, terminado um ciclo, o pensamento do aprendiz nunca é exatamente igual ao que se encontrava no início da realização desse ciclo. Assim, a ideia mais adequada para explicar o processo mental dessa aprendizagem é a de uma espiral, ou seja, uma espiral de aprendizagem (VALENTE, 2005). No entanto, é importante enfatizar que essa construção está acontecendo com relação aos conceitos envolvidos no problema sendo resolvido, bem como sobre a exploração dos recursos tecnológicos, ou seja, atividades que devem estar contribuindo para o desenvolvimento do pensamento computacional.

Segundo, se o ciclo de ações contribui para o processo de construção de conhecimento, cada uma das ações tem componentes relevantes para a construção do pensamento computacional. Não é objeto desse artigo, mas seria importante entender como cada uma dessas ações contribui para o desenvolvimento do pensamento computacional.

Finalmente, as ações identificadas no caso da programação usando a linguagem Logo extrapolam as atividades de programação e têm sido úteis para entender o que acontece com o uso de outros *software* como, por exemplo, o processador de texto, a planilha, o *software* de autoria, os *software* educacionais e as atividades de educação a distância usando plataformas *online* (VALENTE 2005). Nesses casos, dependendo do *software* sendo utilizado a descrição pode variar. Ela é uma série de comandos para a programação; um texto juntamente com comandos de formatação, no caso do processador de texto; ou mesmo um clique do *mouse* no

caso de um *software* que permite a seleção de atividades. O mesmo acontece com a execução que pode ser do conjunto de comandos da linguagem de programação, produzindo um resultado bem específico; ou a execução da formatação do texto (e nunca do conteúdo do texto em si) no caso dos processadores de texto. As reflexões e depurações também devem variar de acordo com os resultados produzidos pelo computador, podendo ser mais profundas, provocando mudanças conceituais ou pequenas alterações na atividade sendo realizada.

Mais recentemente, o ciclo de ações e a espiral de aprendizagem foram utilizados para entender como a construção de conhecimento pode acontecer na produção de narrativas digitais (ALMEIDA; VALENTE, 2012). Isso significa que o pensamento computacional pode ser explorado por meio de outras atividades educacionais.

3 EXEMPLOS DE COMO O PENSAMENTO COMPUTACIONAL PODE SER EXPLORADO NA EDUCAÇÃO

O relatório do *workshop* produzido pelo *National Research Council* em 2011 (USA NATIONAL RESEARCH COUNCIL, 2011) descreve diversos contextos nos quais o pensamento computacional pode ser trabalhado, como nas atividades diárias, nos *games* e na gamificação, no jornalismo, e nas áreas de Ciências, Engenharia etc. Outros trabalhos apontam uma série de atividades que podem ser realizadas como: atividades que não usam das tecnologias (*Computer Science Unplugged*), a própria programação, a robótica, a produção de narrativas digitais, a criação de *games*, e o uso de simulações para a investigação de fenômenos (LEE et al., 2011; LEE; MARTIN; APONE, 2014).

3.1 Atividades sem o uso das tecnologias

Essas atividades estão sendo propostas e estudadas como parte do projeto *Computer Science Unplugged* desenvolvido pela Universidade de Canterbury, Nova Zelândia (BELL et al., 2009), que tem como objetivo o ensino de conceitos da Ciência da Computação sem o uso do computador. A ideia é desenvolver atividades como jogos, truques de mágica e competições para mostrar às crianças o tipo de pensamento que é esperado de um cientista da computação. Porém, como observado pelos autores, isso não significa o desenvolvimento de atividades de simulação de um computador, por exemplo, mas a resolução de um problema para atingir um determinado objetivo e, nesse processo, lidar com conceitos fundamentais de

Ciência da Computação. Por exemplo, tentar completar um mapa de piratas. Trata-se de um problema que pode ser caracterizado como um autômato finito e a atividade consiste em percorrer o campo do jogo, tentando encontrar um caminho para a "Ilha do Tesouro".

Embora essas atividades sejam fáceis de ser implementadas e possam ser realizadas com alunos que ainda não têm acesso à tecnologia e em praticamente em qualquer lugar, elas têm sido criticadas pelo fato de, como apontado pelos pesquisadores que participaram do *National Research Council* em 2011 (USA NATIONAL RESEARCH COUNCIL, 2011), o trabalho com as tecnologias digitais ter algo especial que não pode ser explorado por outras atividades. Por exemplo, Grover e Pea (2013) reconhecem o valor das atividades que não usam as tecnologias digitais, porém argumentam que elas podem não ser tão proveitosas por manter os alunos distantes de experiências com as tecnologias digitais. Isso pode acarretar o desenvolvimento de uma visão distorcida da computação, ou ainda distanciá-los da própria área uma vez que a prática de programar o computador é fundamental e praticamente única.

3.2 ProgramaçãoScratch

O *Scratch* tem como base a linguagem Logo, porém a programação é baseada em uma linguagem de blocos visuais, projetados para facilitar a manipulação da mídia por programadores novatos. O *Scratch* substitui a digitação do código por blocos, sendo que cada um corresponde a uma ação específica que o computador realiza. O bloco pode ser escolhido, arrastado e encaixado em outros blocos para a formação de instruções para o computador. Esses blocos facilitam o processo de descrição das instruções para a máquina uma vez que a sintaxe das instruções é definida pelo encaixe dos blocos, contribuindo para minimizar esse tipo de erro, que é muito comum em linguagem de programação baseada na codificação de comandos. As outras características da programação, identificadas na espiral de aprendizagem (VALENTE, 2005), ainda estão presentes, como a execução dos blocos, a reflexão que o aluno realiza e a depuração.

As atividades de programação *Scratch* enfatizam a manipulação da mídia, que tem uma forte ressonância com as atividades nas quais as crianças e jovens estão interessados, como a criação de histórias animadas, jogos e apresentações interativas. Por exemplo, as crianças podem animar histórias tipo dramas, comum ou mais personagens, reproduzindo experiências do cotidiano, como animais perseguindo pessoas e animais que falam.

Como afirma Resnick (criador do *Scratch*), a ideia é criar histórias, animação e compartilhar essa produção por intermédio da *Scratch Wiki* (2015), e não enfatizar tanto o uso de programação para resolução dos problemas clássicos da Computação como o cálculo da série de Fibonacci e cálculo do mínimo divisor comum[i].

Com base no estudo de atividades encontradas na comunidade *Scratch online* e nas oficinas *Scratch*, os pesquisadores Brennan e Resnick (2012) identificaram três dimensões que, segundo esses autores, estão envolvidas no pensamento computacional: conceitos computacionais (conceitos empregados na definição de programas, como interação, paralelismo, condicionais), práticas computacionais (práticas de como desenvolver programas, como ser incremental ou interativo, depurar, reusar), e perspectivas computacionais (perspectivas que o programador desenvolve sobre o mundo à sua volta e sobre si mesmo, como capacidade de expressão, de conexão).

Além do *Scratch*, Mannila e colaboradores (2014) mencionam outras linguagens visuais que também podem ser utilizadas como *Snap*!, Alice, que possibilitam criar programas por meio da combinação de comandos icônicos na forma de peças de um quebra-cabeça que só podem ser encaixadas se corresponderem a um código sintaticamente correto (MANNILA et al., 2014).

