Smart Learning Environments

RESEARCH Open Access

A conceptual framework for teaching computational thinking in personalized OERs

Jewoong Moon[1], Jaewoo Do[2*], Daeyeoul Lee[3] and Gi Woong Choi[4]

* Correspondence: jaewoo.do@gmail.com
[2]Korean Educational Development Institute(KEDI), 7, Gyohak-ro, Deoksan-eup, Jincheon-gun, Chungcheongbuk-do, Republic of Korea
Full list of author information is available at the end of the article

Abstract

Interests towards teaching programming skills have risen recently in the realm of computing education. Learning how to program not only enables learners to develop computing applications, but it can also enhance learners' computational thinking (CT) practice. CT refers to learners' ability to approach ill-structured tasks systematically based on algorithmic thinking in computing. Along with growing academic interests towards CT in recent studies, researchers have emphasized the role of teaching programming in facilitating learners' problem-solving skills. Emerging OERs have expanded learners' opportunities to engage in hands-on programming exercises; yet a challenge still remains as to how learners' programming exercises can be tailored to accommodate individual differences in terms of learners' digital literacy skills. There is still a lack of in-depth discussions on how to support learners' personalized learning experiences during programming exercises associated with CT. This study hence proposes a conceptual framework that seeks to consider how to promotelearners' personalized learning experiences and enhance their CT skills in OERs. Through extensive reviewing of literature, this study provides several implications for further research.

Keywords: Open Educational Resource, Computational Thinking, Personalization, Learning Analytics

Growing interests towards twenty-first-century skills have brought scholars' attention to the further role of smart learning environments in improving students' digital literacies. Learners' digital literacies span beyond becoming familiar with particular technologies, and encompass comprehensive skills to implement and apply technologies in learners' problem-solving contexts (Griffin & Care, 2014), that are closely associated with learners' computational thinking (CT). CT refers to a way of problem-solving, including systematic analyses and implementations (Shute, Sun, & Asbell-Clarke, 2017). CT aims to enhance learners' potentials in solving ill-structured problems by enabling learners to go through a series of computing executions based on computational logic. CT is not just about computer skills, it is a set of thinking skills—such as algorithmic, design, and mathematical thinking that are vital to solving problems using a computer (Lee et al., 2011, p.32).

This study focused on learners' programming exercises in developing personalized OER design framework for learners' CT development. While we highlight that CT goes beyond mere acquisition of programming skills (Shute et al., 2017), it is still useful to

utilize programming exercises as a way to improve CT competencies (Hoppe & Werneburg, 2019; Lye & Koh, 2014). Engaging learners in programming exercises not only improve their skills in a specific programming language, it could enhance their CT competencies. CT is not a single entity, but a multifaceted concept—comprising multiple abilities that are needed when applying skills for computing-application developments. Attaining a sense of CT requires learners' thorough understanding of a particular problem context, as well as programming languages and algorithms. Prior research indicated that the first objective of CT is to learn how to solve a real-world issue by constructing computing applications (Aho, 2012). Constructionists have concentrated on leveraging learners' CT through hands-on programming exercises that contain genuine and complicated problem-solving tasks.

Many online platforms as open educational resources (OERs) have emerged to teach CT, —ranging from Massive Open Online Course (MOOC) (Bonk, Lee, Reeves, & Reynolds, 2015) to gamified coding platforms such as *CodeCombat* (Saines, Erickson, & Winter, 2013), *Hour of Code* (Wilson, 2014), and *ctGameStudio* (Werneburg, Manske, & Hoppe, 2018). These OERs have widened individuals' learning opportunities for computer programming. The emergence of visual programming and smart-computing technologies empowered learners to easily apply their programming skills to their daily lives. Compared to traditional face-to-face environments, which are mostly instructor-led within classroom-like settings, online OER platforms enable learners to practice programming languages without constraints on time and location. Such OERs on computing education have become potential environments where learners can choose educational materials that are tailored to their educational needs.

