Uma Revisão Sistemática de Literatura sobre Autoavaliação de Pensamento Computacional de Jovens

Amilton Rodrigo de Quadros Martins – Fundação IMED / InovaEdu - Laboratório de Ciência e Inovação para Educação – amilton.martins@imed.edu.br

Guilhermina Lobato Miranda - Universidade de Lisboa / Instituto de Educação - gmiranda@ie.ulisboa.pt

Adelmo Antonio da Silva Eloy – Universidade de São Paulo / Escola Politécnica - eloy.adelmo@gmail.com

Resumo: O objetivo deste estudo foi realizar uma Revisão Sistemática de Literatura sobre autoavaliação de Pensamento Computacional de estudantes com idades entre 10 e 15 anos. O foco da revisão foi o da autoavaliação feita por meio de escalas, questionários ou testes de conhecimento. Começamos por fazer uma revisão do conceito de Pensamento Computacional, seguida dos resultados da revisão, contendo trabalhos, métodos, instrumentos utilizados e resultados alcançados. Concluímos que já existem instrumentos validados para medir o Pensamento Computacional na sua multidimensionalidade: uns para avaliar o conhecimento ou a transferência de conhecimento, outros a percepção e atitude dos estudantes face a este construto. Como trabalhos futuros, serão usados os achados para a proposta de um sistema de autoavaliação de Pensamento Computacional.

Palavras-chave: Pensamento Computacional, Avaliação, Educação em Ciência da Computação.

A Systematic Literature Review on Young People's Computational Thought Self-Assessment

Abstract:
This study presents a Systematic Literature Review on self-assessment of Computational Thinking of students aged between 10 and 15 years. The review focused on self-assessment using scales, questionnaires, and proficiency tests. We first review the computational thinking's definition; from that, we detail the systematic review, including methods and research instruments used, key studies identified and a discussion of their findings. We conclude that there already are validated instruments for measuring computational thinking in its multidimensionality: some of them focus on assessing knowledge or knowledge transfer, others target the perception and attitude of students towards this construct. As future work, the findings will be used for the proposal for a computational thinking self-assessment system.
Keywords: Computational Thinking, Assessment, Computer Science Education.

1. Introdução

A defesa pelo ensino e aprendizagem de habilidades de programação desde os anos iniciais da Educação Básica não é recente. Nos anos 60 do séc. XX, Alan Perlis já afirmava que todos deveriam aprender a programar como parte da sua educação e que a Teoria da Computação de Alan Turing, com a execução automatizada de processos por intermédio de uma máquina, mudaria muitos aspetos da atividade humana. Esta necessidade é sustentada na ideia de que a programação é uma forma de explorar o processo de conhecimento (Guzdial, 2008). Esta ideia foi concretizada por Seymour Papert nos anos 80 com o projeto Logo, uma linguagem de programação acessível desde os primeiros anos da escolaridade básica. Naquele momento, a computação começou a ganhar força e invadiu milhares de escolas em todo o mundo (Grover & Pea, 2013).

Em 2006, Jeanette Wing apresentou o conceito de Pensamento Computacional, como uma habilidade humana, essencialmente intelectual e criativa, de criar abstrações do mundo, com foco em resolver problemas, desenhar sistemas e compreender o comportamento humano, cujas soluções poderiam ser materializadas por meio da programação da computadores (Aho, 2012; Wing, 2006, 2008). Desde então, o Pensamento Computacional passou a ser reconhecido como uma habilidade fundamental para o séc. XXI, assim como a leitura, escrita e aritmética (Guzdial, 2008; Wing, 2011, 2012).

Vários outros autores trazem definições complementares para o Pensamento Computacional, conceituando-o, por exemplo, como o processo de pensamento usado para formular problemas, cujas soluções, feitas através de abstrações matemáticas, podem ser representadas como passos e algoritmos computacionais (Aho, 2012), e como base conceitual para resolver problemas de maneira eficaz e eficiente, algoritmicamente - com ou sem computadores -, criando soluções reutilizáveis em diferentes contextos (Shute, Sun, & Asbell-Clarke, 2017).

O Pensamento Computacional usa o raciocínio heurístico para resolver problemas, partindo de uma abstração – inclusive de tempo e espaço – potencialmente generalizável, seja ela uma ideia ou criação, na qual o(a) autor(a) incorpora sua filosofia de vida e curiosidade de forma criativa e única. Se generalizada e modelada, ela poderá ser executada e automatizada por pessoas, máquinas ou uma combinação de ambos. O processo de automatização, por outro lado, requer a representação objetiva de informações iniciais que compõem tal ideia, por meio de dados ou objetos (entendo-os como uma coleção de informações). As informações de entrada passam, então, por um processo de transformação – que podemos chamar de computação – que resulta na saída de informação, que pode ser também entendida como dado processado ou objeto transformado. (Grover & Pea, 2013; Wing, 2006, 2011).