3.3 Robótica pedagógica

A robótica pedagógica consiste na "utilização de aspectos/abordagens da robótica industrial em um contexto no qual as atividades de construção, automação e controle de dispositivos robóticos, propiciam aplicação concreta de conceitos, em um ambiente de ensino e de aprendizagem" (D'ABREU, 2012). O dispositivo robótico pode ser desconectado do computador, e nesse caso, o robô é programado fornecendo uma série de instruções diretamente a ele, de modo que executando essas instruções sequencialmente, o robô realiza uma determinada tarefa. No caso de o robô ser conectado a um computador, ele pode ser programado em termos de acionamento de motores e de sensores que devem responder de acordo com a programação e as reações do ambiente. Por exemplo, ele pode ser programado para, caso encontre um obstáculo, ser capaz de contorná-lo e seguir o seu percurso.

Assim, de modo geral, as atividades de robótica pedagógica podem ser vistas como programação, com a vantagem de trabalhar com objetos concretos, como máquinas que se movem como elevadores, máquina de lavar roupa etc., cujo comportamento é produzido pela combinação de conceitos abstratos de diferentes áreas do conhecimento, como Ciências, Matemática; e conhecimentos de Engenharia, como automação, controle de mecanismos eletromecânicos. Todas essas atividades envolvem etapas como concepção, implementação, construção, automação e controle do mecanismo.

Atividades de robótica pedagógica foram realizadas com bastante sucesso com alunos do 5º ano de uma das escolas que participaram do Projeto Um Computador por Aluno (UCA-UNICAMP), para a qual as professoras foram formadas e, juntamente com alunos monitores, desenvolveram essas atividades como parte das atividades de sala de aula (BASTOS; D'ABREU; GIACHETTO, 2014).

3.4 Produção de narrativas digitais

As narrativas digitais consistem no uso das TDIC na produção de narrativas que tradicionalmente são orais ou impressas. Na literatura são conhecidas como histórias digitais, relatos digitais, narrativas interativas, narrativas multimídia, narrativas multimidiáticas, ou *digital storytelling*. Elas acrescentam novas possibilidades uma vez que o digital permite a criação de diferentes letramentos. Além da escrita podem ser usadas imagens, animação, vídeos e sons. Outro fato importante é que elas passam a ter as mesmas propriedades de um programa computacional. A sua elaboração envolve as mesmas ações como descrição, execução, reflexão e depuração, possibilitando a realização da espiral de aprendizagem. Além disso, pelo fato de representar os conhecimentos que o aprendiz usa na sua narrativa, elas constituem uma "janela na mente" do aprendiz permitindo que esses conhecimentos possam ser explicitados, identificados e passíveis de serem depurados.

A narrativa digital pode ser elaborada usando uma linguagem de programação como o *Scratch*. Por outro lado, ela pode ser elaborada por *software* que não enfatizam a programação, mas o uso de algum tipo de *software* para apresentar a história de uma forma divertida. Nesse caso podem ser usados, por exemplo, recursos de um PowerPoint ou de outro *software* de autoria.

Outro aspecto bastante importante das narrativas digitais é a possibilidade de variações que elas oferecem, como narrativas construídas basicamente com imagens, ou narrativas sonoras (rádio na escola) ou a combinação de diferentes recursos computacionais, como vídeo, texto e Prezy ou PowerPoint, e de atividades presenciais e computacionais (teatro tradicional combinado com tecnologia).

3.5 Criação de games

Os jogos digitais ou os *games* são sistemas (SALEN; ZIMMERMAN, 2003) constituídos de basicamente quatro elementos: a estética, entendida como o desenho dos personagens, uso de som, música, cores; a narrativa, a história por detrás do *game*; a mecânica, como as regras funcionam, o que é válido ou o que pode ser feito ou não como parte da trama; e a tecnologia, os *software* usados bem como os dispositivos que executar o *game*. Portanto, estão envolvidos diversos conhecimentos de diversas áreas como Artes, Comunicação, Programação e, dependendo da narrativa, conhecimentos de Matemática, Ciência etc. Como afirma Burn (2007),os jogos digitais podem ser vistos como textos multimodais, capazes de estabelecer pontes entre os diversos conhecimentos presentes no currículo, além de combinar processos criativos e artísticos.

Por outro lado, toda essa engenharia pode ser explorada do ponto de vista educacional, colocando os alunos na posição de desenvolvedores de *games*. A criação de jogos pode ser vista como uma atividade rica para a aprendizagem, com o potencial de poder integrar diferentes áreas do conhecimento, normalmente desintegradas na organização do currículo tradicional.

Essa tem sido a estratégia escolhida por um dos grupos de pesquisa do *London Knowledge Laboratory*, que desenvolve o *software MissionMaker* para estudantes criarem jogos digitais (DE PAULA; VALENTE; BURN, 2014). Por intermédio desse *software*, o aluno pode escolher objetos para montar cenários (como salas, portas, objetos manipuláveis, personagens, que podem ser escolhidos pelos usuários) e ativar objetos por meio de regras lógicas produzidas através de uma programação rudimentar baseada em objetos e regras na forma condicional “se condição, ação”.

Nesse contexto de produção de jogos digitais, as atividades realizadas pelos aprendizes utilizam as concepções de programação, aliadas a uma série de outros conhecimentos. Certamente, a criação de jogos digitais tem todas as características para a exploração de conceitos do pensamento computacional. Assim, os pesquisadores envolvidos nesse projeto realizaram os primeiros estudos em duas escolas, verificando, por exemplo, a possibilidade de alunos do 5º ano de uma das escolas usarem esse *software* e o que conseguiam produzir. Os resultados se mostraram bastante promissores (DE PAULA; VALENTE; BURN, 2014).

3.6 Uso de simulações

Em Ciências, por exemplo, muitos fenômenos podem ser simulados, permitindo o desenvolvimento de atividades ou a criação de um mundo-do-faz-de-conta, onde certas atividades não são passíveis de serem desenvolvidas no mundo real. Em Matemática, a animação também pode se tornar importante para a elaboração de gráficos dinâmicos, permitindo que a variação de alguns parâmetros produza efeitos imediatos nos gráficos.

Software para a realização de simulações de fenômenos de física, química, biologia e meio ambiente ou para a exploração de diversos temas em Matemática são encontrados na internet, como o *site* do PhET, do *Interactive Simulation Project*, desenvolvido pela Universidade de Colorado (PhET, 2010). Nesse repositório é possível encontrar uma grande variedade de simulações explorando conceitos das mais diferentes áreas do conhecimento. Nessas simulações, o aprendiz pode escolher situações específicas ou valores para variáveis pré-determinadas e observar como o fenômeno definido se comporta.

Com base nos resultados obtidos, o aprendiz deve refletir sobre as escolhas feitas e tentar entender como as variáveis afetam o comportamento do fenômeno e, com isto, ele pode construir a representação matemática do fenômeno.

A escolha das situações e das variáveis pode ser vista como parte da programação, uma vez que o aprendiz pode realizar o mesmo ciclo de ações da programação. Nesse caso, a descrição do fenômeno está praticamente definida, faltando somente a descrição e o registro do conjunto de situações ou variáveis que ele usou[ii]. Por outro lado, é importante saber como alterar as variáveis e estabelecer estratégias de como variá-las para poder tirar conclusões importantes para a compreensão do fenômeno.