OERs on computing education have evolved to promote students' deeper understanding of CT via highly-interactive programming exercises. A collection of block-based programming platforms enabled learners to deeply engage with manipulating programming language (Resnick et al., 2009; Wilkerson-Jerde, Wagh, & Wilensky, 2015). Agent-based and gameful learning contexts in programming platforms (Werneburg et al., 2018; Wilensky, 1999) also contributed to enhancing students' situated learning in CT. Others allowed learners to engage in how their programming exercises can be connected with other disciplines (e.g., Science, Technology, Engineering, and Mathematics (STEM)) (Sengupta & Farris, 2012; Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013). However, despite increasing developments of OERs on computing education, there are still challenges to overcome. While OER platforms enable learners to be pervasively involved in programming exercises online, a way to systematically enhance learners' CT through programming exercises is not well understood. When considering the multifaceted and complex nature of CT, additional support to personalize learners' experiences is essential. Conceptualizing a framework to lay out how learners' CT is assessed and promoted provides a better understanding of how to further design and develop personalized OERs with programming exercises. With this in mind, this study explores how OERs can be designed to support learners' CT improvements through personalized instruction.

Purpose

This study proposes a conceptual framework that illustrates how to support learners' CT developments through personalized designs of OERs. This study investigated how OERs can provide learners with personalized learning experiences in programming exercises.

This study proposes a learning-analytics approach to integrating personalization within an OER framework. Based on extensive literature reviews, this study explored latent attributes of CT and illustrated how learning analytics (LA) can enhance learners' CT in OERs. This study has an overarching research question as follows: What is a conceptual framework for executing personalized OERs to support learners' CT improvement?

Underlying foundations

Computational thinking

Essential elements of computational thinking

As early constructionist scholars revealed the impact of programming exercises in education (Harel & Papert, 1991; Kafai, 2006; Kafai & Burke, 2013; Papert & Harel, 1991), researchers have continuously investigated how learners' programming exercises would promote learners' CT. After Wing (2006) came up with the term *computational thinking*, scholars have explored how programming exercises can enhance learners' CT. Initially, programming exercises were considered to be learning activities that only focus on building computing applications, but the concept of CT has expanded such a notion.

Generally, CT is described as learners' capabilities to solve genuine and complex problems using a way of thinking as either computer or computer scientist approach. CT is not limited to logical actions for computer programming but broadly describes the approach to systematically perform their problem-solving skills toward various challenges. There have been scholarly discussions surrounding the definition of CT. According to Brennan and Resnick (2012), CT is associated with the essence of programming skills. Brennan and Resnick stated several CT concepts: (a) sequence, (b) loops, (c) parallelism, (d) events, (e) conditionals, (f) operators, and (g) data. Both sequence and loop stand for learners' procedures to develop programming codes. Parallelism addresses the design of multi-processors of information. Events refer to the code development of how computing applications detect external inputs by users. Conditionals are a set of thresholds to operate the request based on different computing circumstances. Operators are a set of computer codes that are represented by numerical signs. Lastly, data stands for information that requires additional decoding. Shute et al. (2017) explained that CT is a way to resolve complicated problems strategically. Their review addressed a total of six CT facets: (1) decomposition, (2) abstraction, (3) algorithms, (4) debugging, (5) iteration, and (6) generalization. Both decomposition and abstraction include patterning a problem as manageable chunks and then extracting features. Algorithms represent learners' logical reasonings to approach a confined problem. Debugging and iteration are the steps to evaluate their works to find, correct, and refine the errors when implementing a solution. Lastly, generalization is an action seeking opportunities to transfer the solution to broader contexts. Compared to Brennan and Resnick (2012)'s view, their classification takes into consideration CT features that are related to generic problem-solving. Four iterative actions of students which can improve CT that are commonly discussed by researchers are problem decomposition, reformulation, implementations, and evaluations.

