Revista Brasileira de Informática na Educação – RBIE
Brazilian Journal of Computers in Education
(ISSN online: 2317-6121; print: 1414-5685)
http://br-ie.org/pub/index.php/rbie

Submission: 13/06/2019; 1st round notif.: 17/11/2019; New version: 13/12/2019; 2nd round notif.: 24/03/2020
Camera ready: 25/03/2020; Edition review: 25/03/2020; Available online: 02/06/2020; Published: 02/06/2020

Formação Continuada de Professores para o Ensino de Algoritmos e Programação na Educação Básica: Um Estudo de Mapeamento Sistemático

Professional Teacher Development for Teaching Algorithms and Programming in K-12: A Systematic Mapping Study

Fabióla M. Kretzer
Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
fabiola.kretzer@grad.ufsc.br

Christiane Gresse von Wangenheim
Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
c.wangenheim@ufsc.br

Jean C. R. Hauck
Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
jean.hauck@ufsc.br

Fernando S. Pacheco
Instituto Federal de Santa Catarina (IFSC)
fspacheco@ifsc.edu.br

Resumo

Atualmente, a computação mostra-se cada vez mais influente no nosso dia-a-dia. Devido a essa importância, o ensino da computação deveria ser iniciado já na Educação Básica. Essa integração do ensino de computação na escola requer professores formados, no entanto, atualmente poucos se formam em cursos de licenciatura em computação. Por isso, algumas iniciativas visam integrar a formação em computação de maneira interdisciplinar, voltada a professores de outras áreas por meio de formação continuada. Neste contexto, o artigo apresenta um mapeamento sistemático visando a identificação de unidades instrucionais (UI) existentes para formação continuada de professores e suas características. Como resultado, foram encontradas 16 UI. A maioria das UI se concentra no ensino de algoritmos e programação, com poucas cobrindo também o conhecimento pedagógico e tecnológico. As UI encontradas variam amplamente em termos de modo de ensino e duração. Observa-se que, devido à falta de informações sobre como as UI foram desenvolvidas, há oportunidades de pesquisas futuras nessa área. No entanto, os resultados demonstram também primeiros indícios da viabilidade e da contribuição positiva da formação continuada de professores para o ensino de computação, contribuindo para a popularização dessas competências.
Palavras-Chave: *computação; aprendizagem; formação de professores; Educação Básica*

Abstract

Today, computing has become increasingly influential in our day-to-day lives. Due to this importance, computing needs to be popularized in K-12 giving students the opportunity to learn basic computing competencies. Integrating computer education into school requires trained teachers, however, currently such teachers are scarce. Therefore, some initiatives aim at integrating computer education in an interdisciplinary way, training in-service teachers from other areas. In this context, this article presents the result of a systematic mapping study aiming on the identification of existing instructional units for training in-service teachers and their characteristics. As a result, 16 instructional units (IUs) were found. Most UIs focus on teaching algorithms and programming, with few also covering pedagogical and technological knowledge. The encountered IUs vary

Cite as: Kretzer, F. M., Gresse von Wangenheim, C, Hauck, J. C. R. & Pacheco, F. S. (2020). Professional Teacher Development for Teaching Algorithms and Programming in K-12: A Systematic Mapping Study (Formação Continuada de Professores para o Ensino de Algoritmos e Programação na Educação Básica: Um Estudo de Mapeamento Sistemático). Brazilian Journal of Computers in Education (Revista Brasileira de Informática na Educação - RBIE), 28, 389-419. DOI: 10.5753/RBIE.2020.28.0.389

largely in terms of teaching mode, duration. We also observed, due to a scarcity of information on how the IUs were developed the necessity for further research in this area. However, the results demonstrate also first indications of the viability and the positive contribution of in-service teacher training for computer education, contributing to the popularization of computing competencies.
Keywords: *computing; learning; teacher training; K-12*

1 Introdução

Os avanços tecnológicos estão mudando as sociedades do século XXI, trazendo numerosos desafios para a Educação Básica (CSTA, 2016). A velocidade de tais avanços associada ao número crescente de profissões que dependem da computação vem exigindo que todos os alunos da Educação Básica possam se tornar não apenas alfabetizados em Tecnologia da Informação (TI), mas tenham a oportunidade de aprender de forma independente e usar tecnologia a medida que ela evolui, incluindo o uso ativo da computação e a criação de artefatos de TI (CSTA, 2011). Conceitos e práticas de computação são considerados importantes para os estudantes, os quais utilizam a computação em sua vida cotidiana e na economia global (Grover & Pea, 2013; Barr & Stephenson, 2011; Barr et al., 2011). Ao proporcionar oportunidades de aprendizado de computação na escola, estimula-se o processo de criação, inovação e adaptação em um ambiente de mudanças rápidas, contínuas e fundamentais (Naughton, 2012; Google & Gallup, 2015). O ensino de computação é também considerado uma forma de operacionalizar a aprendizagem do pensamento computacional, que se refere aos procedimentos de pensamento envolvidos na criação de soluções algorítmicas, ou passo-a-passo, que podem ser executadas por um computador (Wing, 2006; Romero et al., 2017; Selby, 2012).

Alguns dos principais conceitos que fazem parte do ensino de computação envolvem algoritmos e programação (CSTA, 2016; Romero et al. 2017; Chen et al. 2017). Algoritmos são sequências de passos em uma ordem definida para resolver problemas ou completar uma tarefa. Eles podem ser automatizados usando linguagens de programação, as quais possibilitam que os sistemas computacionais entendam e executem partes ou todo o algoritmo proposto (SBC, 2017; Yadav, 2016). Os conceitos de algoritmos e programação envolve as habilidades de resolução de problemas e controle de todos os sistemas de computação (CSTA, 2016). Essas habilidades, são típicas do pensamento computacional, assim, algoritmos e programação servem como possível meio para ensinar o pensamento computacional (Selby, 2012; CSTA, 2016).

Atualmente, as escolas estão começando a incorporar o ensino de conceitos e práticas de algoritmos e programação por meio de desenvolvimento de jogos, aplicativos ou robótica com linguagens de programação visuais baseadas em blocos, como Scratch (MITb, 2019) e App Inventor (MITa, 2019). Embora vários recursos estejam se tornando disponíveis (incluindo diretrizes curriculares, planos de aula e materiais de ensino, ambientes de programação baseados em blocos, etc.) (Grover & Pea, 2013), em geral, as escolas continuam enfrentando desafios na preparação de um número suficiente de professores formados para atender a crescente necessidade de ensino de computação (Goode, 2007; INEP, 2018).

A implementação de ensino de computação eficaz na escola exige professores motivados, dedicados e com competências computacionais, pedagógicas e tecnológicas, a fim de ensinar computação de uma forma que realmente envolva os alunos (Gal-Ezer & Stephenson, 2010; Goode, 2008; Bower, 2017). Idealmente, os professores de computação seriam preparados por meio de um abrangente programa de formação de professores, normalmente por meio de programas de graduação ou pós-graduação (Berges et al., 2013) como cursos de licenciatura em computação (INEP, 2018). Embora tais programas ofereçam aos professores uma compreensão profunda da disciplina, essa solução tem se mostrado um

caminho lento que requer vários anos para ser concluído (Granor et al., 2016). As matrículas nesses programas são também cada vez mais reduzidas pelo crescimento contínuo do emprego na indústria, já que o incentivo para seguir carreiras lucrativas na indústria de TI é muito maior do que o incentivo para a licenciatura (Goode, 2007). Por exemplo, em 2018 no Brasil o número de concluintes em cursos para formação de professores de computação foi de 0,1% do total de concluintes em cursos de formação de professores, muito menor do que de outras áreas, como por exemplo matemática, que possui 10% do total de concluintes (INEP, 2018). Como resultado, professores formados no ensino de computação no nível da Educação Básica são escassos (Ni & Guzdial, 2012; INEP, 2018).

Uma alternativa para implementar o ensino de computação em maior escala na Educação Básica pode ser a integração interdisciplinar (Lye & Koh, 2014; Gresse von Wangenheim et al., 2017). Portanto, é importante recrutar e formar um corpo substancial de professores da Educação Básica mesmo que formados em outras áreas (Goode, 2008; Cooper et al., 2015). Consequentemente, é recomendado oferecer a formação continuada para criar condições no curto prazo para a popularização da computação em escolas (Lamprou et al., 2017; Granor et al., 2016).

A formação continuada é entendida como um componente essencial da profissionalização, abrangendo os diferentes saberes e as experiências profissionais dos professores (Brasil, 2016). Esses professores têm conhecimento de conteúdo da área de estudo que ensinam (por exemplo, história, artes, ciências, etc.), conhecimento pedagógico geral, bem como conhecimentos e habilidades profissionais gerais, mas normalmente não possuem competências em computação (Gresse von Wangenheim et al., 2017). Isso significa que existe uma lacuna de competência que pode influenciar significativamente na implementação do ensino de computação afetando o sucesso na aprendizagem do aluno. A fim de ensinar computação de maneira eficaz e envolvente, os professores precisam ter conhecimento de conteúdo em computação (especialmente algoritmos e programação), conhecimento de conteúdo pedagógico (como ajudar os alunos a aprender computação) e também conhecimento tecnológico (para instalar e manter a infraestrutura de TI necessária) (ISTE, 2019). Se os professores não se sentirem preparados para ensinar computação, os alunos podem ter experiências negativas de aprendizagem (Bower et al., 2017). Isso é ainda mais crítico, pois mudanças podem se tornar intimidantes para professores que não possuem formação específica em computação (Google & Gallup, 2016), criando uma barreira de resistência. Além disso, as percepções de computação dos professores irão influenciar diretamente a forma de ensinar algoritmos e conceitos de programação (Milton et al., 2007).

A preparação de professores para facilitar a instrução é crucial para proporcionar aos alunos o acesso a uma educação de qualidade (Gal-Ezer & Stephenson, 2010). Os professores precisam ter um amplo conhecimento não somente de conteúdo técnico e tecnologia computacional (CSTA, 2016), mas também de conteúdo pedagógico para poderem incorporar o ensino de competências de computação de maneira eficaz em suas disciplinas (Gal-Ezer & Stephenson, 2010; Lye & Koh, 2014). Ao enfrentar esses desafios de preparar e apoiar os professores da Educação Básica, uma questão que surge são os tipos de programas de formação continuada existentes em computação para professores atuantes em outras disciplinas. No caso específico do Brasil, faz parte da Política Nacional de Formação de Profissionais da Educação Básica "promover a atualização teórico-metodológica nos processos de formação dos profissionais do magistério, inclusive no que se refere ao uso das tecnologias de comunicação e informação nos processos educativos" (Brasil, 2016).

No entanto, as revisões de literatura existentes focalizam programas de preparação de professores (de outras disciplinas) ainda não atuantes em sala de aula (Armoni, 2011). Outras revisões fornecem somente um resumo de vários programas de formação de professores

como parte de uma visão de diversos programas de forma geral (Mannila et al., 2014). Algumas pesquisas relatam o estado da prática, mas são limitadas a países específicos (Partanen et al., 2016), tipos específicos de programas, como de universidades, ou o modo de instrução, como os MOOC (*Massive Open Online Courses*) (Spradling et al., 2015). Lockwood e Mooney (2017) apresentam um resumo de várias oficinas de professores publicadas até 2016, porém não fornecem um mapeamento sistemático e detalhado de suas características, pois o estudo se concentra em um escopo mais amplo do que somente a formação de professores. Como resultado, as descrições de programas de preparação de professores (atuantes em outras disciplinas) para computação são raras na literatura e, em alguns casos, obsoletas, faltando detalhes sobre suas características específicas.

Nesse contexto, o presente artigo apresenta um mapeamento sistemático das características dos programas de formação continuada para ensino de computação na Educação Básica (especialmente algoritmos e programação com linguagens de programação baseadas em blocos) destinados a professores atuantes em outras disciplinas. Os resultados do nosso estudo podem ajudar responsáveis por programas de formação continuada a selecionar, desenvolver ou melhorar programas, bem como orientar desenvolvedores de currículos. Também esperamos que a discussão possa apoiar ainda mais a preparação de professores comprometidos com a incorporação do ensino de computação na Educação Básica.

2 Fundamentação Teórica

2.1 Conteúdo da Formação Continuada

Com foco na formação continuada de professores atuantes em outras disciplinas e com o objetivo de incorporar o ensino de computação de forma interdisciplinar, assume-se que os professores já possuem conhecimento sobre assuntos específicos (como estudos sociais, ciências, artes etc.), conhecimento pedagógico geral, assim como experiência no ensino de sua área disciplinar na Educação Básica. No entanto, é importante que os professores sejam preparados de forma abrangente para apoiar o ensino de computação. Isso inclui não apenas o conhecimento de conteúdo de computação, mas também o conhecimento pedagógico de conteúdo e tecnologia, com o objetivo de ajudar os alunos a aprenderem a computação de uma maneira eficaz e envolvente (ISTE, 2019). Além disso, ao compreender a integração do ensino em computação de uma forma interdisciplinar, a computação não deve ser vista como um *add-on*, mas possuir foco nas conexões entre tecnologia, conteúdo da área específica e pedagogia enquanto atuam no contexto de sala de aula (Harris et al., 2009). Portanto, todos esses aspectos podem ser abordados de forma integrada, para preparar e apoiar os professores (Harris et al., 2009).