As possibilidades para a exploração dos conceitos relacionados com o pensamento computacional são inúmeras e bastante diversificadas. Essas possibilidades criam ricas oportunidades de pesquisa como, por exemplo, entender as especificidades de cada uma dessas atividades e como elas contribuem para o desenvolvimento desses conceitos; como formar professores para saber explorar essas possibilidades no contexto educacional; ou como implantar essas atividades integradas com as atividades curriculares.

Além dessas diferentes maneiras como o pensamento computacional pode ser explorado na educação, membros da ISTE/CSTA (2011) também examinaram as estratégias como o pensamento computacional pode ser implantado nas disciplinas do K-12 (Educação Básica), identificando os desafios que esse processo pode enfrentar, e as formas com que os educadores universitários podem participar. O trabalho desenvolvido por Barr e Stephenson (2011) discute múltiplas definições propostas para o pensamento computacional, e como os nove conceitos identificados podem ser trabalhados nas disciplinas de Ciência da Computação, Ciências, Tecnologia, Matemática e Línguas.

4 O PENSAMENTO COMPUTACIONAL NA EDUCAÇÃO BÁSICA

As estratégias para implantação das tecnologias na Educação Básica, adotadas por diferentes países, podem ser classificadas em termos de três grandes categorias: atividades de Ciência da Computação, como a programação, sendo subdividida em duas outras subcategorias a) programação fora da sala de aula, e b) a inserção de disciplinas no currículo que usam tecnologias para explorar temas relativos ao letramento digital ou *computer literacy*; a inclusão de disciplinas no currículo nas quais são desenvolvidas atividades que exploram conceitos do pensamento computacional, como jogos e robótica; e a exploração dos conceitos do pensamento computacional de maneira transversal, por meio de atividades que usam as tecnologias em diferentes disciplinas do currículo.

4.1 Atividades de Ciência da Computação, especialmente a programação

A programação pode ser entendida no contexto de Ciência da Computação ou *coding*, na qual a atividade consiste em fornecer instruções ao computador a fim de resolver um conjunto de problemas tradicionais. A ênfase é o aprendizado de programação, de conceitos de Ciência da Computação e o preparo para o mercado de trabalho. Esse

aprendizado acontece por meio de diversas disciplinas específicas que trabalham esses conceitos. Segundo o relatório *European Schoolnet* (2014) e o estudo de Mannila et al. (2014), a maioria dos países realiza esse tipo de atividade relacionada com o Ensino Médio. No Brasil, escolas técnicas do Ensino Médio adotam essa mesma linha.

Diversos autores sugerem que a exploração das atividades de Ciência da Computação, especialmente atividades de programação, são importantes para o desenvolvimento de conceitos relacionados ao pensamento computacional (GROVER; COOPER; PEA, 2014; SETTLE et al., 2012; SCHULTE, 2013). No entanto, elas podem ocorrer tanto fora da sala de aula quanto como parte das atividades curriculares.

4.1.1 Atividades de programação fora da sala de aula

Nos Estados Unidos da América (EUA), a preocupação com o desenvolvimento do pensamento computacional ainda não atingiu as mudanças no currículo da Educação Básica (K-12). As iniciativas nesse sentido estão partindo de empresas e organizações sem fins lucrativos, incluindo o *Code.org*, o *College Board*, o *National Mathand Science Initiative*, o *Teach for America*, e o *Project Lead the Way*, que estão envolvidas em atividades de produção de materiais curriculares ou promoção e oferecimento de cursos de formação para professores ou para a comunidade em geral, interessada na área de Ciência da Computação (WHITE HOUSE, 2013).

Por exemplo, a *Code.org* (CODE, 2016) foi fundada em 2013 e tem contribuído enormemente para a sensibilização sobre a falta de disciplinas de Ciência da Computação na Educação Básica dos EUA. Os funcionários e voluntários dessa organização têm trabalhado para possibilitar que as atividades de Ciência da Computação façam parte da Educação Básica, desenvolvendo materiais para os professores usarem com seus alunos. A partir de julho de 2014, foram criados três cursos de Ciência da Computação, sendo o primeiro para alunos do início da Educação Básica (4-6 anos), o segundo para iniciantes com idade acima de 6 anos e o terceiro para alunos mais experientes com idades acima de 6 anos. Esses cursos misturam atividades *online*, tutoriais autoguiados e atividades que os alunos realizam presencialmente, fora da plataforma.

A Google montou uma página de lições prontas para serem usadas em sala de aula e exemplos de programas para educadores da Educação Básica (GOOGLE, 2015). Eles também estão financiando oficinas para professores para implantação de atividades de Ciência da Computação em todos os níveis de ensino (GOOGLE, 2015).

O Brasil segue, em menor escala, os passos dos EUA. Ainda não está sendo pensada a mudança curricular, porém existem diferentes iniciativas no sentido de promover o letramento em programação, como a Escola de *Hackers*, da Prefeitura Municipal de Passo Fundo (RS); o Programaê!, da Fundação Lemann; e o Go Code, da Fundação Maurício Sirotsky Sobrinho (INSTITUTO AYRTON SENNA, 2015).

Essas iniciativas são louváveis e têm produzido resultados importantes. No entanto, elas estão centradas estritamente na programação de computadores. Isso pode trazer benefícios importantes, como a promoção da criatividade e do pensamento computacional, como tem sido argumentado (WHITE HOUSE, 2013; ROYAL SOCIETY, 2012); ou, por outro lado, podem produzir resultados não muito interessantes quando coloca a ênfase na técnica de programação com o objetivo de desenvolver programadores, ou promover a disseminação da Ciência da Computação e o desenvolvimento de pessoas que pensam como os profissionais da computação.

A ênfase na Ciência da Computação, especificamente na programação, tem sido criticada por diferentes pesquisadores, inclusive da própria área. Denning (2009), por exemplo, entende que, o tem sido proposto como "pensamento computacional" é amplamente utilizado por pessoas nas Ciências e, portanto, não é válido caracterizá-lo como uma contribuição única da Ciência da Computação. Hemmendinger (2010) entende a proposta do pensamento computacional vinculada à Ciência da Computação como tendo um tom arrogante, sugerindo que os cientistas da computação estão tentando dizer às pessoas de outras áreas como elas devem pensar. No entanto, como Denning, Hemmendinger argumenta que o pensamento computacional não deve ser território restrito da Ciência da Computação. Na verdade, promover o pensamento computacional é ajudar as pessoas a pensarem como um economista, um físico, um artista, ou seja, a entender como usar a computação para resolver os seus problemas, para criar e para descobrir novas questões que podem ser exploradas produtivamente. E conclui propondo que "os cientistas da computação podem contribuir, mas devemos ter cuidado para não falar como se fôssemos os únicos a levar as pessoas a uma terra

prometida. No final, talvez devêssemos falar menos sobre o pensamento computacional, e nos concentrar mais no fazer computacional" (HEMMENDINGER, 2010, p.6).