Learning environments to enhance computational thinking

A theoretical notion of CT has started from understanding learners' programming actions when building a computing application (Brennan & Resnick, 2012). In order to

improve CT, researchers highlighted the importance of identifying an ill-defined problem (Wing, 2011), formulating a prototype (Settle et al., 2012), and evaluating its functionality (Ota, Morimoto, & Kato, 2016) during programming exercises. Through iterative evaluations in programming exercises, learners can gradually acquire a sense of CT.

Such notions are reflected in constructionist computing platforms. Ever since the introduction of *LOGO* (Papert & Harel, 1991), *NetLogo* (Wilensky, 1999) and many other OERs on CT education have been introduced to promote learners' CT competencies systematically. The key objective of these OERs was to assist students in identifying the logic of computational processing. These platforms were designed and developed to better support students' intuitive understandings of CT. One of the ways to support such understandings is through visual representations. For instance, *Scratch* (Resnick et al., 2009), *AgentSheets* (Repenning & Sumner, 1995), and *ViMap* (Sengupta & Farris, 2012) have utilized the blocks of functions as visual representations to help students understand how computational logics are structured. Oftentimes, these block-based programming settings are integrated into a gamified learning environment. For example, Werneburg et al. (2018) provided guided discovery for learners in their environment where learners are guided and prompted to make decisions on which programming codes should be implemented to solve their in-game challenges.

Beyond current computer-aided and unplugged practices (Bell, Alexander, Freeman, & Grimley, 2009; Wohl, Porter, & Clinch, 2015), several computerized OER platforms also expanded opportunities to engage the public in computer programming. *MIT App Inventor* aimed to promote novice students' awareness of CT when building a computer application. Using *Scratch*, the system encourages learners to experience the self-development of mobile software via their hands-on visual programming practices (Abelson, Wolber, Morelli, Gray, & Uche, 2012). The platform *Hour of Code* by Code.org has offered online interactive modules for programming practices. This online platform was designed for K-12 learners to familiarize themselves with coding practices and practice with gameful online exercises. Similarly, in France, a group of researchers initiated their national project *Class'Codes*, which offers multiple web-interactive programming modules (Informatics Europe, 2017). This self-paced online environment is specialized to offer responsive interfaces of programming exercises (e.g., robotics, network system, information processing, and creative programming).

Despite the large expansion of OERs, several challenges still exist in building OERs to enhance learners' CT. First, the number of guidelines on how to design programming exercises in association with CT is still limited. Although numerous OERs have integrated learners' programming exercises to foster their CT, conceptualizations of OERs to layout associations between programming exercises and CT competencies are still lacking. Second, while various OERs have been designed for self-paced learning, only a limited number of studies considered providing a personalized learning experience that addresses the individual differences of learners during the programming exercises. Because novice learners in computing OERs are likely to experience many challenges, it is necessary to provide in-situ scaffolding to support their success in programming exercises (Sengupta et al., 2013). Hence, personalized learning support is necessary to provide personalized instruction tailored to individual students' progression. However, a lack of design inquiries in prior studies calls for further conceptualizations on how to deliver personalized support to learners, as a lens for formative assessment.

Formative assessments and adaptations of teaching computational thinking

Learners' challenges in CT have been a major issue in computing education. Researchers consistently raised a question on how to support learners' acquisition of CTs throughout programming exercises (Lye & Koh, 2014). Researchers have explored ways to measure learners' CT competency developments during programming exercises. Prior smart learning environments mostly aimed to promote learners' interactive programming exercises; however, their assessments failed to observe learners' in-situ and meaningful improvements of CT in programming exercises (Grover, Cooper, & Pea, 2014; Lee et al., 2012).

Relevant to this claim, researchers have explored ways to develop and implement assessments for CT competencies. Hosseini (2017) attempted to evaluate learners' programming behaviors by using the evidence-centered design (ECD) framework. The ECD framework is a conceptual model that illustrates how learners' knowledge, skills, and abilities (KSA) appear in correspondence to major task features (Mislevy & Haertel, 2006). Under this framework, this study classified key observable variables that are related to learners' programming skills: prior programming experiences, aptitude in mathematics and sciences, and intrinsic motivation and comfort level. The study suggested several programming task products: code compatibility, the correctness of the program, the number of programming code lines, number of learners' requests, time-duration of the code writing, and the basic programming concepts.