2.1.1 Conhecimento de conteúdo sobre computação

Para possibilitar o ensino de computação é importante ter conhecimento sobre os principais conceitos e práticas de computação que normalmente são abordados no ensino de computação na Educação Básica (Mishra & Koehler, 2006; CSTA, 2016). Isso inclui vários conceitos e práticas fundamentais, os quais devem estar de acordo com o contexto e os objetivos de aprendizagem, conforme apresentado na Tabela 1, seguindo a Estrutura da Educação Básica em Ciência da Computação (CSTA, 2016).

Tabela 1: Conceitos e práticas fundamentais no ensino de computação (CSTA, 2016).

Conceitos Fundamentais	**Prática Fundamentais**
Sistemas de computador	Promover uma cultura de computação inclusiva
Redes e Internet	Colaborar na computação
Dados e Análise	Reconhecer e definir problemas computacionais
Impactos do computador	Desenvolver e usar abstrações
Algoritmos e Programação	Criar artefatos computacionais
	Testar e refinar artefatos computacionais
	Comunicar sobre computação

Assim, foca-se tipicamente em algoritmos e programação, ensinando os alunos a programar vários tipos de software, tais como jogos, animações ou aplicativos móveis (Lye & Koh, 2014). Neste sentido, ambientes de programação baseados em blocos, como Scratch (MITb, 2019), Alice (Alice, 2019) ou App Inventor (MITa, 2019) são geralmente usados com alunos iniciantes. Essas linguagens de programação baseadas em blocos motivam o aprendizado de conceitos de programação, concentrando-se na lógica e nas estruturas envolvidas na programação, não exigindo a aprendizagem de sintaxe e semântica necessárias em linguagens de programação textual, auxiliando e, assim, reduzindo a carga cognitiva dos estudantes (Kelleher & Pausch, 2005).

Desta forma, durante a formação, é importante que os professores desenvolvam competências de computação que abranjam pelo menos esses conceitos e práticas fundamentais, incluindo especialmente as competências em algoritmos e programação e, assim desenvolver sistematicamente artefatos computacionais usando linguagens de programação. Para poder ensinar essas competências, indica-se que os professores alcancem esses resultados de aprendizagem no nível de aplicação com relação a Taxonomia de Bloom (Bloom et al., 1964), atingindo pelo menos o estágio de competência de acordo com o Modelo Dreyfus de Aquisição de Habilidades (Dreyfus & Dreyfus, 1981).

2.1.2 Conhecimento pedagógico e do conteúdo

Ao focar nos professores atuantes em sala de aula, os quais concluíram sua formação pedagógica em outra área, assume-se que esses profissionais já possuem uma compreensão dos objetivos educacionais e estratégias pedagógicas em geral, bem como experiências práticas aplicando-os em sua área de atuação. No entanto, considerando as características específicas de cada área temática, os professores também precisam de conhecimento pedagógico e do conteúdo específico para planejar e ensinar computação usando estratégias e métodos eficazes. Isso também inclui uma compreensão dos currículos de computação para a Educação Básica e respectivos objetivos de aprendizagem.

Professores eficazes devem conhecer uma gama de estratégias pedagógicas destinadas a ajudar os alunos a aprender sobre um determinado assunto. O ensino de computação na Educação Básica inclui diversas estratégias pedagógicas, variando de instrução direta (por exemplo, aulas expositivas) a estudos independentes (Guzdial, 2008). De acordo com os objetivos de aprendizagem que visam ensinar a aplicação de algoritmos e conceitos de programação, observa-se uma predominância de estratégias ativas de aprendizagem, o que permite aos alunos aplicar as competências a serem aprendidas. Essas estratégias incluem exercícios, como o desenvolvimento de código para problemas bem definidos, a adoção de abordagens construtivistas como aprendizagem situada, aprendizado baseado em projetos, entre outros, lidando com problemas complexos, abertos e mal definidos de maneira criativa (Kelleher & Pausch, 2005). Também é importante que os professores entendam quais tópicos os alunos consideram fáceis ou difíceis de aprender, quais ideias (muitas vezes equivocadas) os alunos levam consigo para a sala de aula e como ajudá-los a resolver suas

dificuldades e equívocos. Além disso, os professores necessitam entender como os alunos desenvolvem e aprendem conceitos de computação (Mishra & Koehler, 2006).

Com a visão do aprendizado de programação como parte de um processo sistemático de desenvolvimento de *software*, indica-se aos professores saberem como apoiar os alunos a aprender práticas essenciais, tais como criar, testar e refinar artefatos computacionais. Considerando a relevância das habilidades de colaboração e comunicação nesse contexto, os professores também necessitam de conhecimento das práticas pedagógicas que apoiem o desenvolvimento dessas habilidades, como a programação em pares, por exemplo.

As diferentes estratégias de instrução necessitam diversos tipos de materiais instrucionais, incluindo artefatos de *software* (por exemplo, especificação de requisitos de um sistema de software pré-definido, casos de uso, histórias de usuário, exemplos de código), folhas de exercícios, slides, vídeos, entre outros (Lye & Koh, 2014). Assim, os professores também devem possuir a oportunidade de saber encontrar material instrucional disponível, como selecionar o mais adequado e/ou desenvolver o seu próprio.

Os professores também possuem o papel de avaliar a aprendizagem dos alunos. A avaliação da aprendizagem do aluno é geralmente feita pelo professor usando diversos métodos, como observações, questionários, entrevistas, etc. A fim de avaliar as atribuições de programação, normalmente, as avaliações baseadas no desempenho são adotadas analisando de forma manual ou automaticamente os artefatos (por exemplo, *software*) criados pelos alunos (Gresse von Wangenheim et al., 2017; Moreno-Leon & Robles, 2015). No contexto das abordagens de aprendizagem baseadas em jogos, as pontuações do jogo também podem ser usadas para avaliação (Rusu et al., 2011). A avaliação por pares é uma outra maneira, na qual os artefatos criados pelos alunos são avaliados por seus próprios pares (De Kereki & Manataki, 2016).

2.1.3 Conhecimento tecnológico

O ensino em computação normalmente envolve o uso de uma infraestrutura tecnológica. Isso inclui *hardware* de computador, computadores e *hardware* específico, como parte de projetos de robótica (como microprocessadores, sensores, servomotores) (Mishra & Koehler, 2006; Gal-Ezer & Stephenson, 2010). Assim, um pré-requisito para o ensino de programação é a preparação dos respectivos ambientes de programação. Muitas das atuais linguagens de programação baseadas em blocos são acessíveis *on-line* via *browser* para facilitar seu acesso. No entanto, devido a problemas de acesso à internet, pode ser necessário instalar também uma versão independente de tal ambiente nos computadores da escola (Papadakis & Orfanakis, 2018). Outras questões a serem consideradas como parte da infraestrutura de TI estão relacionadas ao gerenciamento de contas dos alunos para acessar o ambiente de programação, bem como o armazenamento e *backup* dos artefatos computacionais criados (Martin & Soares, 2016; Papadakis & Orfanakis, 2018). É importante considerar também a capacidade de vários ambientes de programação de realizar testes (como fornecido pelo *App Inventor*), sendo necessária também a configuração adequada da internet sem fio na escola (Schiller et. al., 2014; MITa, 2019). Assim, de acordo com o contexto específico das unidades de ensino e ambientes de programação a serem aplicados, os professores necessitam saber como instalar e manter a infraestrutura tecnológica necessária.

2.2 Características da Formação Continuada

A formação continuada de professores pode visar diferentes níveis de objetivos de aprendizagem de acordo com modelos contínuos de formação de competências do professor, como, por exemplo, representado pelo TIC (Tecnologia da Informação e Comunicação) - Padrão de Competência para Professores (Garofalakis, 2015) (Figura 1).

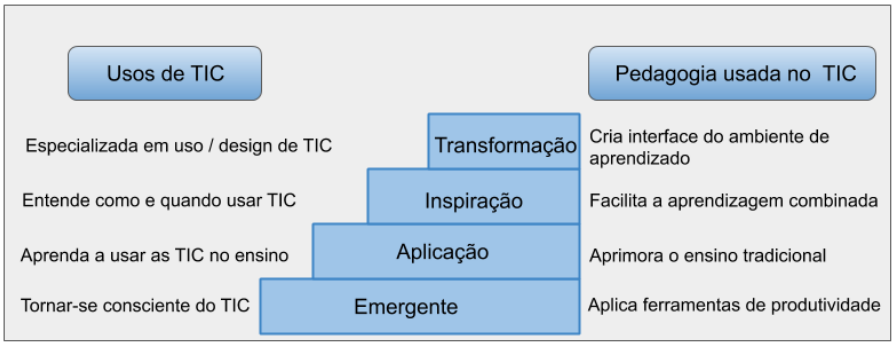

Figura 1: Etapas da formação de professores em TIC - Padrão de Competência para Professores (Garofalakis, 2015).

As primeiras ações da formação continuada podem se concentrar na conscientização e compreensão básica da computação, trabalhando também com a percepção dos professores, aumentando sua confiança e motivação para ensinar computação nas suas disciplinas. Os próximos passos tipicamente incluem uma formação mais abrangente sobre ensino de computação, permitindo que os professores apliquem unidades de ensino existentes, bem como conhecimento sobre como e quando utilizá-las para ensinar computação de forma eficaz e envolvente. Com o objetivo de transformar as unidades de ensino existentes (por exemplo, por meio da incorporação de forma interdisciplinar), as ações de formação continuada podem visar um nível mais elevado de competência, abordando aspectos sobre como usar e projetar unidades de ensino para o ensino de computação.

A formação continuada de professores atuantes em sala de aula para o ensino da computação abrange desde programas de formação nacionais/estaduais/municipais até a aprendizagem por meio de estudo individual (Brasil, 2016). Existem diversos modelos de formação continuada para professores, que variam de altamente adaptados a altamente especializados (Koellner & Jacobs, 2014; Brasil, 2016). Além disso, o acesso onipresente as ferramentas de computação levaram a vários modos e formatos de unidades instrucionais (UI), que consistem em um conjunto de aulas planejadas para ensinar determinados objetivos de aprendizagem a um público-alvo específico (Hill et al., 2005). Isso inclui cursos presenciais, por videoconferência, aprendizagem *on-line* (incluindo MOOC) e/ou instrução combinada (envolvendo uma mistura das aprendizagens presenciais e *on-line*) (Ravitz et al., 2017). Abordagens não exclusivamente presenciais facilitam o alcance de um número maior (geograficamente disperso) de professores, além de permitir que eles organizem a formação em torno de seu próprio horário na escola (Sardessai & Kamat, 2011).

Outra forma consiste no desenvolvimento de práticas em grupo (Wenger et al., 2002), nas quais os professores trabalham juntos para um objetivo comum, por exemplo, implementando o ensino em computação, compartilhando suas experiências e ajudando uns aos outros (Ravitz et al., 2017). Outra maneira pode ser por orientação, onde o aprendizado pode ocorrer dentro do ambiente escolar, liderado por professores com competências em computação e capazes de apoiar seus colegas no aprendizado de computação e na aplicação em suas aulas. Essas formas também podem ser combinadas, por exemplo, iniciando a formação continuada por um curso de formação e, em seguida, ajudando os professores nas primeiras aplicações por meio de orientação e/ou práticas em grupo.

Além disso, os programas de formação são desafiados a projetar e oferecer cursos com o intuito de os professores não apenas ingressarem em suas salas de aula com um equilíbrio de teoria e prática, o qual lhes permita transmitir o conteúdo necessário aos alunos de maneira completa, mas também que continuem a desenvolver seus conhecimentos e habilidades durante a sua carreira. A formação continuada envolve a reflexão sobre a aplicação do ensino

de computação em suas aulas, e também promove a participação ativa em comunidades de prática, organizações profissionais e/ou grupos de interesse (Lloyd & Cochrane, 2006).

3 Metodologia de Pesquisa

No intuito de identificar e caracterizar as UI existentes foi realizado um estudo do estado da arte e prática mundial na formação de professores da Educação Básica voltada ao ensino de computação. O estudo é realizado utilizando o processo de mapeamento sistemático proposto por Petersen et al. (2015).

3.1 Definição do Protocolo da Revisão

O objetivo deste mapeamento é investigar a questão de pesquisa: Quais UI existem para a formação continuada em ensino de computação de professores da Educação Básica de outras áreas? Em razão da abrangência dessa questão de pesquisa, foram formuladas as seguintes perguntas de análise:

PA1. Quais UI existem?
PA2. Qual(is) competências de computação e de pedagogia são ensinadas na UI?
PA3. Quais são as características instrucionais da UI?
PA4. Para quais contextos as UI são projetadas/aplicadas?
PA5. Como a UI foi desenvolvida?
PA6. Como a qualidade da UI foi avaliada?

Foi utilizada a base *Scopus*[1] para efetuar a busca, por integrar as bases de mais do que 11 mil editoras científicas, incluindo ACM, Elsevier, IEEE e Springer. Também foram conduzidas pesquisas no *Google*[2] para minimizar o risco de omitir artigos descrevendo UI não encontradas via outras fontes.

Com base nas perguntas de análise, foram definidos os critérios de inclusão e exclusão utilizados para a seleção de artigos relevantes. Foram incluídos artigos que contém a descrição de uma UI para professores "*in-service*" na Educação Básica (anos finais do Ensino Fundamental) enfocando o ensino de computação, especificamente algoritmos e programação utilizando uma linguagem de programação baseada em blocos. Por outro lado, excluímos materiais que não possuem foco especificamente no ensino de computação. Também foram excluídas UI voltadas a outros níveis educacionais, como p. ex. ensino infantil, superior e/ou apenas para Ensino Médio. Não consideramos também materiais disponíveis apenas em formato de resumos ou apresentações. A fim de determinar os termos para executar a busca, os conceitos relacionados à questão de pesquisa foram determinados, conforme apresentado na Tabela 2.