Não devemos confundir, contudo, Pensamento Computacional, que está relacionado com os conceitos de abstração e autoria, com outros conceitos afins como letramento informático ou letramento digital. Estes dizem respeito ao uso técnica e socialmente correto que as pessoas devem fazer das TDIC (Tecnologias Digitais de Informação e Comunicação) no seu cotidiano, como usuários ou consumidores de aplicativos, serviços na internet ou redes sociais (Wing, 2006). É necessário, ainda, fazer uma distinção entre Pensamento Computacional e programação, pois o segundo é apenas parte do primeiro. Pensamento Computacional situa-se na dimensão das ideias, não artefatos. Pensar computacionalmente significa pensar em múltiplos níveis de abstração, enquanto programar é declarar sua abstração para uma máquina de computar. Cada vez que usamos um programa, ele contém materializações de ideias e conceitos dos seus criadores, sua forma de resolver problemas, organizar informações, conversar e interagir com os outros humanos (Wing, 2006, 2008, 2012).

O conceito de abstração, portanto, é fundamental no desenvolvimento do Pensamento Computacional. Abstrações bem definidas devem produzir uma solução correta, eficiente, escalável, com custo baixo, boa estética e elegância. Para chegar a isso, a codificação da solução precisa ser sucinta, declarativa e não ambígua. Para que seja gerenciável, o processo de abstração demanda a decomposição de tarefas complexas em parte menores e solucionáveis, e para isso é preciso modelar adequadamente aspectos relevantes do problema e suas correlações (Wing, 2012).

As abstrações são feitas em camadas, e suas camadas escondem sistemas complexos. Por exemplo, uma tela simples de um caixa eletrônico pode ser usada por um cidadão semianalfabeto, mas esconde um software usando dezenas de servidores e algoritmos distribuídos, camadas de segurança e criptografia, sistemas com inteligência artificial, redes com conexões de fibra ótica submarina que atravessam continentes. As abstrações são feitas para tornarem o mundo complexo simples e acessível. Ainda existem abstrações mais robustas, feitas para profissionais, com várias camadas que permitem o reuso em cascata de sistemas ou partes deles, gerando o encapsulamento de ideias e

tecnologias, simplificando o acesso a equipamentos remotos e algoritmos desconhecidos com as APIs (*Application Programming Interface*), cujo conteúdo do código não conhecemos no detalhe, mas sabemos o que essa abstração deve fazer e nos retornar (Wing, 2008).

O Pensamento Computacional deve, ainda, ser reconhecido como uma habilidade social fundamental para a Sociedade Moderna, não uma habilidade mecânica. Essa habilidade deve ter a capacidade de simplificar atividades do cotidiano, propondo rotinas mecânicas, portanto automatizáveis, que melhoram a vida das pessoas (Grover & Pea, 2013; Wing, 2006). Apesar do nome ser novo, o conceito é muito antigo. Pessoas já computavam, ou seja, executavam tais automações desde os primeiros registros do ser humano na Terra até o séc. XIX. Basta ver os grandes empreendimentos do passado: a agricultura, as pirâmides, as estruturas complexas de civilizações antigas, as redes de comércio por terra ou mar da antiguidade, a organização estratégica dos exércitos da Idade Média e outros exemplos (Castells, 1996). Já fomos, e ainda somos, máquinas de computar ideias abstraídas pelos outros humanos, distribuídos no espaço e no tempo.

Porém, desde meados do séc. XIX, e de forma exponencial no séc. XXI, temos máquinas não humanas – as máquinas autômatas, os computadores, robôs, drones e softwares, capazes de automatizar as ideias das pessoas, ou seja, computar essas ideias. Se no início as automações eram computadas por pessoas, passando depois a ser feitas por máquinas físicas e mecânicas, hoje são realizadas por máquinas elétricas e digitais, e logo serão feitas por máquinas biológicas e quânticas (Wing, 2012).

Com isso, podemos compreender que computar é automatizar nossas abstrações, e nessa lógica, a Ciência da Computação tem a missão de potencializar a Tecnologia, Ciência e Sociedade, devendo ser distribuída e acessível (na concepção ampla da palavra) para todas as pessoas, sejam pobres ou ricas, ou com qualquer limitação (Wing, 2008, 2012). No futuro, poderemos criar abstrações muito profundas, sofisticadas e complexas de sistemas inteiros, simulações da vida com alta fidelidade, analisando dados massivos de forma instantânea. Tal processo envolverá, sem se limitar a, o uso de sons, coordenadas, fotos e vídeos, hoje de nossos celulares, e logo de sensores que estarão por todo o lado, reconhecendo padrões e prevendo acontecimentos de forma muito assertiva (Wing, 2012). A conexão entre a computação, filosofia e a ética já propõe um modelo de pensamento voltado à Moral das Máquinas e seus efeitos na vida dos humanos (García-Peñalvo, 2018).