De todo modo, é interessante notar que não existe nenhum país que prevê o ensino de programação no currículo formal do Ensino Fundamental (K-9). Segundo Mannila e colaboradores (2014), isso se deve provavelmente ao fato de não existir um consenso sobre como e quando a programação deve ser ensinada às crianças nesse nível de ensino.

4.1.2 Disciplinas no currículo sobre computer literacy ou letramento digital

A outra vertente de uso da programação entende o *coding* como uma parte do processo de programação. Atualmente, a programação tem sido interpretada como parte do letramento digital ou da inclusão digital, podendo ser usada como um meio de autoexpressão e de participação social; como uma ferramenta para conceber e criar coisas, e desenvolver a criatividade; ou como uma maneira para as crianças ampliarem suas experiências e colocarem em prática suas próprias ideias. Nesse sentido, a codificação da solução do problema é uma das últimas coisas a serem feitas, sendo que o foco deve consistir em entender as potencialidades das tecnologias digitais para a realização de uma série de atividades relacionadas com praticamente todas as disciplinas curriculares (MANNILA et al., 2014).

O letramento digital faz parte das atividades da Educação Básica de praticamente todos os países da Comunidade Europeia e dos EUA. No entanto, isso acontece de diversas maneiras. Em alguns países, essa atividade é desenvolvida como uma disciplina específica, geralmente denominada de *Information Communication Technology (ICT)* ou *Information Technology (IT)* como, por exemplo, na Lituânia, Holanda, Suécia (MANNILA et al., 2014). Além disso, em alguns países essa disciplina é prevista ao longo do equivalente ao Ensino Fundamental II, como no caso da Lituânia, Portugal, Grécia, Irlanda; ou algumas disciplinas previstas em alguns anos específicos, como no caso da Bulgária, onde a disciplina de Informática é obrigatória no 9º ano (EUROPEAN SCHOOLNET, 2014).

Por exemplo, na Lituânia, desde 2005, o foco principal é o desenvolvimento de conhecimentos de informática. As noções básicas são trabalhadas no 5º e 6º anos nas diciplinas de *IT*, centradas no uso do Logo e do *Scratch*. No 9º e 10º anos são previstas 34 horas de um módulo opcional sobre projetos de algoritmos e programação. Essa introdução à

programação serve de base para o ensino de informática no Ensino Médio ou para os alunos que escolhem a especilização em Ciência da Computação (MANNILA et al., 2014).

Já em outros países, como Dinamarca e EUA, a inclusão de tópicos relacionados com informática acontece como parte de outras disciplinas. No caso da Dinamarca, ela acontece desde o primeiro ano até o 10º como parte da disciplina de Ciências. Nos EUA, essa inclusão depende do estado ou do sistema local de ensino. Em geral, o letramento digital é realizado por meio de atividades que são realizadas como parte das disciplinas de Língua Inglesa, que prevê temas como padrões para leitura: integrar e avaliar conteúdo apresentado em diversas mídias e formatos; e normas para a escrita: uso de tecnologia, incluindo a Internet, para produzir e publicar textos e para interagir e colaborar com outros; e para reunir informações relevantes a partir de fontes impressas ou digitais, avaliara credibilidade e precisão de cada fonte, e integrar a informação, evitando plágio (MANNILA et al., 2014).

No entanto, a maioria dos países está procurando alterar essas atividades relacionadas ao letramento digital para poder explorar os conceitos sobre o pensamento computacional, sendo que alguns deles, como a Inglaterra, já fizeram esse movimento.

4.2 O Pensamento computacional como disciplina curricular

O país que está mais adiantado na implantação do pensamento computacional é a Inglaterra. A partir de setembro de 2014, entrou em vigor o Currículo Nacional Inglês (UK DEPARTMENT FOR EDUCATION, 2013), que implanta a disciplina de *Computing*, substituindo a antiga *ICT*, que focava o letramento digital, a qual foi considerada "rasa" e "irrelevante" (UK DEPARTMENT FOR EDUCATION; GOVE, 2012). Para tanto, foi desenvolvido o projeto *Computing at School* propondo um guia denominado *Computing in the National Curriculum: a guide for primary teachers*. (BERRY, 2013).

O novo currículo tem como objetivo criar condições para elucidar como as tecnologias digitais funcionam, quais seus impactos e relações com a sociedade e, as diferentes formas de sua utilização em diferentes contextos e situações, criando, assim, condições para que os alunos compreendam as lógicas dessas tecnologias e sejam capazes de refletir sobre a presença dessas tecnologias na sociedade. Como exemplo, espera-se que os alunos do Primeiro Ciclo – o primeiro estágio da educação compulsória na Inglaterra, composto pelos dois primeiros anos, frequentados por crianças com idades entre 5 e 7 anos –

estejam aptos a "compreender o que são algoritmos [...]; usar raciocínio lógico para prever o comportamento de programas simples; criar e depurar programas simples" (UK DEPARTMENT FOR EDUCATION, 2013, p. 218).

Os pesquisadores e idealizadores das políticas da Educação na Inglaterra entendem que as atividades usando as tecnologias digitais não devem promover o ensino de "*software* de escritório" (BUCKINGHAM, 2007), mas os meios para que os alunos compreendam os sistemas computacionais, como eles funcionam, e como são projetados e programados. Além disso, devem criar condições para promover o "pensamento computacional", definido pela *Royal Society* (2012, p. 29) como "o processo de reconhecer aspectos computacionais no mundo que nos cerca, e a aplicação de ferramentas e técnicas da Ciência da Computação para compreender e refletir sobre sistemas naturais e artificiais".

É importante destacar que o argumento econômico desempenhou um papel relevante na inclusão do *Computing* no currículo inglês. Uma das preocupações que levou à alteração do currículo computacional foi a falta de mão de obra qualificada para o mercado de trabalho ligado às tecnologias. Essa preocupação foi expressa, inclusive, por grandes empresas de tecnologia, como Microsoft e Google, que estiveram envolvidas na revisão e na elaboração do novo currículo (DREDGE, 2014). Certamente, esse é um elemento que não pode ser ignorado na análise do novo currículo computacional inglês. Por outro lado, é inegável que esse novo currículo tem grande influência dos conceitos do pensamento computacional, focando as bases conceituais que sustentam as tecnologias digitais ao invés dos seus usos em si.

O aspecto que merece crítica quanto a essa proposta da Inglaterra é o fato de o pensamento computacional estar centrado em uma única disciplina e não ser integrado aos assuntos curriculares de todas as disciplinas, como tem sido proposto por Almeida e Valente (2011). No entanto, merece destaque positivo o fato de as atividades propostas não focarem somente a programação. O governo e entidades de financiamento estão incentivando os grupos de pesquisas das universidades e de empresas no desenvolvimento de materiais explorando diversos contextos para trabalhar os conceitos do pensamento computacional. Por exemplo, no *London Knowledge Laboratory*, vinculado a *Institute of Education* da Universidade de Londres dois projetos estão sendo desenvolvidos nesse sentido. O Projeto *ScratchMaths*, cujo objetivo é apoiar pensamento computacional e matemático através da programação usando a linguagem *Scratch* (HOYLES, 2015; SCRATCHMATHS, 2015).

Outro projeto é o desenvolvimento do *software MissionMaker* para que os alunos possam produzir jogos que se assemelham aos que eles usam cotidianamente (DE PAULA; VALENTE; BURN, 2014).