Tabela 2: Palavras-chave.

Termos	**Sinônimos**	**Tradução (inglês)**
unidade instrucional	curso	*course, MOOC, training*
professores		*teacher*
educação básica	ensino fundamental, ensino médio	*school, k-12*
ensino	aprendizagem	*learning, teaching*
computação	programação, ciência da computação	*computer science, programming*

[1] *www.scopus.com*
[2] *www.google.com*

A partir de buscas de teste foi ajustada a *string* de pesquisa utilizando os termos identificados (Tabela 3).

Tabela 3: *String* de busca.

(course OR MOOC OR training) AND (teacher) AND (school OR "k-12") AND (learning OR teaching) AND ("computer science" OR programming)

Critério de qualidade. Consideramos somente artigos que apresentam uma descrição substancial da UI referente às perguntas de análise definidas. Como essa pesquisa não visa a realização de uma meta-análise não foi utilizado o *research design* apresentado nos artigos como critério de qualidade.

3.2 Execução da Busca

A busca foi realizada em outubro de 2018 conforme definido pela primeira autora e revisada pelos coautores com expertise na área do ensino de computação. A busca no *Scopus* retornou 764 artigos, dos quais foram considerados, pela ordem de relevância, os 200 primeiros para a análise. Também foram realizadas pesquisas no *Google* e analisados os 200 resultados mais relevantes. A partir dos resultados da busca, foi realizada a seleção de artigos relevantes em duas etapas utilizando os critérios de inclusão e exclusão. Na primeira etapa, os artigos potencialmente importantes foram selecionados com base na leitura e análise do título, resumo e palavras-chave. Na segunda etapa foi realizada a seleção com base na análise do texto completo dos artigos considerados relevantes na etapa anterior. A seleção foi revisada e discutida entre os autores até um consenso ser atingido levando em consideração os critérios de inclusão/exclusão e o critério de qualidade.

Tabela 4: Quantidade de artigos por etapa de seleção por repositório.

Fonte de dados	**Resultados da pesquisa**	**Artigos analisados**	**Seleção depois da 1° etapa**	**Seleção depois da 2° etapa**
Scopus	764	200	54	12
Google	246.000.000	200	5	4

Durante a primeira e segunda etapas da busca, vários artigos foram desconsiderados por relatarem apenas informações sobre a organização de um curso para professores, sem considerar o conteúdo a ser ensinado (Cao & He, 2009) e outros, por retratar a formação apenas para professores do Ensino Médio (Hamlen et al., 2018; Morelli et al., 2015). Alguns artigos foram desconsiderados por não apresentarem detalhes da UI (Hodhod et al., 2016) e outros por não se enquadrarem nos critérios de inclusão/exclusão. Também foi realizada uma análise dos artigos secundários indicados nas referências dos artigos primários encontrados na seleção dos artigos, resultando na identificação de mais um trabalho relevante (Cho et al., 2014).

4 Resultados

Os artigos foram lidos de forma completa e os dados extraídos pelos autores. A extração de dados foi dificultada, em alguns casos, pela falta de um protocolo estruturado para as publicações nessa área, resultando em artigos desprovidos de alguns detalhes sobre a UI. Isso impôs aos pesquisadores inferir algumas informações apresentadas no artigo de maneira implícita, incluindo, por exemplo, o método de análise de dados, idioma da UI e a necessidade de competências prévias de computação. Assim, com base nas perguntas de

análise foram extraídas as informações dos artigos. Em seguida é apresentada a análise dos dados referentes a cada uma das perguntas de análise.

PA1. Quais UI existem?

Ao total foram identificadas 16 unidades instrucionais para a formação continuada, voltadas ao ensino de computação para professores *in-service* da Educação Básica (Tabela 5), especificamente do 6º ao 9º ano do Ensino Fundamental.

Tabela 5: Unidades instrucionais.

Citação	Referência bibliográfica
(Partanen et al., 2016)	Partanen, T.; Mannila, L.; Poranen T. (2016). Learning programming online: a racket-course for elementary school teachers in Finland. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli, Finland, 178-179.
(Martinez et al., 2016)	Martinez, M. C.; Gomez, M. J.; Moresi, M.; Benotti L. (2016). Lessons Learned on Computer Science Teachers Professional Development. In Proc. of the ACM Conference on Innovation and Technology in Computer Science Education. Arequipa, Peru, 77-82.
(Kay & Moss, 2012) (Kay et al., 2014)	Kay, J. S.; Moss, J. G. (2012). Using robots to teach programming to K-12 teachers. In: Proc. of the Frontiers in Education Conference Proceedings, Seattle, WA, EUA. Kay, J. S.; Moss, J. G.; Engelman, S.; MacKlin T. (2014). Sneaking in through the back door: introducing k-12 teachers to robot programming. In: Proc. of the 45th ACM Technical Symposium on Computer Science Education, Atlanta, GA, EUA, 499-504.
(Cooper et al., 2015) (Cooper et al., 2011)	Cooper, S.; Dann, W.; Lewis, D.; Lawhead, P.; Rodger, S.; Schep, M.; Stalvey, R. H. (2011). A pre-college professional development program. In: Proc. of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education. Darmstadt, Alemanha, 188-192. Cooper, S.; Rodger, S. H.; Schep, M.; Stalvey, R. H.; Dann, W. (2015). Growing a K-12 Community of Practice. In: Proc. of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, MO, EUA, 290-295.
(Liu et al., 2015)	Liu, J.; Wilson, J.; Hemmenway, D.; Xu Y.; Lin, C. (2015). Oh SNAP! A one-week summer computing workshop for K-12 teachers. In: Proc. of the 10th International Conference on Computer Science & Education, Cambridge, Reino Unido, 663-668.
(Liu et al., 2013)	Liu, J.; Lin, C. H.; Potter P.; Hasson, E. P.; Barnett, Z.; Xu Y.; Singleton M. (2013). Going mobile with app inventor for android: a one-week computing workshop for K-12 teachers. In: Proc. of the 44th ACM Technical Symposium on Computer Science Education, Denver, CO, EUA, 433-438.
(Vivian et al., 2014)	Vivian, R.; Falkner, K.; Falkner, N. (2014). Addressing the challenges of a new digital technologies curriculum: MOOCs as a scalable solution for teacher professional development. Research in Learning Technology, 22.
(Hamner et al., 2016)	Hamner, E.; Cross, J.; Zito, L.; Bernstein D.; Mutch-Jones, K. (2016). Training teachers to integrate engineering into non-technical middle school curriculum. In: Proc. of the IEEE Frontiers in Education Conference, Erie, PA, EUA.
(Haden et al., 2016)	Haden, P.; Gasson, J.; Wood, K.; Parsons D. (2016). Can you learn to teach programming in two days? In: Proc. of the Australasian Computer Science Week Multiconference. Canberra, Australia.
(Liu et al., 2014)	Liu, J.; Lin, C. H.; Wilson, J.; Hemmenway, D.; Hasson, E. P.; Barnett, Z.; Xu Y. (2014). Making games a "snap" with Stencyl: a summer computing workshop for K-12 teachers. In: Proc. of the 45th ACM Technical Symposium on Computer Science Education, Atlanta, GA, EUA, 169-174.
(Rusu, 2015)	Rusu, A. (2015). Introducing gaming tools for computing education in STEM related curricula. In: Proc. of the IEEE Frontiers in Education Conference, El Paso, TX, EUA.
(Geldreich et al., 2018)	Geldreich, K.; Talbot, M.; Hubwieser, P. (2018). Off to new shores: preparing primary school teachers for teaching algorithmics and programming. In: Proc. of the 13th Workshop in Primary and Secondary Computing Education, Potsdam, Alemanha.

(von Wangenheim, 2015)	von Wangenheim, A.; Gresse von Wangenheim, C.; Santana Pacheco, F.; Hauck, J.; Ferreira, M. N. (2017). Motivating Teachers to Teach Computing in Middle School – A Case Study of a Physical Computing Taster Workshop for K-12 Teachers. International Journal of Computer Science Education in Schools, 35(1), 368-373.
(Alimisis et al., 2009)	Alimisis, D.; Frangou, S.; Papanicolaou, K. (2009). A Constructivist Methodology for Teacher Training in Educational Robotics In: Proceedings of the TERECoP Course in Greece through Trainees Eyes. In: Proc. of the 9th IEEE Int. Conference on Advanced Learning Technologies, Riga, Latvia. 25-28.
(Cho et al.,2014)	Cho, S.; Pauca, P.; Johnson, D.; James, Y. (2014). Computational Thinking for the Rest of Us: A Liberal Arts Approach to Engaging Middle and High School Teachers with Computer Science Students. In: Proc. of Society for Information Technology & Teacher Education International Conference. Jacksonville, FL, EUA, 79-86.
(Liu et al., 2011)	Liu, J.; Lin, C. H.; Hasson, E. P.; Barnett, Z.; Xu Y. (2011). Introducing computer science to K-12 through a summer computing workshop for teachers. In: Proc. of the 42nd ACM Technical Symposium on Computer Science Education. Dallas, TX, EUA, 389-394.

Um número significativo de UI encontradas foi publicado nos últimos 10 anos, indicando um aumento de importância dessa área. Observando o crescimento de iniciativas para ensinar computação nas escolas, verifica-se a necessidade de professores qualificados para essa função. Um dos principais motivos para o aumento das UI existentes entre 2014 e 2016 (apresentado na Figura 2) possivelmente tem relação com a inclusão de programação e computação como um elemento integrado nas disciplinas, como está acontecendo na Finlândia (Partanen et al., 2016), Austrália e Inglaterra (Vivian et al., 2014).

Figura 2: Quantidade de UI com foco no ensino de computação para professores por ano de publicação.

PA2. Qual(is) competências de computação e pedagogia são ensinadas na UI?

Dentro do foco da pesquisa, todas as UI encontradas visam ensinar práticas de programação para que os participantes possam aplicar computação em suas salas de aula. Algumas UI buscam promover a habilidade e confiança (Kay & Moss, 2012; Hamner et al., 2016), enquanto outra tem o intuito de explicar técnicas pedagógicas referentes ao ensino de programação (Haden et al., 2016). Vivian et al. (2014) também menciona a importância de os participantes entenderem os impactos culturais da tecnologia.

Com o intuito de alcançar os objetivos propostos dos trabalhos encontrados, as competências a serem adquiridas pelos participantes foram classificadas em conceitos (e práticas) de computação, competências pedagógicas e tecnológicas. Os conceitos de computação ensinados por várias UI focam em algoritmos e programação (condicionais, loops e variáveis), e outras incluem desenvolvimento e uso de abstrações. Há exceções que

ensinam também colaboração na computação (Martinez et al., 2016), sistemas de computador (Vivian et al., 2014), criação (Liu et al., 2014) e testes (Liu et al., 2011) de artefatos computacionais, definição de problemas computacionais (Haden et al, 2016) e análise de dados (Liu et al., 2013).

As competências pedagógicas instruídas geralmente são vinculadas com o conhecimento pedagógico. O conhecimento pedagógico é ensinado, em 12 UI, por meio do desenvolvimento de materiais curriculares (Liu et al., 2011), de metáforas e analogias adequadas para os conceitos (Rusu, 2015), explicação de estratégias instrucionais (von Wangenheim et al., 2017) e demonstração de práticas educativas específicas (Haden et al, 2016). Os requisitos curriculares também foram considerados por Partanen et al. (2016).

As competências tecnológicas ensinadas são relacionadas com o conhecimento tecnológico. Porém, observa-se que 7 UI não relatam a inclusão desse tipo de competência. Nas demais UI, o conhecimento tecnológico é coberto por meio de instruções de instalação de *software* (Liu et al., 2011; von Wangenheim et al., 2017), montagem de robôs LEGO (Alimisis et al., 2009) e simulação da execução dos programas em um emulador de *Android* (Liu et al., 2013).

As características encontradas nas UI são: apoio após o término da UI (Cooper et al., 2015), aprendizado com emulador *Android* (Liu et al., 2013), utilização de exemplos com jogos (Liu et al., 2014; Rusu, 2015) e ensino de instruções de instalação do *software* (Liu et al., 2011; von Wangenheim et al., 2017). Algumas UI envolveram os participantes em atividades práticas na maior parte do tempo (Cho et al., 2014). Enquanto isso, algumas focaram em cursos de curta duração, com aulas nos finais de semana e/ou feriados (Haden et al, 2016). Duas UI são projetadas para o modo de ensino *online* (Partanen et al., 2016) (Vivian et al., 2014) e outras utilizam múltiplas linguagens e/ou plataformas (Martinez et al., 2016; von Wangenheim et al., 2017).

Tabela 6: Visão geral das competências de computação, pedagogia e tecnologia das UI.