Em termos de estrutura e evolução do Pensamento Computacional, existem dezenas de propostas para sua conceituação e detalhamento. Neste artigo serão apresentadas as mais citadas e reconhecidas pela comunidade científica, assim como definições mais recentes e complementares às anteriores. Lee et al. (2011), por exemplo, propõem uma progressão, chamada *Use-Modify-Create*, que foi testada com jovens em projetos da NSF (*National Science Foundation*). É baseado na lógica de que as interações devem se aprofundar sucessivamente com suporte apropriado (conceito de *scaffolding*). No estágio *Use*, os alunos jogam ou brincam com a criação de outra pessoa, experimentando-a como consumidores. No *Modify*, eles começam a modificar ou remixar um modelo prévio, propondo níveis crescentes de sofisticação, compreendendo um subconjunto da abstração e automação contida no artefato. Ao iterar aprimoramentos, novas habilidades são desenvolvidas e o que antes era de alguém se torna seu. Com mais confiança, eles passam ao estágio *Create*, onde desenvolvem suas ideias de projeto, a partir de três aspectos: Abstração, Automação e Análise.

Para Brennan e Resnick (2012), com base na perspectiva construcionista de Papert, o Pensamento Computacional tem três dimensões: a) Conceitos computacionais - que as pessoas usam quando programam, recursos como sequências, loops, paralelismo, eventos, condicionais, operadores e dados, b) Práticas computacionais - as práticas usadas à medida que programam, centradas no processo de pensar e aprender, além de "o que" você está aprendendo (conteúdo), até "o como" está aprendendo (metacognitivo), com processos incrementais e iterativos, testes e depurações, reutilização e remixagem, e abstração e

modularização, e c) Perspectivas computacionais - as ideias que as pessoas formam sobre si mesmas e o mundo ao seu redor, como a autoexpressão, conexão e interação, e o questionamento, a partir do domínio de Conceitos e Práticas computacionais

Para Shute, Sun e Asbell-Clarke (2017) que sistematizaram dezenas de investigações sobre o Pensamento Computacional, o mesmo possui seis facetas: A) Decomposição - dividir um problema complexo em partes menores e usar processos sistemáticos para lidar com cada um desses problemas menores, b) Abstração - descobrir padrões dentro de problemas e soluções, extraindo a essência de sistemas complexos, c) Algoritmos - definir sequência de ações para o desenvolvimento de ferramentas e procedimentos reutilizáveis para resolver classes de problemas, d) Depuração – detectar, identificar erros, e corrigir quando uma solução não funcionar como deveria, e) Iteração – repetir os processos de design para refinar as soluções, até que o resultado ideal seja alcançado, e f) Generalização – gerar soluções corretas e eficientes para conjuntos de problemas semelhantes.

Por fim, Kafai, Proctor e Lui (2019) propõem uma visão holística do Pensamento Computacional em três enquadramentos teóricos, semelhantes à Teoria do Círculo Dourado (Simon Sinek, 2011), sendo: a) Cognitivo – trata de habilidades mensuráveis e transferíveis no desenvolvimento de conceitos computacionais como algoritmos, abstração, e práticas como remixagem e iteração (o que fazemos), b) Contextualizado – promove interações sociais, equidade, interesse, desenvolvimento de identidade, criatividade através da criação de aplicativos com significado pessoal, construindo comunidades, apoiando interações sociais (como fazemos), e c) Crítico – aborda o entendimento de justiça e pensamento crítico, busca de mudanças sociais como igualdade de gênero ou raça, por exemplo, criando aplicativos para promover prosperidade, conscientização e ativismo (porque fazemos).

2. Pensamento computacional como construção intelectual e social humana

Autores como Wing (2008) acreditam que o Pensamento Computacional, ou partes dele, está presente na vida de todas as pessoas, pois está ligado a atividades intuitivas e cotidianas. Nesse sentido, se considerarmos o Pensamento Computacional como uma ação humana em direção a um maior grau de abstração e generalização, objetivando resolver problemas do mundo, existe a necessidade de uma profunda análise para a decomposição das partes – compreendendo inclusive as partes iguais e recursivas, e síntese para generalização – e posterior automação em uma máquina de computar (Wing, 2008). Este tipo de procedimento não nos é totalmente estranho, pois fazemos isso no cotidiano explicando a um amigo, por exemplo, como fazer um bolo ou qual é o caminho da nossa casa.

Também não nos é de todo desconhecido o fato de termos de resolver problemas com recursos limitados – como tempo, espaço, energia, que são as restrições do mundo real, e ainda prever uso massivo de dados e múltiplas execuções em lógica de escala, antecipando inclusive alocação de recursos para uso futuro (Wing, 2008). Usamos esta estratégia na nossa vida cotidiana, quando, por exemplo, pensamos o que vamos fazer para o almoço ou jantar, como gerir o dinheiro que temos disponível para este mês ou carregar a bateria do telefone móvel quando vamos fazer um passeio. A prevenção e recuperação de desastres, assim como a redundância, também devem estar presentes no Pensamento Computacional, propondo sempre alternativas com o máximo de disponibilidade de execução da automação. Isso também é natural nas pessoas, pois providenciamos chaves extra de casa e do carro, ou avisamos onde estamos indo como precaução para antevisões de problemas (Wing, 2008).