Researchers have assessed computational thinking by using diverse methods observing the evidence of CT-associated behaviors. For example, Koh, Basawapatna, Nickerson, and Repenning (2014) developed the computational-thinking pattern analysis (CTPA)—using the latent semantic analysis to detect computational thinking patterns within game design contexts. CTPA enables researchers to find nine computational thinking patterns: user control, generation, absorption, collision, transportation, push, pull, diffusion, and hill-climbing by creating a graph. Wilson, Hainey, and Connolly (2012) refined and used a coding scheme to assess students' CT when using *Scratch*. The coding scheme consists of three main categories: programming concepts, code organization, and designing for usability.

In emerging Educational Data Mining (EDM) and LA research, there is research contributing to the assessments of students' CT. Using the *NetLogo* environment, Blikstein (2011) explored multiple indicators of students' CT-related variables: Code size, the time between compilations, and errors. Qualitative observations of this study found the feasibility of each variable to interpret students' CT-related actions. Werneburg et al. (2018) attempted to observe students' programming action patterns via multiple measures (e.g., runs, changes per run, creates, and consecutive changes per create). Basu, Kinnebrew, and Biswas (2014) introduced vector-distance model-accuracy metrics to compare the similarity between students' and an expert's model. This computational modeling was driven by a bag-of-words computational algorithm under natural language processing (NLP). Brennan and Resnick (2012) used the data-capturing toolkit *Scrape*, which was designed to assess learners' acquisition of CT when using *Scratch*. Based on their definition of the key CT skills, they supported tagging each learners' coding blocks that represented their records of CT concepts when using *Scratch*. This application visualized how learners' programming actions occurred when compiling coding blocks in

a certain way. The heatmap of the visualization enabled researchers to indicate the frequencies of how learners used specific types of coding blocks that implicate their fluency of programming skills. Wiggins et al. (2015) and Wiggins, Grafsgaard, Boyer, Wiebe, and Lester (2017) introduced the *JavaTutor*, which is an intelligent tutoring system to present a personalized computing education platform that mostly considers students' cognitive and affective states. The platform features data-driven adaptations of tasks tailored to students' contextual needs. This system adopts the concept of multimodal LA driven by following data collection: gestures, interaction logs, physiological responses, and facial expressions.

Although prior studies have increasingly highlighted CT assessments in many OERs, those approaches rarely suggested how to provide learners with personalized environments associated with their CT competency level. Aligned with this challenge, there are further questions that need be explored: (1) How does an OER observe learners' CT developments without interrupting their flow in programming exercises?; (2) What OER features could support learners' CT developments in personalized instruction?; and (3) How does an OER provide personalized learning experiences based on their CT improvement? These questions coherently emphasize how OERs need to be designed to personalize and support learners' meaningful experiences that are associated with CT.

OERs and personalization

Flexible provisions of OER

OER is defined as "the open provision of educational resources, enabled by information and communication technologies, for consultation, use, and adaptation by a community of users for non-commercial purposes." (UNESCO, 2002, p.24). In other words, OER is educational materials distributed in an open format that can be freely accessed by anyone. Researchers have proven the effectiveness of OER in teaching and learning. One representative benefit of OER is its flexibility (Geser, 2007; Hylén, 2006). OER encourages instructors and learners to choose teaching and learning materials based on their preferences and needs. Instructors can access the best possible teaching resources and have flexibility in using those materials (Blomgren, 2018; Hylén, 2006). Learners can select individual units or courses of the topic by reflecting on their needs and level (Yuan, MacNeill, & Kraan, 2008). Thus, the effective uses of OER support the practice of open learning principle which suggests that "learning provision should be flexible so that learners can increasingly choose where, when, what, and how they learn as well as the pace at which they will learn" (Butcher, 2015, p. 6).