Referência	**Objetivo geral de aprendizagem**	**Conceitos e práticas de computação**	**Competências pedagógicas**	**Competências tecnológicas**
Partanen et al., 2016	Ensinar programação para professores, promovendo criatividade e compartilhamento de artefatos programados e ideias pedagógicas. Dar aos participantes conhecimento suficiente sobre o assunto para que eles sejam capazes não apenas de avaliar e usar materiais de programação prontos, mas também para criar seus próprios exercícios de programação.	Desenvolvimento e uso de abstrações, algoritmos e programação (lógica e repetição)	Conhecimento pedagógico e requisitos curriculares da Finlândia	Conhecimento tecnológico
Martinez et al., 2016	Ensinar aos professores práticas de programação e de ensino para que professores possam aplicar computação em suas salas de aula.	Algoritmos e programação (condicionais, loops, variáveis) e colaboração na computação (por meio do uso de múltiplas linguagens e plataformas)	Conhecimento, pedagógico, por meio da prática de sala de aula para professores	Conhecimento tecnológico
Kay & Moss, 2012; Kay et al., 2014	Fornecer aos professores conhecimentos suficientes de conteúdos de programação de robôs, para poderem resolver problemas e elaborar atividades. Dar aos professores a confiança e as habilidades para iniciar programas extracurriculares de robótica com seus alunos.	Algoritmos e programação (saídas simples, sensores, loops, manipulação de eventos, condicionais, funções e variáveis)	Conhecimento, pedagógico, por meio do uso de formas de integrar a ciência da computação na sala de aula	NI
Cooper et al., 2015; Cooper et al., 2011	Ensinar programação para professores para que possam desenvolver planos de aula específicos de disciplina, integrando computação nas escolas de Ensino Fundamental e Médio. Desenvolver a capacidade de adaptar os tutoriais presentes	Algoritmos e programação	NI	Conhecimento tecnológico

	no site para aplicar em suas aulas.			
Liu et al., 2015	Introduzir conceitos fundamentais de computação para professores do Ensino Fundamental e Médio usando o SNAP! para que possam desenvolver currículos para os assuntos que irão ensinar nos semestres seguintes.	Algoritmos e programação (entrada de dados, condicionais, booleanos, variáveis, laços de repetição, métodos e listas)	NI	NI
Liu et al., 2013	Introduzir conceitos de computação usando o App Inventor para professores do Ensino Fundamental e Médio de diversas áreas. Os participantes devem desenvolver projetos para os assuntos que irão ensinar nos semestres seguintes.	Algoritmos, programação (entrada e saída de um programa, estruturas condicionais, "boolean", variáveis, "loops", métodos e funções), dados e análise	Conhecimento, pedagógico, por meio de material curricular, para suas áreas de conhecimento	Conhecimento tecnológico, por meio da simulação da execução dos programas em um emulador Android
Vivian et al., 2014	Os objetivos estão relacionados aos impactos culturais da tecnologia, o pensamento computacional, o uso de sistemas e dados digitais, bem como conceitos e práticas de programação visual.	Sistemas de computador, algoritmos e programação (programação visual, padrões e representação de dados)	Conhecimento, pedagógico, por meio da criação de um recurso de ensino e um plano de aula	Conhecimento tecnológico
Hamner et al., 2016	Tem por objetivo promover a habilidade, confiança e autoeficácia do professor no projeto e implementação de projetos de robótica em sala de aula.	Algoritmos e programação	Conhecimento, pedagógico, por meio da avaliação do perfil dos estudantes e elaboração de um plano de ensino	Não é uma meta de aprendizado
Haden et al., 2016	Ensinar a habilidade de programação para professores e explicar técnicas pedagógicas em programação.	Algoritmos, programação, reconhecimento e definição de problemas computacionais	Conhecimento pedagógico, por meio da demonstração e exploração de práticas educativas específicas que os professores podem usar imediatamente em suas próprias salas de aula; discussão de questões pedagógicas, como o valor da analogia e metáfora ao explicar conceitos abstratos de computação como fluxo de controle.	NI
Liu et al., 2014	Aprender conceitos fundamentais de computação, programação usando Stencyl e desenvolver projetos para os assuntos que irão ensinar nos semestres seguintes.	Algoritmos, programação (entrada e saída de um programa, estruturas condicionais, "boolean", variáveis, "loops", métodos, classes e listas) e criação de artefatos computacionais	Conhecimento pedagógico, por meio do desenvolvimento de projetos para a sua disciplina	NI
Rusu, 2015	Ensinar a usar a computação como um meio para fazer conexões entre diferentes áreas curriculares e ensinar habilidades de resolução de problemas de ordem superior.	Algoritmos, programação (incluindo reconhecimento de padrões), desenvolvimento e uso de abstrações	Conhecimento pedagógico, por meio da identificação do conceito para ensinar os alunos, desenvolvimento de metáforas adequadas para o conceito, identificação de ferramentas adequadas e utilização de jogos para o ensino.	NI
Geldreich et al., 2018	Oficina de desenvolvimento profissional *in-service* em algoritmos e programação. Acompanhamento e supervisão por um ano letivo na implementação dos novos tópicos.	Algoritmos e programação	Conhecimento pedagógico, com foco em como os professores poderiam implementar esses conceitos no currículo da escola.	NI
von Wangenheim, 2015	Formação de professores *in-service* de Ensino Médio por meio de uma oficina para motivar e despertar o entusiasmo dos professores por esta área. O objetivo da oficina foi ensinar conceitos básicos de computação a professores de qualquer área do conhecimento (e/ou professores que fornecem apoio de TI nas escolas), visando uma integração multidisciplinar da computação no currículo existente.	Algoritmos e programação (loops, condicionais, manipulação de eventos etc.)	Conhecimento pedagógico, incluindo possibilidades de integrar o ensino de computação de forma multidisciplinar em diferentes disciplinas e estratégias instrucionais.	Conhecimento tecnológico, incluindo instalação de Linux
Alimisis et al., 2009	O objetivo do curso foi capacitar os formandos a compreender as perspectivas pedagógicas da robótica educacional e desenvolver atividades robóticas dentro da abordagem construtivista do ensino.	NI	Pedagogia construtivista (Resnick & Ocko, 1991)	Conhecimento tecnológico, por meio da montagem de robôs LEGO.
Cho et al.,2014	Os objetivos da oficina foram: 1) melhorar	Algoritmos, programação	NI	NI

	a percepção social da pesquisa e oportunidades de trabalho em ciência da computação; 2) preparar os professores para se tornarem modelos eficazes para o pensamento computacional; e 3) ajudar a integrar o pensamento computacional em seus currículos.	(representação de dados, pesquisa binária e o problema da mochila), desenvolvimento e uso de abstrações, testar e refinar artefatos computacionais		
Liu et al., 2011	O objetivo é introduzir conceitos de computação para professores de tecnologia, matemática e ciências em todos os níveis do ensino para expor os alunos à computação em idade precoce.	Algoritmos e programação (estrutura do programa e fluxo de execução, objetos, métodos / funções, condicionais, expressão booleana, variáveis, loops, entrada do usuário, lógica booleana e processamento paralelo)	Conhecimento pedagógico, por meio do desenvolvimento de materiais curriculares para os assuntos que irão ensinar nos semestres seguintes, com a ajuda dos tutores.	Conhecimento tecnológico, por meio de instruções de instalação do software.

Com todos esses conceitos e competências ensinados aos participantes das UI, a maioria das UI visa atingir o nível de aprendizagem de aplicação, conforme o modelo TIC - Padrão de Competência para Professores (Figura 1). Somente uma UI foca no nível emergente.

PA3. Quais são as características instrucionais da UI?

A maioria das UI identificadas utiliza o modo de ensino presencial (14 UI) com somente duas realizadas de forma *online*. A duração das UI encontradas varia de 1 a 30 horas/aula (Figura 3). Apenas três delas têm uma duração maior do que 30 horas/aula.

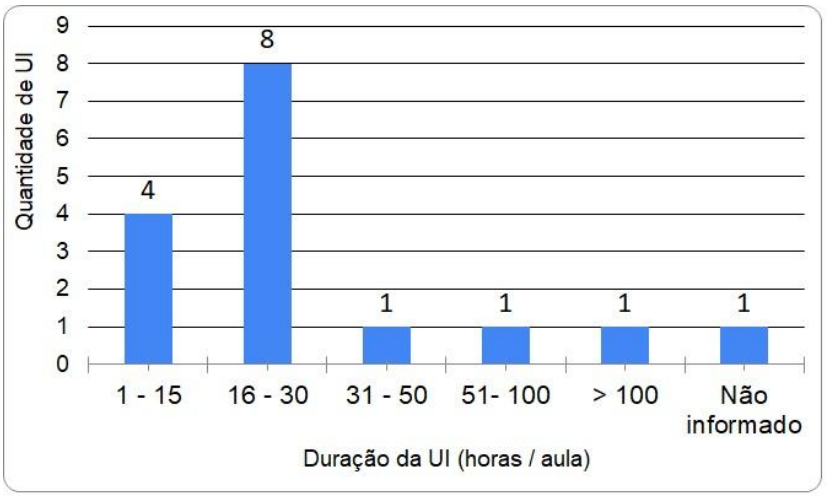

Figura 3: Tempo de duração da UI em horas/aula.

O ensino de conceitos de computação é tipicamente voltado a algoritmos e programação de *software*. Deste modo, as UI aderem a ambientes de programação para aplicar os conceitos e proporcionar experiências práticas. Um modo intuitivo é utilizar ambientes de programação visual baseados em blocos, como Scratch e Alice (desenvolvimento de *software desktop*), os mais utilizados nas UI encontradas (Figura 4). ScratchJr, Code.org, SNAP!, App Inventor e Stencyl também são ambientes de programação baseados em blocos encontrados no mapeamento. LEGO NXT-G e CREATE Lab Visual Programmer também são baseados em blocos e utilizados na programação de robótica. De forma mais pontual são também utilizadas linguagens de programação baseadas em textos (como Racket e Python) e outras ferramentas (como Chatbot e UNC++Duino).

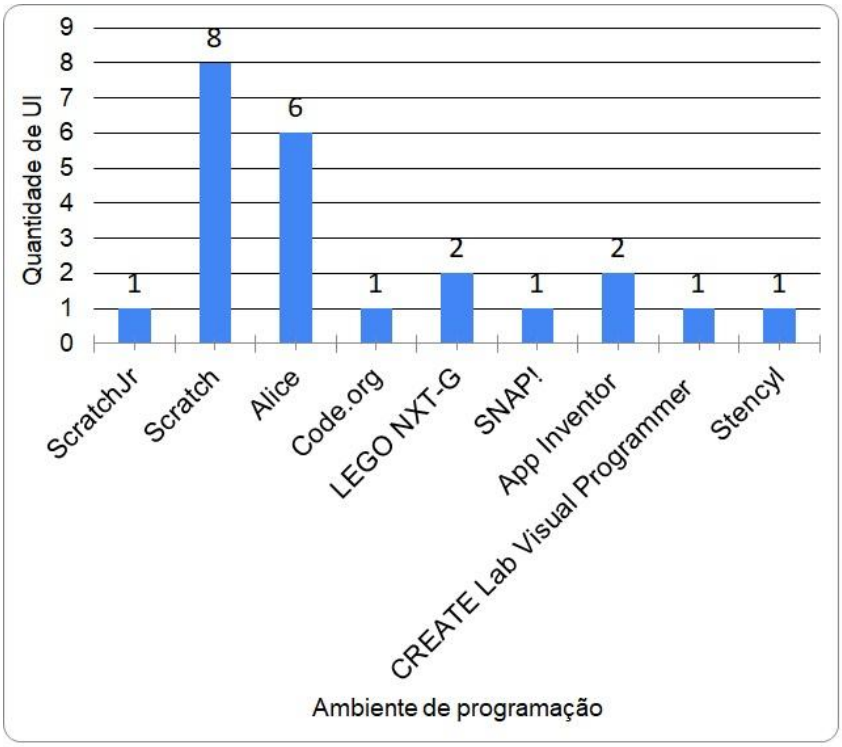

Figura 4: Ambientes de programação.

As UI encontradas geralmente apresentam uma organização bem definida dos tópicos a serem ensinados. Algumas delas abordam metodologias que utilizam experiência prática por meio de tutoriais, com tutores para tirar dúvidas (Partanen et al., 2016; Cooper et al., 2015; von Wangenheim et al., 2017). Outras intercalam aulas expositivas e práticas, geralmente em pares, com o intuito de introduzir conceitos de computação e depois utilizar o conhecimento aprendido para resolver exercícios e promover discussões. Esses exercícios podem variar de atividades práticas utilizando um ambiente de programação até o desenvolvimento de materiais e currículos para serem aplicados na sala de aula. Em algumas UI, os participantes também apresentam os artefatos criados (Hamner et al., 2016; Vivian et al., 2014; Cooper et al., 2015; Partanen et al., 2016).

Nas UI encontradas, são utilizados diversos tipos de materiais instrucionais (Figura 5). Os materiais mais utilizados são exemplos que contemplam programas completos (Haden et al, 2016), planos de aula (Cooper et al., 2015) e jogos (Liu et al., 2014). Exercícios e slides também são muitas vezes empregados em aplicações que intercalam aulas expositivas e práticas (von Wangenheim et al., 2017). Vídeos são apresentados para motivar os participantes e introduzir conceitos de computação (Partanen et al., 2016), principalmente quando o curso é *online* (Vivian et al., 2014). Tutoriais guiando passo-a-passo a programação usando determinados ambientes são inseridos nas aulas práticas (Cooper et al., 2015; von Wangenheim et al., 2017). A variedade dos materiais instrucionais utilizados é apresentada na Figura 5.