De forma complementar, chamamos de *Thinker* (Guzdial, 2008) ou *Computational Thinker* (Brennan & Resnick, 2012) aqueles(as) que empreendem esforço para desenvolver o Pensamento Computacional e que podem ser de qualquer área de conhecimento ou profissão. Ensinar processos cognitivos e socioculturais para esse fim é chamado de

educação computacional. O corpo de pesquisa em educação computacional é composto por cientistas da computação, que entendem essa área de conhecimento sob uma perspectiva prática, racional e teórica, e educadores, que trazem um olhar sobre questões inerentemente humanas (Guzdial, 2008).

A computação é a única situação onde não apenas temos conceitos a serem ensinados, mas temos uma máquina a ser ensinada. Os conceitos que são úteis ao Pensamento Computacional também o são à a máquina, instrumento do letramento computacional ou digital, como usuário das abstrações que o primeiro grupo criou (Wing, 2008). Definição parecida é que a computação deve ser vista como sendo composta por duas dimensões: computadores na educação – como ferramenta de apoio à educação e ao letramento digital – e educação para a computação – como ensino do Pensamento Computacional (García-Peñalvo, 2018). Propostas de desenvolvimento com crianças e jovens nessa área mostram que precisamos desenvolver uma educação crítica e reflexiva sobre o mundo digital, que ajude os cidadãos a resolver problemas usando a tecnologia com a qual vivem no seu cotidiano (García-Peñalvo, 2018; Kafai, Proctor e Liu, 2019).

Em relação ao Pensamento Computacional na educação básica (K12), a CSTA (*Computer Science Teachers Association*) compreende que o papel do Pensamento Computacional é servir como uma metodologia transdisciplinar de solução de problemas, que pode ser automatizada e transferível para várias disciplinas. Já o NRC (*National Research Council*) ressalta a importância de uma boa formação prévia dos professores, pois os estudantes aprendem sobre o Pensamento Computacional ao observar os professores modelando estratégias deste tipo de pensamento e, após orientados, utilizando-as com autonomia (Yadav et al., 2015).

Para Vee (2013), a programação é entendida como um novo letramento do séc. XXI, pois leva em conta trajetória histórica e social, recursos para comunicação e interesse cívico. É um novo sistema de escrita do mundo, uma vez que os computadores e ferramentas digitais são centrais na nossa sociedade. Um letramento precisa levar em conta o contexto social, e essencialmente é feito de: a) Escrita – quando as pessoas representam suas ideias em textos ou símbolos (ou a programação), e b) Leitura - quando as pessoas consomem as ideias produzidas por outros (uso de um software, por exemplo), fazendo uso de uma tecnologia simbólica integrada ao cotidiano das pessoas.

Essa relação pode ser compreendida ao se observar que, na Idade Média, apenas grupos de pessoas como padres e escribas podiam ler e escrever, mas o mundo evoluiu e todas pessoas precisavam dessas habilidades. Hoje, estamos em uma revolução análoga, em que a alfabetização digital é agora uma habilidade essencial para ter sucesso em nosso complexo mundo digital do séc. XXI (Shute, Sun, & Asbell-Clarke, 2017).

3. Revisão sistemática de literatura sobre autoavaliação do Pensamento Computacional

Para compreender o estado da arte sobre a avaliação de Pensamento Computacional, foi desenvolvida uma Revisão Sistemática de Literatura, com base em metodologia proposta por Cooper e Hedges (2009). Para isso, foram feitas pesquisas em bases de dados reconhecidas, critérios de inclusão e exclusão foram utilizados para seleção de estudos, e seus resultados foram codificados e sistematizados.

Em janeiro de 2020, foram coletados estudos nas bases ACM *Digital Library*, IEEE *Digital Library*, *Web of Science* e *Google Scholar*. Os termos de pesquisa foram (*+"Computational Thinking" rubric test +"elementary school")* ou (+"Pensamento Computacional" rubrica teste +"ensino fundamental"), pois são as combinações que melhor designam o objetivo da pesquisa. Foram pesquisadas investigações em inglês e português, sendo ao todo encontrados 35 em inglês na ACM *Digital Library*, 1 em inglês na IEEE *Digital Library*, 7 em inglês no *Web of Science*, e 440 em inglês + 28 em português no *Google Scholar*, totalizando 515 trabalhos.

Foram usados critérios de inclusão e exclusão, permanecendo somente artigos que: a) Publicaram resultados nos últimos 5 anos, dada a atualidade do tema, b) Possuem resultados práticos de natureza empírica quantitativa, c) Avaliam aspectos cognitivos ou comportamentais do Pensamento Computacional em jovens de 10 a 15 anos de idade - 5º ao 9º ano do Ensino Fundamental no Brasil, d) Não têm como objetivo avaliar apenas o código ou o artefato criado pelos jovens.