4.3 O pensamento computacional como uma atividade transversal ao currículo

Em outros países, como Itália, o letramento digital está sendo integrado como tema a ser estudado nas disciplinas da Educação Básica, nas quais são exploradas atividades como atitude crítica na utilização das tecnologias digitais para o trabalho, a vida, a comunicação; e do uso do computador para recuperar, avaliar, manter, produzir, apresentar, compartilhar informações bem como cooperar através da Internet. Os aspectos do pensamento computacional ficam por conta de professores mais motivados e mais entusiastas que trabalham as ferramentas mais comuns da Ciência da Computação, como programação de robôs, e a introdução à programação com linguagens simples, para criar e desenvolver projetos (MANNILA et al., 2014).

Situação semelhante ao que acontece na Itália foi realizada no Brasil com o desenvolvimento do Projeto Um Computador por Aluno (Projeto UCA). Os *laptops* educacionais foram utilizados em situações de sala de aula, nas mais diferentes disciplinas, especialmente do Ensino Fundamental. Na maioria dos casos, essas tecnologias foram usadas na realização de atividades curriculares como produção de textos, de gráficos; *software* de autoria para desenvolvimento de *blogs* ou de narrativas digitais; e para acesso à informação na internet ou em bancos de dados específicos. Professores mais experientes exploraram o uso de simulação, de jogos, programação, robótica e produção de narrativas digitais como o projeto Rádio na Escola (VALENTE; BARANAUSKAS; MARTINS, 2014). Embora as atividades estivessem restritas a uma disciplina curricular, os projetos que os alunos desenvolveram tinham um caráter interdisciplinar, uma vez que podiam explorar conceitos de Ciências, de Matemática, de Português e Arte.

O pensamento computacional ainda não tem sido sistematicamente explorado na Educação Básica de nenhum dos países mencionados. A Finlândia, por exemplo, está propondo alterações em seu currículo, que hoje prevê disciplinas específicas para o trabalho no âmbito do letramento computacional. A partir de 2016, a disciplina de *ICT* deverá focar atividades explorando o pensamento computacional.

Assim, a exploração do pensamento computacional como tema transversal às disciplinas do currículo ainda está em gestação. A falta de uma definição clara sobre o que consiste esse pensamento dificulta ainda mais a sua implementação. Além disso, existem dois outros temas complicadores que devem ser superados para que a inserção do pensamento computacional possa ser efetiva: a formação de professores para realizarem essas atividades e a avaliação do aluno com relação ao desenvolvimento do pensamento computacional.

5. FORMAÇÃO DE PROFESSORES E AVALIAÇÃO DO DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL

A explicitação do que se entende e o que constitui o pensamento computacional e como ele é inserido no currículo permite a definição de estratégias de avaliação, bem como determina o material e os conteúdos do que deve ser contemplado nas atividades de formação de educadores, e o que deve ser realizado nas práticas educacionais e os seus respectivos benefícios.

5.1 Formação de professores

A formação de professores para que possam desenvolver atividades relacionadas ao pensamento computacional tem sido feita tanto no âmbito da formação inicial quanto da continuada. Por exemplo, Yadav e colaboradores (2011; 2014) introduziram um módulo sobre o pensamento computacional como parte do curso "Aprendizagem e Motivação", obrigatório para a formação de professores para a Educação Básica da *Purdue University*. O objetivo desse módulo foi não só expor os alunos a ideias sobre computação, mas mostrar como essas ideias podem ser usadas em suas futuras carreiras docentes. Foram realizadas avaliações dos estudantes usando pré e pós-testes com o objetivo de entender o nível de compreensão e atitude sobre o pensamento computacional, e a influência desse módulo na formação dos estudantes. De maneira geral, o módulo ajudou os estudantes a entenderem que (1) eles podem ensinar conceitos de computação nas salas de aula da Educação Básica (K-12) sem o uso de computadores e (2) conceitos sobre o pensamento computacional podem ser incorporados em todas as disciplinas.

Com relação à formação continuada, Imberman, Sturm e Azhar (2014), docentes da *City University of New York* (CUNY), descrevem o resultado de um *workshop* realizado com 25 professores do equivalente ao Ensino Fundamental II e Ensino Médio. Esse *workshop* tinha como objetivo discutir com esses professores conceitos relacionados com o pensamento computacional e trabalhar uma variedade de recursos para ensinar esses conceitos, como programação *Scratch*, robótica pedagógica, e atividades sem o uso de computadores (*Computer Science Unplugged*).

O *workshop* teve a duração de 15 horas e foi organizado em várias sessões, cada sessão trabalhando um recurso específico. Em cada sessão eram previstas atividades "mão-na-massa", nas quais os professores aprenderam fazendo e sendo auxiliados por facilitadores que discutiam os conteúdos e as atividades sobre o pensamento computacional. A ideia por detrás de cada sessão foi a de fornecer aos professores as condições para que se sentissem confortáveis e motivados para aprender e, finalmente, incorporar os recursos e os conceitos aprendidos em suas práticas de sala de aula.

Os professores foram avaliados antes das sessões da oficina e no final do *workshop*. Várias perguntas dessas avaliações foram diretamente relacionadas com a experiência do professor e sua atitude com relação a cada um dos recursos trabalhados no *workshop*, juntamente com a sua impressão geral sobre o valor do *workshop* para a sua experiência. Os resultados mostraram que os participantes foram unânimes em afirmar (6 concordando e 19 concordando fortemente) que o *workshop* foi útil e indicaram que eles estavam propensos a incluir os recursos trabalhados em suas salas de aula. Os autores da pesquisa concluíram que um *workshop* que fornece uma visão ampla sobre os recursos existentes e uma imersão, mesmo que breve, em cada um desses recursos computacionais, pode ser efetivo em alterar a visão desses professores, que passam a considerar o uso desses recursos em suas práticas de sala de aula. No entanto, os professores participantes do *workshop* também indicaram que as barreiras encontradas nas escolas podem ser um importante obstáculo para que essas atividades não se concretizem.

O trabalho realizado por Mannila e colaboradores (2014) mostra que os professores precisam de apoio para entrar em uma situação na qual eles ainda não dominam o material a ser trabalhado em sala de aula. Isso significa a criação de um ambiente no qual eles possam aprender junto com seus alunos e, para tanto, necessitam do apoio dos gestores, dos colegas da universidade. Esses autores mencionam a experiência do projeto *Teacher for Teacher*

(T4T) desenvolvido na Universidade de Torino, na Itália, envolvendo um grupo de professores de todos os níveis de ensino que, em conjunto com pesquisadores da universidade, desenvolvem atividades de Ciência da Computação nas escolas. O grupo escolhe um conjunto de atividades, que é trabalhado em sala de aula. Com base nos resultados, essas atividades são revisadas, melhoradas e em seguida distribuídas para outros professores, por meio de *workshops* que são realizados no início do ano letivo.

5.2 Natureza e avaliação do pensamento computacional

A explicitação da concepção e dos conceitos relativos ao pensamento computacional, como foram discutidos anteriormente, tem sido a preocupação de diversos pesquisadores e instituições que se interessam por esse tema, como por exemplo, os pesquisadores de diferentes áreas que participaram dos *workshops* organizados pela *USA National Research Council* (2010; 2011); instituições como *International Society for Technology in Education* (ISTE) e a *American Computer Science Teachers Association*, ou empresas como Google e Microsoft. No entanto, ainda não é possível estabelecer um consenso sobre o que consiste o pensamento computacional, embora a definição e os conceitos prevalentes sejam os propostos pela parceria ISTE/CSTA (2011).