Currently, the flexibility of OER enabled researchers and educators to utilize OER to teach CT. MOOC, which is one of the most popular forms of OER, has been used as a tool/method for teaching and learning computational thinking (e.g. Gao, 2016; Mullen et al., 2017). Mullen et al. (2017) designed the High-Performance Computing (HPC) course as a MOOC to assist learner's personalized learning. They designed a self-paced online course that allowed learners to learn HPC focusing on their target system and workplace by adapting MOOC approach in designing an HPC course. Gao (2016) designed a CT course for non-computer majoring students with MOOCs. In the MOOC, this study provided suitable teaching to students from different academic backgrounds with different computing levels. This study adapted a flipped-classroom approach with its MOOC. Non-computer majored

students studied about the topic before class and then interacted with their instructor during the class time. These studies utilized the flexibility of OER to support learner's personalized learning in teaching CT. Although OERs in MOOC environments occurred associated with programming exercises, they were limited in proposing how feasible adoptions can be implemented to purposefully enhance learners' CT skills.

Personalized learning

Recent smart learning environments have proposed their future role as personalized learning support. Both highly-interactive and technology-enriched learning environment contexts enable educators to seek various ways to execute personalized learning. Although personalized learning has received a considerable amount of attention from educators and researchers (Butoianu, Vidal, Verbert, Duval, & Broisin, 2010; Song, Wong, & Looi, 2012), there has been no widely accepted definition of it. The first use of the term "personalized" could be seen in the Personalized System of Instruction (PSI) proposed by Fred Keller and his colleagues for college learners in 1962 (Keefe, 2007). The U.S. Department of Education (2017) recently defined personalized learning as "instruction in which the pace of learning and the instructional approach are optimized for the needs of each learner" (p. 9). Bill and Melinda Gates Foundation (2014) explained that personalized learning has four characteristics: competency-based education, flexible learning environments, personal learning plans, and learner profiles. The purpose of personalized learning is to "build a 'profile' of each student's strength, weakness, and pace of learning" (Educause Learning Initiative (ELI), 2015).

The rapid development of information communication technology (ICT) has made personalized learning feasible (Dawson, Heathcote, & Poole, 2010). Specifically, the developments of both intelligent web-based learning systems (Chen, 2008) and e-learning systems (Chen, Lee, & Chen, 2005) have contributed to supporting learners' personalized learning in online learning environments. Brusilovsky (1999) recommended that personalized learning systems have mechanisms of adaptive learning based on the following rationales: a learning system should offer an optimal learning path for individual learners with different knowledge and learning abilities, but traditional web-based learning systems provide learning materials disregarding the learners' needs. A recent systematic review study on technology-enhanced personalized learning found that personalized data sources including learners' preferences, learner profiles, learning achievements, and learning logs were the main parameters to support personalized learning (Xie, Chu, Hwang, & Wang, 2019). This mechanism suggested how personalized learning can be integrated and implemented in the context of OERs.

Associated with learners' CT, designing personalized online learning environments is necessary to support learners' meaningful practices when learning programming. Previous studies confirmed that novice learners are likely to be overwhelmed by highly complex structures of programming codes (Eckerdal, 2009). Reading and writing programming codes demands learners' prerequisite identifications of both programming functions and grammars to be implemented. Unless learners attain mindful awareness of programming logic, they could not clearly understand basic concepts and their relationships among codes (Chookaew, Panjaburee, Wanichsan, & Laosinchai, 2014). In addition, high variations of learners' familiarity and prior knowledge level in

computer programming can also be a determinant that personalized learning modules should be delivered.

Several case studies demonstrated the implementation examples of personalized online learning environments. Chookaew et al. (2014) proposed a personalized e-learning environment based on learning problems, learning styles, and performance levels. In their personalized e-learning environment, the concept-effect model (Panjaburee, Hwang, Triampo, & Shih, 2010) was used to identify learners' learning problems. The Index of Learning Style (ILS) (Felder & Solomon, 1988) questionnaire was conducted to identify each student's learning style. Based on the index score, they categorized learners' groups (high-, middle-, and low-performance groups) depending on the correctness of the learners' responses (Yan, Hara, Nakano, Kazuma, & He, 2016). However, precedent research on personalized learning was still limited in explaining how learners' behavior metrics can be processed, analyzed, and represented in correspondence with learners' diverse needs and learning paths in programming contexts.