Para codificação dos resultados, foram usados: AU - Nome dos autores e data de publicação, TI – título da publicação, N – número total de sujeitos, AG – idade ou ano escolar dos sujeitos, TY – tipo de instrumento utilizado (Teste, Escala, Questionário), CO – país onde a pesquisa foi feita, ordenando os trabalhos por data de publicação. Seguem na Tabela I, as 11 pesquisas resultantes.

Tabela 1: Resultados da pesquisa

AU	TI	N	AG	TY	CO
(Oluk, Korkmaz, 2016)	*Comparing Students' Scratch Skills with Their Computational Thinking Skills in Terms of Different Variables*	31	5th	E	Turquia
(Jun, Han, Kim, 2017)	*Effect of design-based learning on improving computational thinking*	35	4th-6th	E/Q	Coreia Sul
(Román-González, Moreno-León, Robles, 2017)	*Complementary Tools for Computational Thinking Assessment*	179	7th-8th	T	Espanha
(Brackmann, 2017)	Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica	135	5th -6th	T	Brasil-Espanha
(Weese, 2017)	*Bringing Computational Thinking to K-12 and Higher Education*	91	5th-9th	T/E/Q	EUA
(Román-González, Pérez-González, Jiménez-Fernández, 2017)	*Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test*	1251	5th-10th	T	Espanha
(Chen et al., 2017)	*Assessing elementary students' computational thinking in everyday reasoning and robotics programming*	121	5th	T/E	EUA
(Rijke et al., 2018)	*Computational thinking in primary school: An examination of abstraction and decomposition in different age groups*	200	1th-6th	E	Holanda
(Leifheit, Tsarava, Moeller, 2019)	*Development of a questionnaire on self-concept, motivational beliefs, and attitude towards programming*	31	7-10 anos	E	Alemanha
(Rodríguez-Martínez, González-Calero, Sáez-López, 2019)	*Computational thinking and mathematics using Scratch: an experiment with sixth-grade students*	47	6th	T	Espanha
(Kukul, Karataş, 2019)	*Computational thinking self-efficacy scale: Development, validity and reliability*	319	5th-7th	E	Turquia

4. Resultados e discussão

Oluk e Korkmaz (2016) apresentam uma comparação de dois tipos de testes diferentes com 31 estudantes da 5ª série na Turquia usando um modelo de pesquisa correlacional. Após aulas de programação básica usando o Scratch durante um período de 6 semanas, foram aplicadas duas avaliações distintas: a primeira avaliação foi feita usando a Ferramenta Dr. Scratch e a segunda avaliou habilidades do Pensamento Computacional usando o *Computational Thinking Levels Scale*, composto por: criatividade, resolução de problemas, pensamento algorítmico, colaboração e pensamento crítico. As pontuações

obtidas pelos alunos usando qualquer uma das avaliações não diferiram de acordo com o sexo ou o período de uso do computador, e ainda foi observada uma relação significativa entre as habilidades de programação e de Pensamento Computacional em ambos testes.

Jun, Han e Kim (2017) utilizaram o Scratch, o *Creative Computing Guidebook* e a Aprendizagem Baseada em Design (DBL) em um curso de 15 horas, para potencializar o Pensamento Computacional em 35 estudantes do 4º ano e 20 do 6º ano fundamental da Coreia do Sul. Foram formados grupos experimental e controle, sendo no primeiro utilizada a metodologia DBL e no segundo a metodologia tradicional do curso. Foram aplicados testes de autoeficácia, autointeresse e Pensamento Computacional antes e após o curso. Foi identificado que ambos os grupos apresentaram autoeficácia, autointeresse e Pensamento Computacional aprimorados, mas o grupo experimental teve um número maior de alterações positivas e significativas do que o grupo controle em todas as áreas, demonstrando que a aplicação do DBL é mais eficaz para desenvolver Pensamento Computacional do que as metodologias tradicionais.

Román-González, Moreno-León e Robles (2017) sumarizaram os tipos de avaliação de Pensamento Computacional em: a) Ferramentas Somativas – em geral testes de aptidão e conhecimento como o *Computational Thinking Test* (CTt), o *Test for Measuring Basic Programming Abilities* e o *Commutative Assessment Test*, entre outros de conhecimento de conteúdos computacionais; b) Ferramentas Formativo iterativas, que fornecem feedback, geralmente de forma automática nos ambientes de programação, como *Dr. Scratch, Ninja Code Village, Computational Thinking Patterns* e *CTP-Graph*; c) Ferramentas de Transferência de Habilidades – que visam avaliar a transferência de habilidades dos alunos para outros tipos de problemas do cotidiano, como as *Bebras Tasks* ou *CTP-Quiz*; d) Escalas de Percepções e Atitudes - como a *Computational Thinking Levels Scale* (Korkmaz, Çakir, e Özden, 2017); e) Avaliações de vocabulário – que visam medir elementos e dimensões expressos verbalmente na linguagem das crianças. Os autores ressaltam que o uso de apenas um indicador de avaliação pode ser impreciso, necessitando de um Sistema de Avaliação combinado. Na pesquisa feita na Espanha, foram feitos dois estudos: O primeiro com 179 estudantes de 7º e 8º anos, onde foi feita uma intervenção usando o *CODE.org*, com objetivo de avaliação de convergência entre o *Computational Thinking Test* (CTt) e o *Bebras Task*, com aplicação pré e pós dos testes. O segundo estudo, com 71 estudantes da 8ª série, onde foi feita uma intervenção de 8 semanas com o Scratch, sendo a análise entre o *Computational Thinking Test* (CTt) e o Dr. Scratch, com pré e pós aplicação do CTt, e pós aplicação do *Dr. Scratch* no projeto final dos estudantes. Os resultados mostraram que a convergência dos resultados não foi total, pois mesmo avaliando o mesmo constructo psicológico, foram feitos em três perspectivas diferentes: a) Somativa, b) Formativo iterativas e c) Transferência de habilidades. Este resultado aponta que nenhuma das avaliações pode ser substituída por outra, sendo as mesmas complementares, cobrindo todos os níveis da Taxonomia de Bloom.