As abordagens utilizadas na avaliação de aprendizes quanto ao desenvolvimento do pensamento computacional têm sido bastante diversificadas, seguindo diferentes procedimentos metodológicos, na tentativa de identificar se os sujeitos desses estudos estão realmente conscientes e procedendo de acordo com os conceitos que foram selecionados e trabalhados durante o processo de intervenção.

Brennan e Resnick (2012) utilizaram em seu estudo três atividades para avaliar se os usuários de *Scratch* procediam de acordo com as dimensões e as práticas computacionais que eles definiram como relacionadas ao pensamento computacional, que foram mencionadas anteriormente. O primeiro procedimento foi a análise do portfólio de projetos dos aprendizes. Esses aprendizes, durante um determinado período, acumularam diferentes projetos, que fazem parte do *site* da Comunidade *Scratch*. Por exemplo, um dos portfólios analisados é de um jovem de 17 anos que acumulou, durante três anos, 49 projetos em seu portfólio. Essa análise procurou identificar a presença ou não das dimensões computacionais nos programas analisados. Para agilizar essa análise, os autores utilizaram um *software Scrape User*

Analysis, que procura blocos *Scratch* e permite visualizar os blocos e a quantidade de blocos usados nos programas de um determinado usuário.

Essa análise apresenta certas limitações, pois é estritamente focada em conceitos computacionais e nada revela sobre o processo ou as práticas computacionais adotadas na produção dos programas. Os programas analisados poderiam ter sido copiados ou simplesmente adaptados de outros usuários.

Para entender sobre as práticas computacionais adotadas, Brennan e Resnick (2012) entrevistaram 31 programadores *Scratch* com idade entre 8 e 17 anos de diferentes localidades dos EUA, Europa e Ásia. Essas entrevistas tiveram duração entre 60 a 120 minutos e o protocolo foi organizado em quatro seções: Antecedentes, Criação do Projeto, Comunidade *online* e Olhando para o Futuro. Esse procedimento se mostrou útil, porém consumiu muito tempo e, em geral, os programadores não se lembravam de alguns detalhes e as entrevistas não exploravam as práticas computacionais em tempo real.

A solução foi utilizar outro procedimento, a análise de cenários, que foi realizado em um contexto escolar, estudando um pequeno número de alunos de diferentes escolas, de praticamente todos os anos escolares e disciplinas. Os autores desenvolveram três conjuntos de projetos *Scratch*, com diferentes graus de complexidade. Cada conjunto consistia de dois projetos, sendo que os projetos envolviam os mesmos conceitos e práticas, porém com diferentes estéticas, procurando adequar aos diferentes interesses dos sujeitos da pesquisa. Em uma série de três entrevistas, os alunos analisaram os cenários, que foram apresentados como projetos que foram criados por outros programadores *Scratch*. Nessas entrevistas, os alunos escolhiam um dos projetos e tinham que: explicar a razão da escolha, descrever como o projeto poderia ser estendido, corrigir um erro, e remixar o projeto, adicionando um recurso.

Esse procedimento ofereceu a oportunidade para explorar sistematicamente aspectos das práticas computacionais e permitiu avaliar o processo em ação e não via memória. Os pontos negativos foram apontados como o tempo gasto nessa avaliação e o fato de que nem sempre os projetos apresentados aos alunos tinham ressonância com seus interesses com relação ao envolvimento com o *Scratch*.

Os procedimentos usados por Brennan e Resnick (2012) avançaram muito com relação ao que tem sido proposto em termos de avaliação do desenvolvimento do pensamento computacional em programadores *Scratch*. Por outro lado, esses procedimentos foram criticados por Grover, Cooper e Pea (2014) que entenderam que o aspecto mais importante da programação não foi avaliado, ou seja, o processo de programar. Esses autores propuseram dois estudos, um presencial e outro a distância, que consistiu no desenvolvimento de um curso sobre programação *Scratch* e um processo de avaliação tanto formativa quanto somativa da aprendizagem.

O Estudo 1 foi realizado com 26 crianças do 7º e 8ª anos e consistiu de uma série de atividades como: uso de exemplos de programas para trabalhar a aprendizagem conceitual; ensino sobre a leitura e rastreamento de códigos de programa; avaliação formativa baseada em testes de múltipla escolha para promover a compreensão do aprendiz e reforçar os conceitos aprendidos; explicitação das ideias fundamentais da Ciência da Computação e pensamento computacional; uso de linguagem acadêmica para explicar conceitos em termos de vocabulário do domínio da Ciência da Computação; e a promoção da aprendizagem ativa, construtivista do *Scratch* através de várias atividades práticas. O Estudo 2 foi realizado com 28 estudantes com média de 12,3 anos de idade e usou as mesmas atividades do primeiro estudo, porém elas foram realizadas totalmente a distância e com um substancial incremento de suporte oferecido à realização das mesmas. A aprendizagem dos estudantes foi avaliada por meio de pré e pós-testes sobre conceitos de programação relacionados com o pensamento computacional.

Os resultados dos pós-testes foram estatisticamente superiores aos dos pré-testes, indicando que os procedimentos adotados foram importantes para o desenvolvimento do pensamento computacional. Além disso, no Estudo 2 os aprendizes tiveram uma performance superior a do Estudo 1.

Com base nesses resultados, Grover, Cooper e Pea (2014) concluíram que a atividade de programação por si só não é suficiente para promover o desenvolvimento do pensamento computacional. O currículo que eles elaboraram tinha como objetivo ajudar os alunos a entender estruturas mais profundas em suas atividades computacionais e avaliar essa aprendizagem por meio de múltiplas formas de avaliação. Para entender o nível de compreensão do aluno, essa avaliação não pode ser feita somente por meio de testes de

múltipla escolha, nem por meio da programação de projetos de livre escolha dos alunos. Além disso, a dedicação de mais tempo de ensino para certos temas e para diferentes exemplos de programação, como aconteceu no Estudo 2, foi fundamental para ajudar os alunos a terem um desempenho superior aos aprendizes do Estudo 1. Assim, é fundamental o papel do professor nessas atividades que tentam promover o desenvolvimento do pensamento computacional.

Pesquisadores como Lu e Fletcher (2009) acreditam que para ampliar e garantir o sucesso da participação dos alunos em atividades de Ciência da Computação é necessário lançar as bases do pensamento computacional antes que os alunos programem. Eles postulam que a programação é para a Ciência da Computação o que a construção de provas de teoremas é para a Matemática, e o que a análise literária é para o Inglês. Assim, por analogia, a programação deve ser utilizada para explorar atividades mais sofisticadas da Ciência da Computação e não ser o primeiro encontro que o estudante tem com essa ciência. Ao invés da programação, atividades relacionadas ao pensamento computacional devem explorar a criação de vocabulários e símbolos que podem ser usados para anotar e descrever atividades computacionais e abstrações, e prover notações que podem ser utilizadas para descrever o processo de construção de modelos mentais. Com essa exposição sobre o pensamento computacional, os alunos terão melhores condições de se preparar para a programação e para o currículo da Ciência da Computação, podendo escolher essa ciência como carreira profissional ou como conteúdo intelectual.