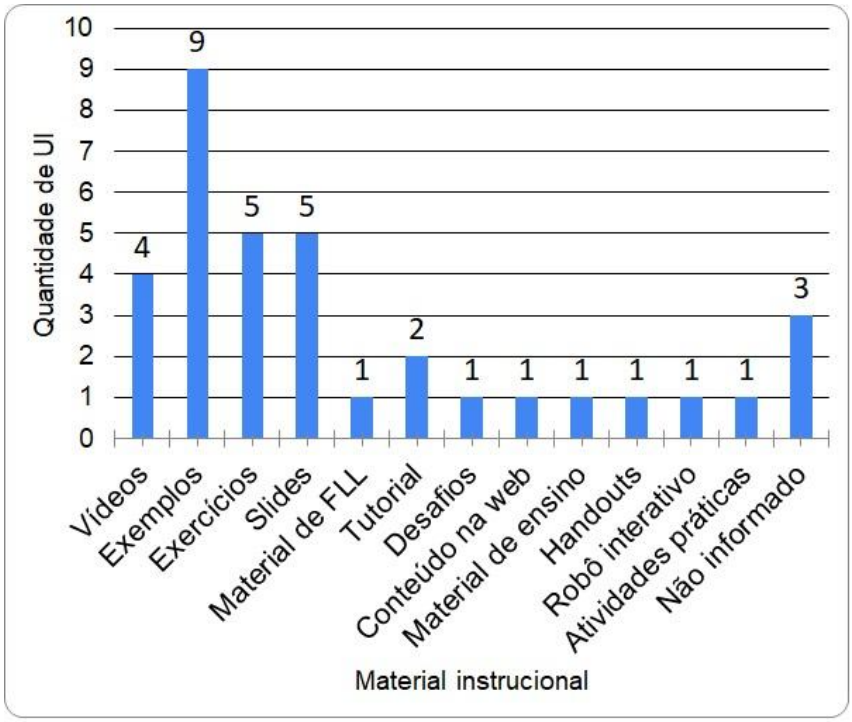

Figura 5: Material instrucional utilizado.

A licença para o uso desses materiais foi informada em apenas três publicações: *Koodiaapinen MOOC* (Partanen et al., 2016), *Creative Commons* (von Wangenheim et al., 2017) e uma indica que o uso é grátis (Cooper et al., 2015). A língua usada nos materiais e nos cursos como um todo, foi inferida de acordo com as informações dos países e pesquisadores que aplicaram a UI com uma predominância do inglês (Figura 6). Somente a UI voltada ao ensino de computação física (von Wangenheim et al., 2017) está disponível em português. A predominância de materiais na língua pode ter relação com os termos da *string* de busca serem definidos em inglês, assim espera-se uma tendência de encontrar materiais nessa língua.

A aprendizagem e o conhecimento que os participantes estão adquirindo durante a aplicação são importantes para o sucesso da UI. Referente a avaliação da aprendizagem, vários artigos indicam a estratégia utilizada, embora cinco publicações não tenham informado o método e uma não inclui a avaliação. As outras onze UI recorreram à autoavaliação por meio de questionários (antes e depois da aplicação) e exercícios de programação, revisão por pares de artefatos de programação e desempenho em lições sobre aspectos pedagógicos (Partanen et al., 2016). Apresentações de artefatos criados e *feedback* do tutor mediante observações também foram utilizados.

Oferecer suporte após o término da UI pode ser uma maneira de passar confiança aos participantes e fazer com que utilizem, em suas escolas, o conteúdo que aprenderam. No entanto, onze publicações não informam se disponibilizam esse tipo de suporte. Uma proporciona na forma *online*, mas apenas durante o curso (Alimisis et al., 2009). Outras quatro oferecem visitas durante o ano seguinte (Cooper et al., 2015), suporte nos semestres seguintes (Liu et al., 2011), possibilidade de procurar apoio (Geldreich et al., 2018) ou trocas de experiências por meio da Comunidade Google+ (Vivian et al., 2014).

As informações referentes às características das UI são resumidas na Tabela 7.

Tabela 7: Visão geral das características das UI.

Referência	Modo	Duração	Ambiente de programação	Método instrucional	Material instrucional	Método de avaliação	Suporte para aplicação	Idioma	Licença
Partanen et al., 2016	MOOC	42 h/a	ScratchJr, Scratch, Racket e Python, com DrRacket e WeScheme	Aulas expositivas, atividades práticas de programação e redação sobre aspectos pedagógicos.	Vídeos motivacionais, vídeos tutoriais com conceitos e exemplos de programação, exercícios e slides	Autoavaliação por meio de exercícios de programação, revisão por pares de artefatos de programação e desempenho em lições sobre aspectos pedagógicos	NI	finlandês	Koodiaapinen MOOC
Martinez et al., 2016	Curso presencial	60 h/a	Alice, Code.org, Chatbot e UNC++Duino	Abordagem baseada em investigação com desafios de programação.	Lições (exercícios) baseadas em questionários e exemplos de planos de aulas	Autoavaliação por meio de pré e pós questionários e observações em sala de aula	NI	espanhol	NI
Kay & Moss, 2012; Kay et al., 2014	Curso presencial	25 h/a	LEGO NXT-G, Scratch e Alice	Aulas expositivas, lições interativas, exercícios, discussões, apresentação dos resultados pelos participantes.	Material sobre o programa de Robótica da FIRST LEGO League (FLL) para crianças de 9 a 14 anos de idade	Questionários de avaliação do conhecimento dos conteúdos antes e depois da oficina	NI	inglês	NI
Cooper et al., 2015 Cooper et al., 2011	Curso presencial	120 h/a	Alice	Atividades práticas orientadas a problemas a resolver e desenvolvimento de um plano de ensino. Apoio para os professores durante o ano escolar.	Tutoriais sobre Alice, exemplos de planos de aula, vídeos e desafios disponíveis via o site do projeto	Apresentação do plano de ensino e *feedback* dados pelos outros professores em relação ao plano	Visitas durante o ano seguinte para ajudar na aplicação	inglês	grátis
Liu et al., 2015	Curso presencial	25 h/a	SNAP!	Aulas expositivas, demonstração de exemplos, atividades práticas de programação e desenvolvimento de currículos para suas aulas e apresentação dos resultados pelos participantes.	Exemplos no SNAP!	Testes antes e depois de cada dia	NI	inglês	NI
Liu et al., 2013	Curso presencial	25 h/a	App Inventor	Aulas expositivas, demonstração de exemplos, atividades práticas de programação e desenvolvimento de currículos para suas aulas e apresentação dos resultados pelos participantes.	Programas exemplo e tarefas (exercícios)	Quiz pré e pós sessão todos os dias	NI	inglês	NI

Vivian et al., 2014	MOOC	NI	Scratch	Aulas expositivas, atividades práticas visando aos participantes se conectarem, compartilharem ideias e atividades e construírem coletivamente recursos on-line correspondentes às áreas de tópicos.	Conteúdo na web, vídeos conceituais e exemplos trabalhados ligados aos objetivos de aprendizagem do currículo australiano	Criação de recursos de ensino, um plano de aula e tarefas para cada seção	Comunidade Google+ para troca de experiências	inglês	NI
Hamner et al., 2016	Curso presencial	4 h/a	Kits de robótica, CREATE Lab Visual Programmer	Aulas expositivas, demonstração de exemplos, atividades práticas de programação.	NI	Artefato criado	NI	inglês	NI
Haden et al., 2016	Curso presencial	20 h/a	Scratch	NI	Slides, exercícios e projeto de programação completo (exemplo)	NI	NI	inglês	NI
Liu et al., 2014	Curso presencial	25 h/a	Stencyl	Aulas expositivas, demonstração de exemplos, atividades práticas de programação e desenvolvimento de currículos para suas aulas e apresentação dos resultados pelos participantes.	Exemplos de jogos	Questionários pré e pós-sessão	NI	inglês	NI
Rusu, 2015	Curso presencial	15 h/a	Scratch, Game Maker, Alice	Aulas expositivas, discussão, atividades práticas, atividades sem utilizar programas de computador, apresentação e *brainstorming.*	NI	NI	NI	inglês	NI
Geldreich et al., 2018	Curso presencial	14 h/a	Scratch	Atividades *unplugged*, exercícios de programação e discussão.	*Handouts*, exercícios e material de ensino	NI	Recebem material on-line e possibilidade de procurar apoio	alemão	NI
von Wangenheim, 2015	Curso presencial	6 h/a	Scratch, Snap!	Aulas expositivas, discussões e atividades práticas por pares.	Slides, vídeos, exemplos, tutorial online e robô interativo.	Não há avaliação dos participantes	NI	português do Brasil	*Creative Commons*
Alimisis et al., 2009	Curso presencial	30 h/a	Sistema LEGO Mindstorms NXT	Atividades práticas, discussão, apresentação dos resultados pelos participantes e aplicação em sala de aula.	NI	*Feedback* em intervalos regulares por meio de observações	Suporte online durante o curso	inglês	NI
Cho et al.,2014	Curso presencial	16 h/a	Scratch, App Inventor	Aulas expositivas, atividades práticas de programação e apresentação dos resultados pelos participantes.	Slides e atividades práticas	NI	NI	inglês	NI
Liu et al., 2011	Oficina presencial	25 h/a	Scratch, Alice	Aulas expositivas e atividades prática.	Slides	NI	Suporte nos semestres seguintes	inglês	NI

PA4. Quais são as características de contexto da UI?

A maioria das UI encontradas tem o foco na Educação Básica como um todo e não um nível específico, envolvendo como participantes professores de diferentes níveis escolares (Figura 6).

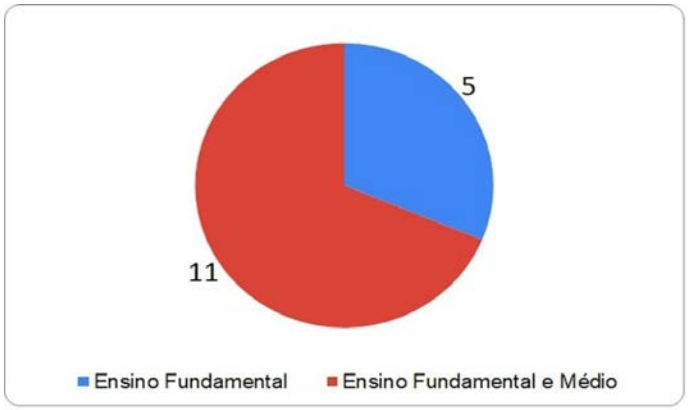

Figura 6: Nível escolar no qual os professores participantes lecionam.

Observando a atual carência de professores formados especificamente na área de computação e a falta da inclusão de conteúdo de computação no currículo base em muitos países (von Wangenheim et al., 2017), uma solução proposta por todas as UI analisadas é uma abordagem interdisciplinar. Desse modo, proporcionam aos professores a possibilidade de fazerem a integração entre o ensino de computação e outras disciplinas consideradas tradicionais, como matemática, história e geografia. Algumas UI também buscam abranger os professores que fornecem apoio de TI nas escolas (Tabela 8).

Tabela 8: Visão geral das características de contexto das UI.

Referência	**Área de conhecimento dos professores *in-service***	**Nível de ensino dos professores *in-service***
Partanen et al., 2016	Matemática	Ensino Fundamental
Martinez et al., 2016	Qualquer disciplina	Ensino Fundamental e Médio
Kay & Moss, 2012; Kay et al., 2014	Qualquer disciplina	Ensino Fundamental e Médio
Cooper et al., 2015; Cooper et al., 2011	Qualquer disciplina	Ensino Fundamental e Médio
Liu et al., 2015	Qualquer disciplina	Ensino Fundamental e Médio
Liu et al., 2013	Qualquer disciplina	Ensino Fundamental e Médio
Vivian et al., 2014	Qualquer disciplina	Ensino Fundamental e Médio
Hamner et al., 2016	Disciplinas tradicionais (como Inglês, História ou Ciência)	Ensino Fundamental
Haden et al., 2016	Qualquer disciplina	Ensino Fundamental e Médio
Liu et al., 2014	Qualquer disciplina	Ensino Fundamental e Médio
Rusu, 2015	Qualquer disciplina	Ensino Fundamental (Anos Finais) e Médio
Geldreich et al., 2018	Qualquer área de conhecimento	Ensino Fundamental
von Wangenheim, 2015	Qualquer área de conhecimento e professores que fornecem apoio de TI nas escolas	Ensino Fundamental
Alimisis et al., 2009	Várias áreas de conhecimento incluindo Matemática, Física, Informática	Ensino Fundamental e em formação

Cho et al.,2014	Várias áreas de conhecimento incluindo Artes, Estudos Sociais e Língua estrangeira	Ensino Fundamental e Médio
Liu et al., 2011	Várias áreas de conhecimento incluindo Matemática, Informática, Leitura, Ciências, Biologia e Educação especial	Ensino Fundamental e Médio

As competências prévias dos participantes não são informadas em várias publicações, porém, de forma geral, professores *in-service* de outras áreas não têm um conhecimento prévio da computação. Uma das unidades (von Wangenheim et al., 2017) somente cita a necessidade de saber usar um computador e outra enfatiza a disposição dos professores em ensinar computação (Martinez et al., 2016).

PA5. Como a UI foi desenvolvida?