O estudo de Brackmann (2017) teve como objetivo desenvolver o Pensamento Computacional na Educação Básica, por meio de 10 horas de atividades exclusivamente “desplugadas” (sem o uso de computadores) com estudantes de 5º e 6º anos, sendo 63 no Brasil e 72 na Espanha. As avaliações usaram o *Computational Thinking Test* (CTt) (Román-González et al., 2017) em pré e pós teste com grupo experimental, que fez as atividades desplugadas, e grupo controle que não fez. Os resultados obtidos através de uma abordagem quase experimental apresentaram uma melhoria significativa no desempenho dos estudantes que tiveram atividades de Pensamento Computacional “desplugadas” em ambos os países.

Weese (2017) descreve diferentes intervenções em um programa de extensão STEM (*Science, Technology, Engineering e Mathematics*) para alunos da 5ª à 9ª série com o Scratch, todas com o objetivo de melhorar a autoeficácia na ciência da computação e Pensamento Computacional, sendo avaliados por um questionário criado pelo autor. O instrumento foi aplicado aos estudantes pré e pós intervenções em um grupo experimental e

grupo controle, que não participou dos cursos. Os resultados apresentados indicam significativa melhoria na percepção de autoeficácia dos estudantes, principalmente dos que participaram de cursos de extensão STEM, se comparados com os que tiveram atividades curriculares na escola.

Román-González, Pérez-González, e Jiménez-Fernández (2017) aplicaram o *Computational Thinking Test* (CTt) em 1.251 estudantes espanhóis da 5ª à 10ª série, que participaram da disciplina eletiva de computação nas suas escolas. A pesquisa apresenta estatística descritiva, evidências de confiabilidade e validade de critério do CTt, comparando com outros testes psicológicos: a bateria de Habilidades Mentais Primárias (PMA) e o teste de solução de problemas RP30. Os resultados mostraram correlações estatisticamente significativas entre Pensamento Computacional e habilidade espacial, capacidade de raciocínio e capacidade de resolução de problemas. Esses resultados são consistentes com as teorias que vinculam o Pensamento Computacional a alguns componentes do modelo de inteligência *Cattel-Horn-Carroll*, e corroboram a ideia de que Pensamento Computacional é uma capacidade para a resolução de problemas.

Chin et al. (2017) propõem um instrumento para avaliar o Pensamento Computacional baseado em rubricas, aplicado a 121 estudantes do 5º ano de uma escola pública dos EUA, que participaram de 6 meses de aulas semanais de robótica. Os itens foram divididos em codificação em robótica e raciocínio de eventos cotidianos, sendo aplicado pré e pós testes, onde um novo currículo de robótica humanoide foi adotado. Os resultados mostraram que o instrumento possui boas propriedades psicométricas e tem o potencial de revelar desafios de aprendizagem dos alunos e crescimento em termos de Pensamento Computacional.

Rijke et al. (2018) avaliaram o Pensamento Computacional em 200 estudantes do ensino fundamental, com idades compreendidas entre 6 e 12 anos na Holanda, com foco em abstração e decomposição, em atividades de computação “desplugada”. A escala foi criada pelos autores, baseada nos construtos de dificuldade percebida, carga cognitiva, fluxo, percepções e experiências, sendo aplicada no final das atividades. Foi constatado que a idade parece estar relacionada com o conhecimento em abstração e decomposição, com um efeito de interação por gênero na tarefa de abstração, porém não foram encontradas diferenças entre estudantes jovens e mais velhos nos construtos dificuldade percebida, carga cognitiva e fluxo.

Leifheit et al. (2019) realizaram um estudo piloto usando o *Computational Thinking Test* (CTt) e um questionário próprio para medir o autoconceito, crenças motivacionais e atitudes em relação à programação, com 31 estudantes de 7 a 10 anos de idade na Alemanha, participantes de um curso extracurricular de Pensamento Computacional de 10 encontros de 90 minutos cada. O curso utilizou jogos de tabuleiro como atividades de computação desplugadas, além do Scratch, plataforma Arduino e o Open Roberta Lab. Os resultados mostraram associações apenas entre os aspectos avaliados pelo questionário, mas não em relação aos resultados do *Computational Thinking Test* (CTt).