Role of LA to support personalized learning

Another area that has impacted personalized learning is LA. Because of their capabilities to accommodate individual student needs, adaptive learning systems are essential in bringing personalized learning. In his discussion of future for adaptive learning systems, Essa (2016) suggests seven characteristics:

- ***Cost-effective*** *to build, maintain, and support;*
- ***Accurate*** *in its assessment of learner characteristics and learner knowledge state;*
- ***Efficient*** *in carrying out decisions and recommendations, such as identifying optimal instructional resources and activities for each learner at each moment in time;*
- *Able to* ***scale*** *to support hundreds of thousands, if not millions, for simultaneous users;*
- ***Flexible*** *in being able to integrate with enterprise systems based on open standards;*
- ***Generalizable*** *to domains beyond STEM disciplines*
- *Able to support* ***transparent*** *open learner models to encourage learners to take greater control and responsibility for their own learning* (p. 1).

LA plays a crucial role in bringing these requirements into reality. With the advancement of machine learning techniques and computing power to process big data, LA enabled researchers and educators to develop and create more effective adaptive learning systems.

Since the emerging of LA, researchers have been utilizing LA for their studies on designing and developing adaptive learning systems. First, researchers utilized LA to evaluate the effectiveness of adaptive learning systems. For instance, Liu et al. (2017) have utilized LA to understand pharmacy learners' usage patterns for learning adaptive system for their chemistry and biology modules and discovered that affective factor such as motivation can be indicative of student success, and further highlighted the importance of alignment between components within the system as well as the role of visualization of data using LA in understanding user behavior. Similarly, Mojarad, Essa, Mojarad, and Baker (2018) used LA techniques to evaluate the effectiveness of an

adaptive learning system called ALEKS (Assessment and Learning in Knowledge Spaces). They used the propensity score matching technique to conduct quasi-experiment to remove bias and discovered that adaptive learning system is, in fact, helpful in terms of learners' pass/fail rate. As illustrated by the examples, LA enables identifying and evaluating factors that impact student learning as well as assessing the overall impact of adaptive learning systems.

Second, researchers and educators have been implementing LA more directly by designing and developing adaptive learning system powered by LA. LA can benefit learners in a number of ways. Adapting from Ifenthaler and Widanapathirana's (2014) categorization, Schumacher and Ifenthaler (2018), lists three benefit types of LA as *summative, real-time, and predictive*. Summative benefits refer to understanding learners in multiple angles such as learning habits, learning paths, and learning goals. Real-time benefits refer to LA's capabilities to instantly assess learners and intervene in learners' behaviors through feedbacks. Lastly, predictive benefits refer to LA's capabilities in predicting learner outcomes and providing recommendations based on those predictions to increase learner success rate.

Because of these benefits, LA can help building more effective adaptive learning systems. For instance, with their integrated course-adapted student LA framework, Aljohani et al. (2019) created a prototype of a student-centered analytical dashboard that provides integrated information on learners' Blackboard usage. The tool would provide integrated feedback in three forms: statistical feedback, textual feedback, and visual feedback. The results suggest that providing personalized feedback for enhanced student engagement is crucial. On the other hand, Gong and Liu (2019) developed and implemented a personalized learning system for a blended learning environment that provided different types of interventions based on LA. The intervention would be a combination of individual and group interventions, online and offline interventions, and systematic and human interventions. The analyses on achievement, online learning behavior, and self-measuring learning engagement suggested that interventions based on LA could help improve learners' performance and engagement, especially for risky learners. Vesin, Mangaroska, and Giannakos (2018) developed a programming tutoring system called ProTuS that consist of following features: interactive visualizations of learner activities, personalization options that can provide personalized recommendations of learning resources for learners and customization options to tailor user interface appearance. In their usability evaluation of LA component of ProTuS, learners found ProTuS to be useful, particularly the interactive visualization features that can potentially increase student engagement. However, they also pointed out that interactivity could increase the complexity of the system. These examples illustrate how LA allowed researchers to develop adaptive learning systems that are more effective and efficient in terms of learner success.