No desenvolvimento das UI observou-se a escassez de informações sobre o processo utilizado, sendo cinco publicações que não abordam essa questão (Martinez et al., 2016; Kay & Moss, 2012; Cooper et al., 2015; Liu et al., 2015; Rusu, 2015). Encontramos informações da metodologia de pesquisa utilizada somente referente a três UI (Vivian et al., 2014; von Wangenheim et al., 2017; Geldreich et al., 2018). A UI de Vivian et al. (2014) foi desenvolvida em três etapas (revisão de computação, desenvolvimento e *design*, implementação e revisão), iniciando com uma revisão quase-sistemática de trabalhos de pesquisa sobre computação na educação. Outras duas se baseiam no *design* instrucional (von Wangenheim et al., 2017; Geldreich et al., 2018), sendo que um trabalho seguiu o modelo ADDIE (von Wangenheim et al., 2017). As UI restantes fundamentam-se em modelos de formação já existentes na literatura (Hamner et al., 2016), trabalhos realizados por universidades (Liu et al., 2014), experiências anteriores (Partanen et al., 2016), exemplos encontrados em livros (Haden et al, 2016) ou uma proposta introduzida por Resnick (Alimisis et al., 2009).

PA6. Como a qualidade da UI é avaliada?

A avaliação é uma etapa importante no processo de desenvolvimento de uma UI e busca avaliar se os objetivos de aprendizagem são atingidos e identificar oportunidades de melhoria. Nos trabalhos encontrados, quatorze UI foram avaliadas por meio de estudo de caso (Figura 7). Na maioria dos estudos, esses dados são obtidos por meio de questionários antes e/ou após o ensino. Duas UI foram avaliadas de forma *ad-hoc*, menos rigorosa, permitindo maior flexibilidade no processo e avaliação da UI (Vivian et al., 2014; Cooper et al., 2015).

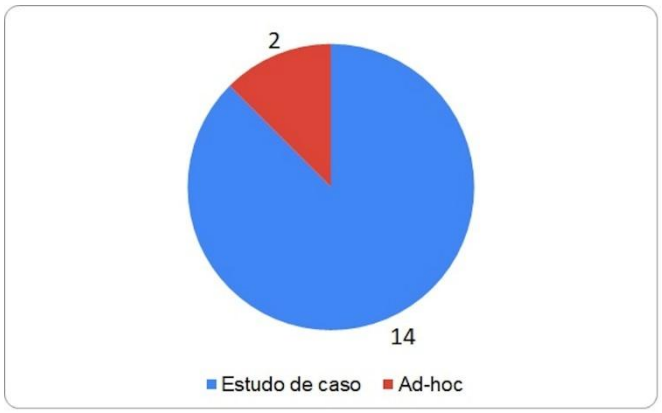

Figura 7: Tipos de estudo.

Nas UI observadas predomina a avaliação de variados fatores (Figura 8). O conhecimento em computação e a satisfação dos participantes são os fatores mais avaliados,

demonstrando a importância do professor sentir que o esforço está trazendo benefícios para seu conhecimento e possuir a compreensão do conteúdo. Alguns estudos também buscam avaliar a aprendizagem (Martinez et al., 2016; Cooper et al., 2015; von Wangenheim et al., 2017; Alimisis et al., 2009) e a capacidade do professor utilizar o que aprendeu em suas aulas (Kay & Moss, 2012; Haden et al, 2016; Rusu, 2015; Alimisis et al., 2009). Outros fatores avaliados são o nível de dificuldade da UI (Partanen et al., 2016; Liu et al, 2015; von Wangenheim et al., 2017), a utilidade dos conteúdos apresentados (Partanen et al., 2016; Cho et al., 2014), o entusiasmo dos participantes (Partanen et al., 2016; von Wangenheim et al., 2017), sua participação nas aulas (Martinez et al., 2016; Vivian et al., 2014), a atitude (Kay & Moss, 2012; Rusu, 2015), experiência (Vivian et al., 2014) e competência dos professores (Liu et al., 2011), além da qualidade da unidade desenvolvida (Rusu, 2015; Alimisis et al., 2009).

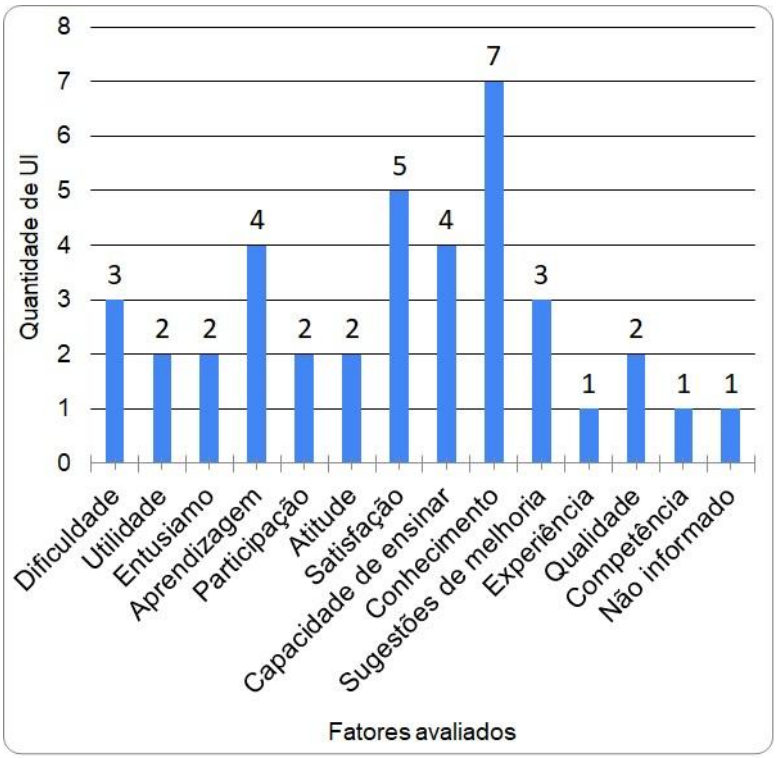

Figura 8: Fatores avaliados.

Os questionários são a principal forma para efetuar a coleta de dados, mas outros métodos também são utilizados (Figura 9). Em algumas UI, os participantes fornecem *feedback* durante uma entrevista com os pesquisadores (Liu et al., 2015) ou a um avaliador externo (Liu et al., 2011). Também foi aplicada avaliação por pares baseada na criação de um recurso de ensino e/ou um plano de aula (Vivian et al., 2014) e observações do comportamento dos professores em sala de aula (Martinez et al., 2016). Em outros trabalhos, foram utilizados autoavaliação do grau de conhecimento (Liu et al., 2013) e diário de ensino (Geldreich et al., 2018).

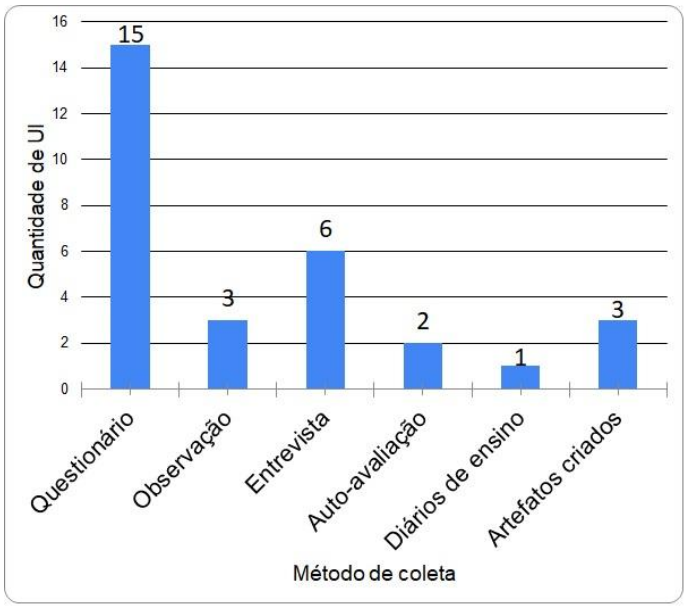

Figura 9: Método de coleta de dados.

Os métodos adotados para a análise dos dados não foram mencionados na metade das publicações (Figura 10). A análise quantitativa foi utilizada por quatro UI, adotando testes de hipótese e/ou estatística descritiva (Kay & Moss, 2012; Cooper et al., 2015; Liu et al., 2013; Cho et al., 2014). A análise qualitativa também foi utilizada para a avaliação de duas UI (Rusu, 2015; Martinez et al., 2016). Outras duas avaliações mencionaram o uso de análise qualitativa e quantitativa (von Wangenheim et al., 2017; Geldreich et al., 2018).

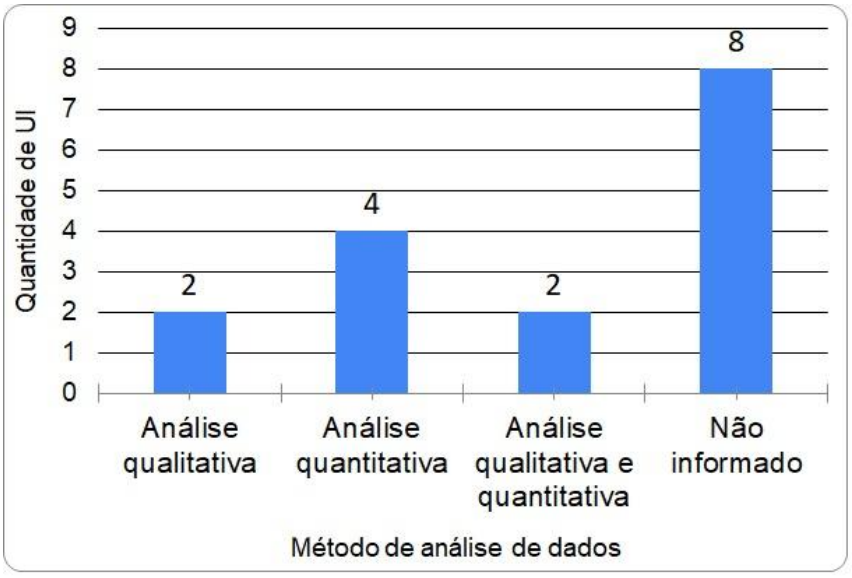

Figura 10: Método de análise de dados.

As amostras utilizadas pelas pesquisas em geral variam de 1 a 65 participantes, possivelmente correspondendo a uma turma de aplicação da UI (Figura 11). Entretanto, cinco publicações apresentam um tamanho maior que 100 professores (Partanen et al., 2016; Martinez et al., 2016; Cooper et al., 2015; Vivian et al., 2014; Hamner et al., 2016).

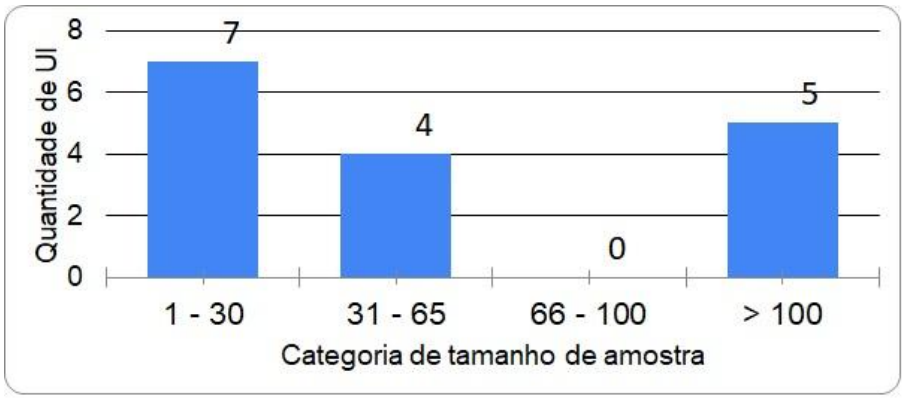

Figura 11: Tamanho da amostra.

As UI selecionadas não têm seus estudos replicados em mais de um contexto na metade dos casos (8 UI). Enquanto em seis UI os estudos foram replicados em mais do que uma turma/contexto (Cho et al., 2014; Rusu, 2015; Haden et al, 2016; Hamner et al., 2016; Kay & Moss, 2012; Martinez et al., 2016; Partanen et al., 2016). Em duas publicações essa informação sobre possível reprodução não foi encontrada (Alimisis et al., 2009; Liu et al., 2011).

Tabela 9: Visão geral da avaliação da qualidade das UI.