Rodríguez-Martínez, González-Calero e Sáez-López (2019) apresentaram uma pesquisa quase experimental com 47 estudantes de 6º ano fundamental na Espanha, contendo grupos experimental e de controle, em duas fases: uma focada no Pensamento Computacional e outra na resolução de tarefas matemáticas. Na etapa inicial da pesquisa, ambos grupos aprenderam programação básica, mas o experimental resolveu tarefas matemáticas com Scratch e o controle sem o Scratch. Foram aplicados o *Computational Thinking Test* (CTt) e o *Mathematical Knowledge Test* (MKT) antes e após intervenção com os dois grupos, e os resultados indicam que o Scratch tem grande potencial, tanto para desenvolver ideias matemáticas, quanto o Pensamento Computacional dos estudantes.

Finalmente, Kukul e Karataş (2019) desenvolveram uma escala composta por 18 itens sob quatro fatores: Raciocínio, Abstração, Decomposição e Generalização, para medição de autoeficácia ligada ao Pensamento Computacional, aplicada em 319 estudantes de 5º a 7º ano na Turquia. Todos estudantes tinham algum tipo de conhecimento prévio em

ferramentas como Scratch, *CODE.org*, *Small Basic* e Alice. Tanto os resultados da análise de confiabilidade aplicada, quanto da análise fatorial confirmatória e os índices de ajuste foram verificados, sendo considerados adequados e aceitáveis, tornando a *Computational Thinking Self-efficacy Scale* uma ferramenta válida e confiável para medição de autoeficácia ligada ao Pensamento Computacional.

5. Considerações finais

Apesar de ser um conceito recente e com definições em construção e debate por pesquisadores, o Pensamento Computacional é uma habilidade que explicita o protagonismo humano na interação com as máquinas de computar, assim como o potencial positivo dessas interações. Nesse sentido, tornam-se essenciais não só o avanço no seu entendimento em termos técnicos, derivados da Ciência da Computação, como seu impacto social e importância para formação de cidadãos autônomos num mundo cada vez mais digital. Como consequência, a pesquisa em torno do desenvolvimento do pensamento computacional em crianças e jovens, onde se situa o estudo apresentado, é vital para que oportunidades educacionais sejam desenhadas e ofertadas de forma intencional e contextualizada.

Como contribuição para o tema, este estudo partiu de 515 trabalhos, e mapeou 11 que evidenciaram formas de intervir e avaliar o desenvolvimento do Pensamento Computacional, explorando especialmente instrumentos de avaliação utilizados pelos próprios aprendizes. Em síntese, quanto aos estudos, pôde-se verificar que: 1) Foram realizadas em 7 países diferentes, com destaque para a Espanha com 4 trabalhos; 2) Utilizaram intervenções com computação “desplugada”, programação e robótica, com destaque para 7 trabalhos com Scratch; 3) Utilizaram vários instrumentos de avaliação, autorais e de terceiros, com destaque para 5 trabalhos que utilizaram o *Computational Thinking Test* (CTt).

Como resultado dessa pesquisa, foi possível verificar que já existem instrumentos autoaplicáveis validados e prontos para serem utilizados em avaliações de Pensamento Computacional de estudantes de 10 a 15 anos, com validação estatística e foco em avaliar conhecimento, transferência de conhecimento, percepção e atitude dos estudantes nesse tema. Como possibilidades de trabalho futuro, destacam-se a replicação das intervenções mapeadas em outros contextos, para validação de resultados e modelagem de soluções escaláveis e adaptação ou tradução de testes validados em contextos específicos, ampliando o seu potencial de aplicação em cenários mais amplos.

Por fim, a definição de bases de estudos e a velocidade de produção acadêmica sobre Pensamento Computacional representam possível limitações desse trabalho, especialmente no que se refere à representação do cenário de estratégias de avaliação dessa habilidade. Contudo, entende-se que ele sirva de referência para que pesquisadores – da Ciência da Computação e da Educação, da academia e da escola – desenham e afiram intervenções em Pensamento Computacional com base em evidências.

6. Referências

AHO, Alfred V. Computation and computational thinking. **The Computer Journal**, v. 55, n. 7, p. 832-835, 2012.

BRACKMANN, Christian Puhlmann. Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica. 2017. Retirado de:https://lume.ufrgs.br/handle/10183/172208.

BRENNAN, Karen; RESNICK, Mitchel. 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**. 2012. p. 25.

CASTELLS, M. A sociedade em rede [portuguese translation of The Rise of the Network Society]. **S o Paulo, Paz e Terra.. S o Paulo, Paz e Terra**, 1996.