No *workshop* realizado pelo *USA National Research Council* (2011), diversos pesquisadores propõem a inclusão de atividades de programação ou de Ciência da Computação como parte das atividades na Educação Básica, embora autores como Joyce Malyn-Smith e Peter Henderson (USA NATIONAL RESEARCH COUNCIL, 2011) compartilhem da visão proposta por Lu e Fletcher (2009).

6 CONSIDERAÇÕES FINAIS

O objetivo deste artigo foi discutir temas relacionados à inclusão do pensamento computacional no currículo da Educação Básica, procurando entender as estratégias que diferentes países estão utilizando para essa inclusão, bem como a preparação de professores para realizar essas atividades curriculares e como avaliar o aluno quanto ao desenvolvimento de conceitos do pensamento computacional. Para isso, foram utilizados documentos da

literatura especializada tanto sobre pesquisas relativas aos aspectos da formação e da avaliação, quanto os documentos sobre políticas que vêm sendo adotadas para nortear a implantação do pensamento computacional no currículo.

O grande desafio dos trabalhos nessa área é o fato de ainda não existir consenso entre os pesquisadores, tanto da Ciência da Computação quanto de pesquisadores da área de tecnologias na educação, sobre em que consiste o pensamento computacional. Sem essa definição é bastante difícil estipular como esse tema pode ser abordado na educação, como formar educadores para essa atividade e como avaliar o aluno.

A análise do material estudado permitiu entender como o pensamento computacional pode ser explorado na Educação Básica. A ênfase não deve ser a programação e nem a inclusão de atividades ou de uma disciplina que foque conceitos da Ciência da Computação no currículo. As ideias sobre o pensamento computacional podem ser trabalhadas em conjunção com as disciplinas do currículo, como tem sido realizado nas escolas da Itália. Elas devem explorar outras atividades como a robótica, as narrativas digitais, trabalhadas pelos professores das diferentes disciplinas, que podem se apropriar desses recursos computacionais por intermédio de *workshops* nos moldes do que foi realizado por Imberman, Sturm e Azhar (2014).

Finalmente, as implicações dessas atividades desenvolvidas pelos alunos devem ser estudadas por meio de observações das atividades que realizam e por meio de atividades que permitam entender o grau de consciência que os alunos têm sobre os conceitos relacionados com o pensamento computacional. Considerando as diferentes atividades que podem contribuir para o desenvolvimento do pensamento computacional, o foco da avaliação não deve ser se o aluno aprendeu ou não a programar, mas o nível de consciência que ele tem sobre conceitos computacionais e como isso se manifesta nas diversas atividades que realiza.

AGRADECIMENTOS

Este trabalho foi possível graças ao apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Processo: 306320/2015-0.

REFERÊNCIAS

ALMEIDA, M. E. B.; VALENTE, J. A. Integração currículo e tecnologias e a produção de narrativas digitais. **Currículo sem Fronteiras**, [S.l], v. 12, n. 3, p. 57-82, 2012. Disponível em: <http://www.curriculosemfronteiras.org/vol12iss3articles/almeida-valente.pdf>. Acesso em: 26 jun. 2015.

_____. **Tecnologias e currículo**: trajetórias convergentes ou divergentes? São Paulo: Paulus, 2011.

BARR, V.; STEPHENSON, C. Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? **ACM Inroads**, v. 2, n. 1, p. 48-54, 2011.

BASTOS, B. L.; D'ABREU, J. V. V.; GIACHETTO, G. F. A. O. Processo de implantação da robótica pedagógica em uma escola integrante do Projeto UCA – UNICAMP. **Revista e-Curriculum**, São Paulo, v. 2, n. 12, maio/out. 2014. Disponível em: <http://revistas.pucsp.br/index.php/curriculum/article/view/20175/15393>. Acesso em: 30 jul. 2015.

BELL, T. et al. Computer Science Unplugged: School students doing real computing without computers. **The New Zealand Journal of Applied Computing and Information Technology**, v. 13, n. 1, p. 20-29, 2009.

BERRY, M. **Computing in the national curriculum**: A guide for primary teachers. Bedford, UK: Computing at School, 2013. Disponível em: <http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf.>. Acesso em: 28 set. 2014.

BRENNAN, K.; RESNICK, M. New frameworks for studying and assessing the development of computational thinking. **AERA 2012**, Vancouver, Canadá, 2012. Disponível em: <http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf>. Acesso em: 30 jul. 2015.

BUCKINGHAM, D. **Beyond Technology**: Children's learning in the age of digital culture. Cambridge, UK: Polity Press, 2007.

BURN, A. The Case of Rebellion: Researching Multimodal Texts. In: COIRO, J.; KNOBEL, M.; LANKSHEAR, C.; LEU, D. J. **Handbook of Research on New Literacies**. New York: Laurence Erlbaum, 2007.

CODE. **Página da organização Code.org**. 2016. Disponível em: <https://code.org/>. Acesso em: 31 jul. 2016.

D'ABREU, J. V. V. Como usar a robótica pedagógica aplicada ao currículo. 2012. Congresso InovaEduca 3.0. **Anais**. Disponível em: <http://inovaeduca.com.br/images/2012/Arquivos/Joao_Villhete_IE3-26-09-12.pdf>. Acesso em: 31 jul. 2015.

DE PAULA, B. H.; VALENTE, J. A.; BURN, A. O uso de jogos digitais para o desenvolvimento do currículo para a Educação Computacional na Inglaterra. **Currículo sem Fronteiras**, v. 14, n. 3, p. 46-71, set/dez 2014.

DENNING, P. J. The profession of it: Beyond computational thinking. **Commun. ACM**, v. 52, n. 6, p. 28-30, June 2009.

DREDGE, S. Coding at school: a parent's guide to England's new computing curriculum. **The Guardian**, London, 04-set-2014. Disponível em: <http://www.theguardian.com/technology/2014/sep/04/coding-school-computing-children-programming>. Acesso em: 30 jul. 2015.

EUROPEAN SCHOOLNET. **Computing our future**: Computer programming and coding. Priorities, school curricula and initiatives across Europe, October 2014.

GOOGLE. **Computer Science Learning Opportunities**. 2015. Disponível em: <https://www.google.com/edu/resources/computerscience/learning/>. Acesso em: 28 jul. 2015.

GROVER, S.; PEA, R. Computational thinking in K-12: A review of the state of the field. **Educational Researcher**, v. 42, n. 1, p. 38-43, 2013.

_____; COOPER, S.; PEA, R. Assessing Computational Learning in K-12. **ITICSE'14**, June 21–25, 2014, Uppsala, Sweden. Disponível em: <http://dx.doi.org/10.1145/2591708.2591713>. Acesso em: 30 jul. 2015.

HEMMENDINGER, D. A plea for modesty.**ACM Inroads**, v. 1, n. 2, p. 4-7, June 2010.

HOYLES, C. **Relato** [mensagem pessoal]. Mensagem recebida por <jvalente@unicamp.br> em 04 jun. 2015.