Referências	**Design de pesquisa**	**Fator(es) avaliado (s)**	**Método(s) de coleta de dados**	**Tamanho da amostra**	**Replicação do estudo**	**Método(s) de análise de dados**
Partanen et al., 2016	Estudo de caso	Nível de dificuldade, utilidade do conteúdo, entusiasmo dos participantes	Artefatos criados e autoavaliação	540 professores	Replicado	NI
Martinez et al., 2016	Estudo de caso	Aprendizagem e participação dos professores	Questionários e observação	106 professores	Replicado	Análise qualitativa
Kay & Moss, 2012; Kay et al., 2014	Estudo de caso	Atitude, capacidade de ensinar e satisfação dos professores	Questionários	44 professores	Replicado	Análise quantitativa (teste de hipótese)
Cooper et al., 2015; Cooper et al., 2011	Ad-hoc	Conhecimento em programação, satisfação dos professores e aprendizagem	Questionários e artefatos criados	> 100	Não replicado	Análise quantitativa (teste de hipótese)
Liu et al., 2015	Estudo de caso	Conhecimento de computação, satisfação dos professores, dificuldade do material instrucional	Questionários e entrevistas	13 professores	Não replicado	NI
Liu et al., 2013	Estudo de caso	Conhecimento de computação, sugestões de melhoria	Questionários, autoavaliação do grau de conhecimento e observações	14 participantes	Não replicado	Análise quantitativa (estatística descritiva)
Vivian et al., 2014	*Ad-hoc*	Participação e experiência dos participantes	Questionários e artefatos criados	1378 professores	Não replicado	NI
Hamner et al., 2016	Estudos de caso	NI	Questionários, entrevistas e observações de aula.	222 professores	Replicado	NI
Haden et al., 2016	Estudo de caso	Capacidade de ensinar	Questionário	14 professores	Replicado	NI
Liu et al., 2014	Estudo de caso	Conhecimento de computação	Questionários e entrevista	16 professores	Não replicado	NI

Rusu, 2015	Estudo de caso	Qualidade da oficina, sugestões de melhoria, capacidade de ensinar, atitude e conhecimento de computação	Questionários	37 professores	Replicado	Análise qualitativa
Geldreich et al., 2018	Estudo de caso	Capacidade de ensinar computação e satisfação	Questionário, entrevistas e diários de ensino	38 professores	Não replicado	Análise quantitativa e qualitativa usando estatística descritiva
von Wangenheim, 2015	Estudo de caso	Dificuldade, conhecimento adquirido, entusiasmo de ensinar computação, sugestões de melhoria e aprendizagem	Questionários	16 professores	Não replicado	Análise quantitativa e qualitativa usando estatística descritiva
Alimisis et al., 2009	Estudo de caso	Qualidade do curso, aprendizagem e capacidade de ensinar	Questionário e entrevista coletiva.	23 professores	NI	NI
Cho et al.,2014	Estudo de caso	Utilidade de desenvolver habilidades em computação	Questionários	47 professores	Replicado	Análise quantitativa
Liu et al., 2011	Estudo de caso	Competência e conhecimento em computação	Questionário e entrevista por avaliador externo	7 professores	NI	NI

O *feedback* das UI, em geral, foi positivo, indicando que o nível de dificuldade e a carga de trabalho estavam razoáveis (Partanen et al., 2016). O conhecimento em programação foi aprimorado (Kay & Moss, 2012; Cooper et al., 2015; Liu et al., 2015; Liu et al., 2013; Liu et al., 2014; Liu et al., 2011), com o desenvolvimento da capacidade de transferir o aprendizado para suas próprias salas de aula (Haden et al, 2016; Rusu, 2015). No entanto, observou-se que essas UI foram insuficientes para preparar professores sem experiência anterior (Martinez et al., 2016), os quais se sentem desconfortáveis para responder perguntas sobre computação aos alunos (Geldreich et al., 2018). Os resultados também destacam que, para permitir que o professor aplique oficinas de forma eficaz, cursos de formação continuada mais longos e suporte contínuo são necessários (von Wangenheim et al., 2017). Alguns participantes avaliaram a computação como benéfica para o ensino (Hamner et al., 2016), no entanto, muitos professores não conseguiram completar o curso (Vivian et al., 2014). Outro ponto observado é que aprender a pedagogia é tão importante quanto aprender conceitos de computação, porque simplesmente ensinar algoritmos e definições pode não promover uma aprendizagem significativa da disciplina (Martinez et al., 2016).

5 Discussão

Em geral, o número de UI encontradas visando à formação continuada de professores *in-service* referentes ao ensino de competências de computação para os Anos Finais do Ensino Fundamental foi muito pequeno (apenas 16 no total). Isso mostra a escassez de trabalhos nessa área e a necessidade de desenvolvimento de mais UI para esse fim.

Em termos de conceitos e práticas de computação ensinados nas UI, observou-se que a maioria foca em conteúdos relacionados a algoritmos e programação. As publicações também destacaram o ensino de competências pedagógicas aos professores, enquanto competências tecnológicas foram abordadas somente em poucas UI.

As UI geralmente possuem uma organização dos tópicos bem definida, dividindo o conteúdo em seções. Vários métodos são utilizados para ensinar esse conteúdo, mas a maioria emprega abordagens de aprendizagem ativa. Tipicamente ocorre a introdução de um

conceito, demonstrações de exemplos e em seguida os participantes desenvolvem algum artefato. Em outras UI, os professores realizam a aula com base em tutoriais e/ou resolução de problemas. O conteúdo é explicado aos professores utilizando diferentes materiais instrucionais, como vídeos, slides, tutoriais e exemplos, proporcionando caminhos de aprendizagem flexíveis e a mistura de conteúdo e exemplos (Vivian et al., 2014). A aprendizagem e o desempenho dos professores nessas aulas são medidos por meio de questionários, observações e artefatos criados.

Os diversos ambientes e plataformas de programação usados também chamam a atenção, abrangendo desde robótica utilizando o LEGO NXT-G (Alimisis et al., 2009) até aplicativos, como App Inventor (Liu et al., 2013) e sistemas para *desktop*, como Scratch (Geldreich et al., 2018). Algumas UI utilizam múltiplas linguagens/plataformas, pois acreditam que ensinar programação não se trata do uso de uma plataforma ou o aprendizado de uma linguagem de programação específica, mas sobre como aprender conceitos de computação (Martinez et al., 2016).

De forma geral, observa-se uma falta de informações sobre a base teórica e o método de desenvolvimento das UI. Apenas três UI apresentam informações sobre a metodologia utilizada para desenvolvê-las sistematicamente, enquanto as outras não apresentam nenhuma informação sobre a base teórica empregada no desenvolvimento. Em razão da falta de rigor científico, a maioria das UI foi avaliada somente por estudos de casos com pequenas amostras e/ou de forma *ad-hoc*.

Ameaças à validade. Com a intenção de minimizar os impactos de ameaças à validade deste estudo de mapeamento, foram tomadas várias ações. Mapeamentos podem sofrer do viés comum de que os resultados positivos têm maior probabilidade de serem publicados do que os negativos (Kitchenham, 2004). Ainda assim, é considerado que esse fator tem apenas uma pequena influência, pois a pesquisa buscou as características de UI para ensinar computação a professores.

Uma ameaça é a omissão de estudos relevantes. Com o intuito de minimizar esse risco foram realizadas pesquisas em várias bases de dados, utilizando também sinônimos na *string* de busca. Para reduzir esse risco ainda mais foram realizadas também buscas no *Google*, além da análise das referências dos artigos encontrados.

A ameaça à seleção de publicações foi reduzida por meio da definição e documentação de um protocolo para a escolha criteriosa de artigos, assim como pela definição detalhada de critérios de inclusão/exclusão. Mesmo encontrando dificuldades na extração de informações das publicações relevantes pela falta de um protocolo de relato nessa área, o processo foi realizado de forma cuidadosa por todos os coautores até que um consenso foi obtido.

6 Conclusão

Neste artigo, apresentamos o estado da arte no ensino de competências de computação para professores *in-service* da Educação Básica, especialmente do 6º ao 9º ano do Ensino Fundamental. No total foram encontradas 16 UI, o que consideramos um número muito pequeno. Essas UI focam principalmente no ensino de conceitos de algoritmos e programação, abordando também conhecimento pedagógico; uma quantidade menor de UI abordam conhecimento tecnológico. Em geral, essas UI são realizadas na forma de oficinas ou cursos com uma duração variando de um ou dois dias até duas semanas. A maioria das UI

visa que os professores possuam o nível de aprendizagem de aplicação após a UI, adotando abordagens ativas de aprendizado, levando os participantes a criar artefatos e resolver problemas relacionados a computação e em alguns casos até a desenvolver o seu próprio material educacional para o uso posterior nas suas disciplinas.

Observamos o baixo rigor científico no desenvolvimento das UI, pela falta de apresentação de algumas informações sobre a avaliação da qualidade da UI. A maioria das UI foi avaliada somente via estudos de caso e/ou de forma *ad-hoc* com amostras pequenas e com uma grande variação dos fatores avaliados. Dessa forma fica evidente a necessidade de futuras pesquisas nessa área para compreender melhor quais conteúdos e estratégias instrucionais são importantes para assegurar uma formação eficaz e eficiente dos professores, que de fato lhes possibilitem incluir o ensino de computação a suas disciplinas.

Agradecimentos

Este trabalho foi apoiado pelo CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), uma entidade do governo brasileiro focada no desenvolvimento científico e tecnológico.

Referências

Alice. Alice – Tell Stories. Build Games. Learn to Program. Disponível em: <https://www.alice.org/>. Acesso em: 12 de junho de 2019.

Alimisis, D.; Frangou, S.; Papanicolaou, K. (2009). A Constructivist Methodology for Teacher Training in Educational Robotics. In: Proceedings of the TERECoP Course in Greece through Trainees Eyes. In: Proc. of the 9th IEEE International Conference on Advanced Learning Technologies, Riga, Latvia. doi: 10.1109/ICALT.2009.86.

Armoni, M. (2011). Looking at Secondary Teacher Preparation Through the Lens of Computer Science. ACM Transactions on Computing Education, 11(4), Article 23. doi: 10.1145/2048931.2048934.

Barr, D.; Harrison J.; Conery, L. (2011) Computational Thinking: A Digital Age Skill for Everyone. Learning & Leading with Technology, 5191, 20-23. [GS Search]

Barr, V.; Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. ACM Inroads, 2(1). doi: 10.1145/1929887.1929905.

Berges, M.; Hubwieser, P.; Magenheim, J.; Bender, E.; Margaritis-Kopecki, M.;Neugebauer, J.; Schaper, N.; Schubert, S.; Ohrndorf, L. (2013). Developing a competency model for teaching computer science in schools. In: Proc. of the 18th ACM Conference on Innovation and Technology in Computer Science Education, Cambridge, Reino Unido, 327-327. doi: 10.1145/2462476.2465607.

Bloom, B. S.; Engelhart, M. D.; Furst, E. J.; Hill, W. H.; Krathwohl, D. R. A. (1956). Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook 1: Cognitive Domain. New York: David McKay. [GS Search]

Brasil. Decreto nº 8.752, de 9 de maio de 2016. Dispõe sobre a Política Nacional de Formação dos Profissionais da Educação Básica. Diário Oficial da União: Seção 1, Brasília, DF: Atos do Poder Executivo, n. 88, p. 5-6, 2016. Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2016/Decreto/D8752.htm>. Acesso em: 12 de junho de 2019.

Bower, M.; Wood, L. N.; Lai, J. W.; Howe, C.; Lister, R.; Mason, R.; Highfield, K.; Veal, J. (2017). Improving the Computational Thinking Pedagogical Capabilities of School Teachers. Australian Journal of Teacher Education, 42(3). doi: 10.14221/ajte.2017v42n3.4.

Cao, X.; He, K. (2009). Web-Based Training Framework for China School Teachers Educational Technique Ability Training. In: Proc. of the First International Workshop on Education Technology and Computer Science, Wuhan, Hubei, 626-630. doi: 10.1109/ETCS.2009.673.

Chen, G.; Shen, J., Barth-Cohen, L.; Jiang, S.; Huang, X.; Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. Computers & Education, 109, 162–175. doi: 10.1016/j.compedu.2017.03.001.

Cho, S.; Pauca, P.; Johnson, D.; James, Y. (2014). Computational Thinking for the Rest of Us: A Liberal Arts Approach to Engaging Middle and High School Teachers with Computer Science Students. In: Proc. of Society for Information Technology & Teacher Education International Conference. Jacksonville, FL, EUA, 79-86. [GS Search]

Cooper, S.; Dann, W.; Lewis, D.; Lawhead, P.; Rodger, S.; Schep, M.; Stalvey, R. H. (2011). A pre-college professional development program. In: Proc. of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education. Darmstadt, Alemanha, 188-192. doi: 10.1145/1999747.1999801.

Cooper, S.; Rodger, S. H.; Schep, M.; Stalvey, R. H.; Dann, W. (2015). Growing a K-12 Community of Practice. In: Proc. of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, MO, EUA, 290-295. doi: 10.1145/2676723.2677255.

CSTA (2011). The CSTA Standards Task Force. CSTA K-12 Computer Science Standards Revised 2011, New York, NY, EUA. Disponível em: <http://scratch.ttu.ee/failid/CSTA_K-12_CSS.pdf>. Acesso em: 12 de junho de 2019.

CSTA. (2016). K-12 Computer Science Framework. Disponível em: <http://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf>. Acesso em: 12 de junho de 2019.

De Kereki, I. F.; Manataki, A. (2016). "Code Yourself" and "A Programar": a bilingual MOOC for teaching Computer Science to teenagers. In: Proc. of the Frontiers in Education Conference, Erie, PA, EUA. doi: 10.1109/FIE.2016.7757569.

Dreyfus, S. E.; Dreyfus, H. L. (1980). A Five-Stage Model of the Mental Activities Involved in Directed Skill Acquisition. Washington: Storming Media. [GS Search]

Gal-Ezer, J.; Stephenson, C. (2010). Computer science teacher preparation is critical. ACM Inroads, 1(1), 61-66. doi: 10.1145/1721933.1721953.

Garofalakis, J. (2015). Enhancing teachers' digital skills - Developing digital competencies in the school of tomorrow. In: Proc. of the Classroom Conference, Atenas, Grécia. [GS Search]

Geldreich, K.; Talbot, M.; Hubwieser, P. (2018). Off to new shores: preparing primary school teachers for teaching algorithmics and programming. In: Proc. of the 13th Workshop in Primary and Secondary Computing Education, Potsdam, Alemanha. doi: 10.1145/3265757.3265783.

Goode, J. (2007). If you build teachers, will students come? the role of teachers in broadening computer science learning for urban youth. Journal of Educational Computing Research 36(1), 65–88. doi: 10.2190/2102-5G77-QL77-5506.

Goode, J. (2008). Increasing Diversity in K-12 Computer Science: Strategies from the Field. In: Proc. of the 39th SIGCSE Technical Symposium on Computer Science Education, Portland, OR, EUA. doi: 10.1145/1352322.1352259.