CHEN, Guanhua, et al. Assessing elementary students' computational thinking in everyday reasoning and robotics programming." **Computers & Education** 109, p. 162-175, 2017.
MARCOS-GARCÍA, José-Antonio; MARTÍNEZ-MONÉS, Alejandra; DIMITRIADIS, Yannis. DESPRO: A method based on roles to provide collaboration analysis support adapted to the participants in CSCL situations. **Computers & Education**, v. 82, p. 335-353, 2015.
HEDGES, L. V.; COOPER, H. Research synthesis as a scientific process. **The handbook of research synthesis and meta-analysis**, v. 1, 2009.
LEE, Irene et al. Computational thinking for youth in practice. **Acm Inroads**, v. 2, n. 1, p. 32-37, 2011.
GARCÍA-PEÑALVO, Francisco J. Pensamiento computacional. 2018. Retirado de: https://repositorio.grial.eu/handle/grial/1182.
GROVER, Shuchi; PEA, Roy. Computational thinking in K–12: A review of the state of the field. **Educational researcher**, v. 42, n. 1, p. 38-43, 2013.
GUZDIAL, M. Education: Paving the way for CT. **Communications of the ACM**, v. 51, n. 8, p. 25-27, 2008.
JUN, SooJin; HAN, SeonKwan; KIM, SooHwan. Effect of design-based learning on improving computational thinking. **Behaviour & Information Technology**, v. 36, n. 1, p. 43-53, 2017.
KAFAI, Yasmin; PROCTOR, Chris; LUI, Debora. From Theory Bias to Theory Dialogue: Embracing Cognitive, Situated, and Critical Framings of Computational Thinking in K-12 CS Education. In: **Proceedings of the 2019 ACM Conference on International Computing Education Research**. 2019. p. 101-109.
KORKMAZ, Özgen; CAKIR, Recep; ÖZDEN, M. Yaşar. A validity and reliability study of the computational thinking scales (CTS). **Computers in Human Behavior**, v. 72, p. 558-569, 2017.
KUKUL, Volkan; KARATAS, Serçin. Computational Thinking Self-Efficacy Scale: Development, Validity and Reliability. **Informatics in Education**, v. 18, n. 1, p. 151-164, 2019.
LEIFHEIT, Luzia et al. Development of a Questionnaire on Self-concept, Motivational Beliefs, and Attitude Towards Programming. In: **Proceedings of the 14th Workshop in Primary and Secondary Computing Education**. 2019. p. 1-9.
OLUK, Ali; KORKMAZ, Özgen. Comparing Students' Scratch Skills with Their Computational Thinking Skills in Terms of Different Variables. **Online Submission**, v. 8, n. 11, p. 1-7, 2016.
RIJKE, Wouter J. et al. Computational Thinking in Primary School: An Examination of Abstraction and Decom-position in Different Age Groups. **Informatics in education**, v. 17, n. 1, p. 77, 2018.
RODRÍGUEZ-MARTÍNEZ, José Antonio; GONZÁLEZ-CALERO, José Antonio; SÁEZ-LÓPEZ, José Manuel. Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. **Interactive Learning Environments**, v. 28, n. 3, p. 316-327, 2020.
ROMÁN-GONZÁLEZ, Marcos; MORENO-LEÓN, Jesús; ROBLES, Gregorio. Complementary tools for computational thinking assessment. In: **Proceedings of International Conference on Computational Thinking Education (CTE 2017), S. C Kong, J Sheldon, and K. Y Li (Eds.). The Education University of Hong Kong**. 2017. p. 154-159.
ROMÁN-GONZÁLEZ, Marcos; PÉREZ-GONZÁLEZ, Juan-Carlos; JIMÉNEZ-FERNÁNDEZ, Carmen. Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. **Computers in Human Behavior**, v. 72, p. 678-691, 2017.
SHUTE, Valerie J.; SUN, Chen; ASBELL-CLARKE, Jodi. Demystifying computational thinking. **Educational Research Review**, v. 22, p. 142-158, 2017.
SINEK, Simon. Start with Why: How Great Leaders Inspire Everyone to Take Action. **Penguin USA**, 256, 2011.
VEE, Annette. Understanding computer programming as a literacy. **Literacy in Composition Studies**, v. 1, n. 2, p. 42-64, 2013.
WEESE, Joshua Levi. **Bringing computational thinking to K-12 and higher education**. 2017. Tese de Doutorado. Kansas State University.

WING, Jeannette M. Computational thinking. **Communications of the ACM**, v. 49, n. 3, p. 33-35, 2006.
WING, Jeannette M. Computational thinking and thinking about computing. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, v. 366, n. 1881, p. 3717-3725, 2008.
WING, Jeannette. **Research** notebook: computational thinking—what and why? The Link Magazine, Spring. **Carnegie Mellon University, Pittsburgh. Retrieved**, v. 1, p. 2019, 2011.
WING, Jeannette. Computational Thinking. *Microsoft Asia Faculty Summit*, 2012.
YADAV, Aman et al. Computational thinking in elementary and secondary teacher education. **ACM Transactions on Computing Education (TOCE)**, v. 14, n. 1, p. 1-16, 2014.