IMBERMAN, S. P.; STURM, D.; AZHAR, M. Q. **Computational thinking:** expanding the toolkit. Consortium for Computing Sciences in Colleges. 2014.

INSTITUTO AYRTON SENNA. **Letramento em Programação**. 2015. Disponível em: <http://www.institutoayrtonsenna.org.br/todas-as-noticias/letramento-em-programacao/>. Acesso em: 28 jul. 2015.

ISTE/CSTA. **Computational Thinking Teacher Resource**. 2 ed., 2011. Disponível em: <http://csta.acm.org/Curriculum/sub/CurrFiles/472.11CTTeacherResources_2ed-SP-vF.pdf>. Acesso em: 29 jul. 2015.

LEE, I.; MARTIN, F.; APONE, K. Integrating computational thinking across the K–8 curriculum. **ACM Inroads**, v. 5, n. 4, p. 64-71, 2014.

_____; et al. Computational thinking for youth in practice. **ACM Inroads**, v. 2, n. 1, p. 32-37, 2011.

LU, J. J.; FLETCHER, G. H. Thinking about computational thinking. In: **Proceedings of the 40th ACM Technical Symposium on Computer Science Education**, p. 260-264, 2009.

MANNILA, L. et al. Computational Thinking in K-9 Education. **ITiCSE-WGR'14**, Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference. June 21-25, 2014, Uppsala, Sweden, p. 1-29. Disponível em: <http://dx.doi.org/10.1145/2713609.2713610>. Acesso em: 29 jul. 2015.

MANTOAN, M. T. E. O Processo de Conhecimento – tipos de abstração e tomada de consciência. **NIED-Memo 27**. NIED-UNICAMP; Campinas. 1994.

PAPERT, S. (1980) Mindstorms. Children, computer and powerful ideas. New York: Basic Books. Traduzido como **Logo**: Computadores e Educação. São Paulo: Brasiliense, 1985.

_____. Teaching Children Thinking, **Logo Memo nº 2**, 1971. Disponível em: <https://archive.org/stream/bitsavers_mitaiaimAI_471587/AIM-247_djvu.txt>. Acesso em: 28 jul. 2015.

PHET **Interactive Simulations**. 2010. Disponível em: <http://phet.colorado.edu/>. Acesso em: maio 2015.

PIAGET, J. **Abstração Reflexionante**: relações lógico-aritméticas e ordem das relações espaciais. Porto Alegre: ArtMed, 1995.

ROYAL SOCIETY. **Shut down or restart?** The way forward for computing in UK Schools. 2012. Disponível em: <https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf>. Acesso em: 30 jul. 2015.

RUSHKOFF, D. **Program or be programmed**: ten commands for the digital age. Berkeley: Soft Skull Press, 2011.

SALEN, K.; ZIMMERMAN, E. **Rules of Play**: Game design fundamentals. London: MIT Press, 2003.

SCHULTE, C. Reflections on the role of programming in primary and secondary computing education. In: **Proc. of the 8th WiPSCE**. ACM, November, p. 11-13, 2013.

SCRATCHMATHS. **Projeto ScrathMaths**. 2015. Disponível em: <http://www.lkl.ac.uk/cms/index.php?option=com_content&task=view&id=603&Itemid=91>. Acesso em: 30 jul. 2015.

SCRATCH WIKI. **Página do Scratch Wiki**. 2015. Disponível em: <http://wiki.scratch.mit.edu/wiki/Scratch_Wiki_Home>. Acesso em: 30 jul. 2015.

SETTLE, A. et al. Infusing computational thinking into the middle and high-school curriculum. In: **Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education**, ITiCSE '12, New York, NY, USA, p. 22-27, 2012.

UK DEPARTMENT FOR EDUCATION. **The national curriculum in England**: framework document. London: DfE, 2013. Disponível em: <https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/335116/Master_final_national_curriculum_220714.pdf>. Acesso em: 31 jul. 2015.

_____; GOVE, M. **"Harmful" ICT curriculum set to be dropped to make way for rigorous computer science**. 2012. Disponível em: <https://www.gov.uk/government/news/harmful-ict-curriculum-set-to-be-dropped-to-make-way-for-rigorous-computer-science>. Acesso em: 31 jul. 2015.

USA NATIONAL RESEARCH COUNCIL. **Report of a Workshop of Pedagogical Aspects of Computational Thinking**.Washington, D.C.: The National Academies Press. 2011. Disponível em: <http://www.nap.edu/catalog/13170/report-of-a-workshop-on-the-pedagogical-aspects-of-computational-thinking>. Acesso em: 29 jul. 2015.

_____. **Report of a Workshop on the Scope and Nature of Computational Thinking 2010**. Washington, D.C.: The National Academies Press. 2010. Disponível em: <http://www.nap.edu/catalog.php?record_id=12840>. Acesso em: 31 jul. 2015.

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) Departamento de Multimeios, Mídia e Comunicação, Instituto de Artes (IA), Universidade Estadual de Campinas (UNICAMP). Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=000857072&opt=4>. Acesso em: 30 jul. 2015.

_____. (Org) **Computadores na Sociedade do Conhecimento**. Campinas, SP: UNICAMP/NIED, 1999.

_____. **Computadores e Conhecimento** - repensando a educação. Campinas, SP: UNICAMP/NIED, 1993.

_____; BARANAUSKAS, M. C. C; MARTINS, M. C. (Org.). **ABInv Aprendizagem Baseada na Investigação**. Campinas, SP: UNICAMP/NIED, 2014. Disponível em: <http://www.nied.unicamp.br/?q=content/abinv-aprendizagem-baseada-na-investiga%C3%A7%C3%A3o>. Acesso em: 30 jul. 2015.

WHITE HOUSE. **Computer Science is for Everyone!** 2013. Disponível em: <https://www.whitehouse.gov/blog/2013/12/11/computer-science-everyone>. Acesso em: 28 jul. 2015.

WING, J. M. Computational Thinking: what and why. **Thelink**, 2011. Disponível em: http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why. Acesso em: 28 jul. 2015.

_____. Computational thinking. **Communications of the ACM**, v. 49, n. 3, p. 33-35, 2006.

YADAV, A. et al. Computational thinking in elementary and secondary teacher education. **Transactions on Computing Education**, v. 14, n. 1, p. 5-16, March 2014.

_____. Introducing Computational Thinking in Education Courses. **SIGCSE'11**, March 9–12, 2011, Dallas, Texas, USA, 2011.

ZAPATA-ROS, M. Pensamiento computacional: Una nueva alfabetización digital. **Revista de Educación a Distancia**, v. 46, n. 4, p. 1-47, 2015. Disponível em: <http://www.um.es/ead/red/46/zapata.pdf>. Acesso em: 05 ago. 2016.

Artigo recebido em 09/08/2016.
Aceito para publicação em 23/09/2016.

Notas

[i] Entrevista concedida pelo pesquisador Mitchel Resnick, diretor do grupo *Lifelong Kindergarten* do *Media Laboratory*, *Massachusetts Institute of Technology (MIT)*, Cambridge, EUA, à José Armando Valente, em 01.05.2014.

[ii] Algumas simulações registram automaticamente as diferentes tentativas feitas pelos usuários. Quando isso não acontece, ele tem que fazer esse registro manualmente.