Design framework

Through a literature review of studies on relevant topics (i.e., teaching and assessing CT, flexible provisions of OER, and personalized design), we conceptualized a personalized OER framework that integrates LA techniques. We distilled key CT competencies and mapped out how we can measure CT competencies with multiple learning-analytics-driven measures. This framework was specifically designed for OER contexts

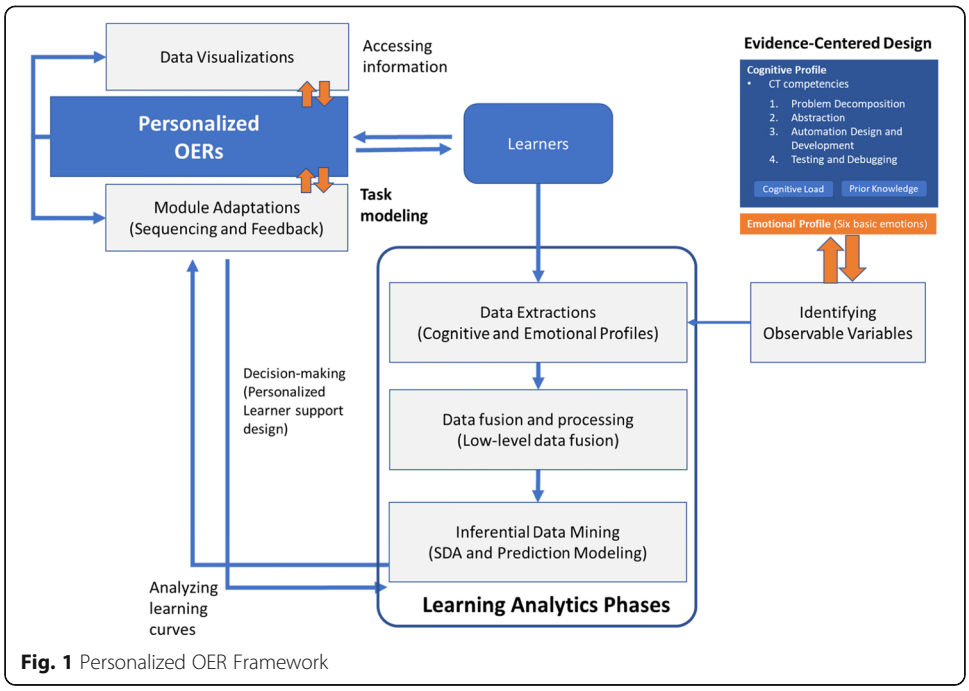

Fig. 1 Personalized OER Framework

in online learning environments. This design framework comprises steps to collect, mine, and analyze learners' interaction data in an OER system. Following sections explain how learners' online profiles are defined, sampled, and processed in the framework. Figure 1 portrays the proposed personalized OER design framework.

Data extractions

In this stage, an OER system specifies the extent of data extractions for collecting learners' profiles when computer programming. This framework addresses two types of learners' profiles: cognitive profiles and emotional profiles. The cognitive profiles refer to their mental processing—relating to their CT competencies and related attributes (cognitive load and prior knowledge), the emotional profiles address learners' affective flow. Table 1 is the example matrix that mapped out learners' profiles, key attributes and observable variables in the proposed framework.

Cognitive profile

Identifying learners' cognitive profiles requires systematic implementations of LA measures. Hence, this framework adopted the evidence-centered design model (ECD) (Mislevy & Haertel, 2006). Following the ECD model, we distilled target CT competencies relating to programming exercises.