Google & Gallup (2015). Searching for Computer Science: Access and Barriers in U.S. K-12 Education. Dísponível em : <http://services.google.com/fh/files/misc/searching-for-computer-science_report.pdf>. Acesso em: 12 de junho de 2019.

Google & Gallup (2016). Trends in the State of Computer Science in U.S. K-12 Schools. K-12 Education. Dísponível em: <https://services.google.com/fh/files/misc/trends-in-the-state-of-computer-science-report.pdf>. Acesso em: 12 de junho de 2019.

Granor, N.; DeLyser, L. A.; Wang, K. (2016). TEALS: Teacher Professional Development Using Industry Volunteers. In: Proc. of the 47th ACM Technical Symposium on Computing Science Education, Memphis, TN, EUA, 60-65. doi: 10.1145/2839509.2844589.

Gresse von Wangenheim, C.; Alves, N. d. C.; Rodrigues, P. E.; Hauck, J. C. R. (2017). Teaching Computing in a Multidisciplinary Way in Social Studies Classes in School – A Case Study. International Journal of Computer Science Education in Schools, 1(2). doi: 10.21585/ijcses.v1i2.9.

Grover, S.; Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38–43. doi: 10.3102/0013189X12463051.

Guzdial, M. (2008). Education Paving the way for computational thinking. Communications of the ACM. 51(8), 25-27. doi: 10.1145/1378704.1378713.

Haden, P.; Gasson, J.; Wood, K.; Parsons D. (2016). Can you learn to teach programming in two days?. In: Proc. of the Australasian Computer Science Week Multiconference, Canberra, Australia. doi: 10.1145/2843043.2843063.

Hamlen, K.; Sridhar, N.; Bievenue, L.; Jackson, D. K.; Lalwani, A. (2018). Effects of Teacher Training in a Computer Science Principles Curriculum on Teacher and Student Skills, Confidence, and Beliefs. In: Proc. of the 49th ACM Technical Symposium on Computer Science Education. Baltimore, MD, EUA, 741-746. doi: 10.1145/3159450.3159496.

Hamner, E.; Cross, J.; Zito, L.; Bernstein D.; Mutch-Jones, K. (2016). Training teachers to integrate engineering into non-technical middle school curriculum. In: Proc. of the IEEE Frontiers in Education Conference, Erie, PA, EUA. doi: 10.1109/FIE.2016.7757486.

Harris, J.; Koehler, M.; Mishra, P. (2009). What Is Technological Pedagogical Content Knowledge? Contemporary Issues in Technology and Teacher Education. 9. [GS Search]

Hill H.; Rowan B.; Ball D. (2005). Effects of Teachers' Mathematical Knowledge for Teaching on Student Achievement. American Educational Research Journal, 42(2), pp. 371-406. doi: 10.3102/00028312042002371.

Hodhod, R.; Khan, S.; Kurt-Peker, Y.; Ray, L. (2016). Training Teachers to Integrate Computational Thinking into K-12 Teaching. In: Proc. of the 47th ACM Technical Symposium on Computing Science Education, Memphis, TN, EUA, 156-157. doi: 10.1145/2839509.2844675.

INEP (2018). Sinopse estatística da Educação Superior 2018. Brasília: Inep, 2019. Dísponível em: <http://inep.gov.br/sinopses-estatisticas-da-educacao-superior>. Acesso em: 27 de novembro de 2019.

ISTE (2019). ISTE Standards for Computer Science Educators. Dísponível em : <https://www.iste.org/standards/for-computer-science-educators>. Acesso em: 12 de junho de 2019.

Kay, J. S.; Moss, J. G. (2012). Using robots to teach programming to K-12 teachers. In: Proc. of the Frontiers in Education Conference, Seattle, WA, EUA. doi: 10.1109/FIE.2012.6462375.

Kay, J. S.; Moss, J. G.; Engelman, S.; MacKlin, T. (2014). Sneaking in through the back door: introducing k-12 teachers to robot programming. In: Proc.of the 45th ACM Technical Symposium on Computer Science Education. Atlanta, GA, EUA, 499-504. doi: 10.1145/2538862.2538972.

Kelleher, K.; Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing surveys, 37(2), 83-137. doi: 10.1145/1089733.1089734.

Kitchenham, B. (2004). Procedures for Performing Systematic Reviews. Technical Report Keele University, Keele, UK. [GS Search]

Koellner, K.; Jacobs, J. (2015). Knowledge, Instruction, and Student Achievement Distinguishing Models of Professional Development: The Case of an Adaptive Model's Impact on Teachers' On behalf of: American Association of Colleges for Teacher Education. Journal of Teacher Education. 66, 51-67. doi: 10.1177/0022487114549599.

Lamprou, A.; Repenning, A.; Escherle, N. A. (2017). The Solothurn Project — Bringing Computer Science Education to Primary Schools in Switzerland. In: Proc. of the 22th Annual Conference on Innovation and Technology in Computer Science Education. Bologna, Itália. doi: 10.1145/3059009.3059017.

Liu, J.; Lin, C. H.; Hasson, E. P.; Barnett, Z.; Xu, Y. (2011). Introducing computer science to K-12 through a summer computing workshop for teachers. In: Proc. of the 42nd ACM Technical Symposium on Computer Science Education, Dallas, TX, EUA. doi: 10.1145/1953163.1953277.

Liu, J.; Lin, C. H.; Potter P.; Hasson, E. P.; Barnett, Z.; Xu, Y.; Singleton, M. (2013). Going mobile with app inventor for android: a one-week computing workshop for K-12 teachers. In: Proc. of the 44th ACM Technical Symposium on Computer Science Education, Denver, CO, EUA, 433-438. doi: 10.1145/2445196.2445324.

Liu, J.; Lin, C. H.; Wilson, J.; Hemmenway, D.; Hasson, E. P.; Barnett, Z.; Xu, Y. (2014). Making games a "snap" with Stencyl: a summer computing workshop for K-12 teachers. In: Proc. of the 45th ACM Technical Symposium on Computer Science Education, Atlanta, GA, EUA. doi: 10.1145/2538862.2538978.

Liu, J.; Wilson, J.; Hemmenway, D.; Xu Y.; Lin, C. (2015). Oh SNAP! A one-week summer computing workshop for K-12 teachers. In: Proc. of the 10th International Conference on Computer Science & Education, Cambridge, Reino Unido. doi: 10.1109/ICCSE.2015.7250329.

Lye, S. Y.; Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? Computers in Human Behavior, 41(C), 51-61. doi: 10.1016/j.chb.2014.09.012.

Lockwood, J.; Mooney, A. (2017). Computational Thinking in Education: Where does it Fit? A systematic literary review. International Journal of Computer Science Education in Schools. 2. [GS Search]

Lloyd, M.; Cochrane, J. (2006). Celtic knots: Interweaving the elements of effective teacher professional development in ICT. 21. [GS Search]

Mannila, L.; Dagiene, V.; Demo, B.; Grgurina, N.; Mirolo, C.; Rolandsson, L.; Settle A. (2014). Computational Thinking in K-9 Education. In: Proc. of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference. New York, NY, EUA, 1-29. doi: 10.1145/2713609.2713610.

Martin, N.; Soares, A. (2016). Organizing an App Inventor Summer Camp for Middle School Girls: What the Experts Don't Tell You. Information Systems Education Journal (ISEDJ), 14. [GS Search]

Martinez, M. C.; Gomez, M. J.; Moresi, M.; Benotti, L. (2016). Lessons Learned on Computer Science Teachers Professional Development. In: Proc. of the ACM Conference on Innovation and Technology in Computer Science Education. Arequipa, Peru, 77-82. doi: 10.1145/2899415.2899460.

Milton, M.; Rohl, M.; House, H. (2007). Secondary Beginning Teacher's Preparedness to Teach Literacy and Numeracy: A Survey. Australian Journal of Teacher Education, 32(2). [GS Search]

Mishra, P.; Koehler, M. J. (2006). Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge, 108(6), 1017-1054. [GS Search]

MITa (2019). App Inventor - Explore. Disponível em: <http://appinventor.mit.edu/explore/>. Acesso em: 12 de junho de 2019.

MITb (2019). Scratch. Disponível em: <https://Scratch.mit.edu/>. Acesso em: 12 de junho de 2019.

Morelli, R.; Uche, C.; Lake, P.; Baldwin L. (2015). Analyzing Year One of a CS Principles PD Project. In: Proc.of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, MO, EUA. doi: 10.1145/2676723.2677265.

Moreno-León, J.; Robles, G. (2015). Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects. In: Proc. off the 10th Workshop in Primary and Secondary Computing Education, Londres, Reino Unido, 132–133. doi: 10.1145/2818314.2818338.

Naughton, J. (2012). Why All Our Kids Should Be Taught How to Code. The Guardian. Guardian News and Media. Disponível em: https://www.theguardian.com/education/2012/mar/31/why-kids-should-be-taught-code Acesso em: 12 de junho de 2019.

Ni, L.; Guzdial, M. (2012). Who am I? understanding high school computer science teachers' professional identity. In: Proc. of the 43rd Technical Symposium on Computer Science Education, Raleigh, NC, EUA, 499-504. doi: 10.1145/2157136.2157283.

Papadakis, S.; Orfanakis, V. (2018). Comparing novice programing environments for use in secondary education: App Inventor for Android vs. Alice. International Journal of Technology Enhanced Learning (IJTEL), v. 10, n. 1/2, p. 44. doi: 10.1504/IJTEL.2018.088333.

Partanen, T.; Mannila, L.; Poranen, T. (2016). Learning programming online: a racket-course for elementary school teachers in Finland. In: Proc. of the 16th Koli Calling International Conference on Computing Education Research, Koli, Finland, 178-179. doi: 10.1145/2999541.2999567.

Petersen, K.; Vakkalanka, S.; Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: an update. Information and Software Technolology, 64, p. 1–18. doi: 10.1016/j.infsof.2015.03.007.

Ravitz, J.; Stephenson, C.; Parker, K.; Blazevski, J. (2017). Early Lessons from Evaluation of Computer Science Teacher Professional Development in Google's CS4HS Program. ACM Transactions on Computing Education, 17(4), Article 21. doi: 10.1145/3077617.

Resnick, M.; Ocko, S. (1991). Lego /LogoLearning Through and About Design. In: S. Papert and I. Harel Constructionism. New Jersey:Ablex Publishing Corporation, 141-150. [GS Search]

Romero, M.; Lepage, A.; Lille, B. (2017). Computational thinking development through creative programming in higher education. International Journal of Educational Technology in Higher Education, 14. doi: 10.1186/s41239-017-0080-z.

Rusu, C.; Roncagliolo, S.; Rusu, V.; Collazos, C. (2011). A Methodology to establish usability heuristics. In: Proc. of the 4th International Conference on Advances in Computer-Human Interactions, Guadeloupe, France, 59-62. [GS Search]

Rusu, A. (2015). Introducing gaming tools for computing education in STEM related curricula. In: Proc. of the IEEE Frontiers in Education Conference, Dallas, TX, EUA. doi: 10.1109/FIE.2015.7344166.

Sardessai, S.; Kamat, V. V. (2011). Using a Blended Approach to In-service Teacher Training: A Case Study. In: Proc. of the IEEE International Conference on Technology for Education. Washington, EUA, 266-269. doi: 10.1109/T4E.2011.54.

SBC (2017). Referenciais de Formação em Computação: Educação Básica. Disponível em: <http://www.sbc.org.br/documentos-da-sbc/send/131-curriculos-de-referencia/1166-referenciais-de-formacao-em-computacao-educacao-basica-julho-2017>. Acesso em: 12 de junho de 2019.

Schiller, J.; Turbak, F.; Abelson, H.; Dominguez, J.; McKinney, A.; Okerlund, F.; Friedman, M. (2014). Live Programming of Mobile Apps in App Inventor. In: Proceedings of the 2nd Workshop on Programming for Mobile & Touch. Portland, EUA, 1-8. doi: 10.1145/2688471.2688482.

Selby, C. C. (2012). Promoting computational thinking with programming. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education (WiPSCE '12). New York, EUA, 74-77. doi: 10.1145/2481449.2481466.

Spradling, C.; Linville, D.; Rogers, M. P.; Clark, J. (2015). Are MOOCs an appropriate pedagogy for training K-12 teachers computer science concepts? Journal no Computer Science in Colleges, 30(5), 115-125. [GS Search]

Vivian, R.; Falkner, K.; Falkner, N. (2014). Addressing the challenges of a new digital technologies curriculum: MOOCs as a scalable solution for teacher professional development. Research in Learning Technology, 22. doi: 10.3402/rlt.v22.24691.

von Wangenheim, A.; Gresse von Wangenheim, C.; Pacheco, F. S.; Hauck, J.; Ferreira, M. N. (2017). Motivating Teachers to Teach Computing in Middle School – A Case Study of a Physical Computing Taster Workshop for K-12 Teachers. International Journal of Computer Science Education in Schools, 35(1), 368-373. doi: 10.21585/ijcses.v1i4.17.

Wenger, E.; McDermott, R. A.; Snyder, W. (2002). Cultivating communities of practice: a guide to managing knowledge. Boston: Harvard Business School Press.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35. doi: 10.1145/1118178.1118215.

Yadav, A.; Hong, H.; Stephenson, C. (2016). Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms. TechTrends, 60(6), 565-568. doi: 10.1007/s11528-016-0087-7.