CT competencies

– **Problem decomposition:** If a problem statement in an online learning task is presented within individual programming exercises, learners are asked to decompose a target problem to multiple smaller chunks of programming tasks. Learners are requested to identify the essential features of the problem and then to

Table 1 Example data extraction and analytics matrix

Profile types	Attribute types	Observable variables	Data types for collecting evidence	Example data processing algorithms
Cognitive	Problem decomposition	Identifying a logic of initial programming codes	• Logic identification ◦ Programming code tagging (v) ◦ Verbal annotation records on each code (v) ◦The amounts of created codes (n) ◦Document-similarity score (n)	• Verbal reflection records: Topic modeling (Latent Dirichlet allocation: LDA) • Normalization of keywords: *bag-of-words* • Document-similarity analysis: *TF-IDF* (cosine and Jaccard indices) • Extractions of key coding patterns: SPM (GSP and SPADE algorithms) • Score normalization for data fusion: *CombSum* and *ComMNZ*
	Abstraction	Required programming code formulation Fluent representations of code formulations	• Code formulation ◦ Programming code tagging (v) ◦ Total number of codes (n) ◦ Frequency of key coding patterns (n) • Fluent representation ◦ Total number of new codes (n) ◦ Procedural code similarity	
	Automation design and development	Developing a required loop system Building an efficient loop structure	• Loop-system development ◦ Frequency of key coding patterns (n) – specific to the loop system. ◦ Correct expressions of loop codes (v) • Loop-design efficiency ◦ The number of codes (n) ◦ Required implementations of loop codes (v) ◦ Total time to modify codes on an application (n)	
	Testing and Debugging	Effective modifications of existing code structures Identifying and correcting error codes	• Effective modification of existing code structures ◦ Total number of runs (n) ◦ Frequency of changes in existing codes (n) ◦ Total time to modify codes on an application (n) ◦ Verbal annotation records on each code (v).	

Table 1 Example data extraction and analytics matrix *(Continued)*

Profile types	Attribute types	Observable variables	Data types for collecting evidence	Example data processing algorithms
			• Identifying and correcting error codes ◦ Frequency of correct and erroneous runs (n) ◦ Total number (time) of failures on runs (n)	
	Cognitive load	Mental exertions in programming	• Mental effort estimation ◦ Code development efficiency (n)	• Efficiency estimation (the number of codes / (Total time for code developments) with z-score normalization
	Prior knowledge	Learners' declarative and procedural knowledge as to programming languages	• Code formulations by using pseudo-codes ◦ Verbal annotation records (v) ◦ Frequency of erroneous implementations (n). ◦ Attemps to change values on pseudo codes.	
Emotional	Plutchik's six basic emotion (e.g., sadness, surprise, trust, and joy)	Articulations of emotional expressions Behaviors that represent positive and negative emotional states • Engaged states to participate in tasks (positive). • Quitting a system (negative)	• Articulations of emotional expressions ◦ Real-time chatting logs (v) • Engaged states to participate in tasks ◦ Pause time (n) ◦ Frequency of starting a new task (n) • Quitting a system ◦ Frequency of attempts to quit a system (n)	• Emotional state classifier (probability estimation; i.e. emotional predictor designed by recurrent neural network) • Topic modeling (*LDA*)

(n) Numerical data, *(c)* Categorical data, *(v)* Verbal data

propose how individual solutions for smaller problems are interconnected with performing computing algorithms.
- **Abstraction:** Abstraction explains learners' pattern recognitions and developments when editing programming codes. This competency stands for learners' capabilities of conceptualizing programming code designs.
- **Automation design and development:** After identification of a problem statement, learners are required to conduct symbolic representations of their programming codes to simulate automated algorithms.
- **Testing and debugging:** Testing indicates the executions of initial programming artifacts. Testing aims to examine unexpected errors during the artifact's executions. Based on the results of testing, debugging is conducted. Debugging is the refinements of previous programming implementations until the data processing of the artifact successfully runs. Testing focuses on learners' act of the evaluation, and debugging is defined as learners' attempts to modify their programming structures to be advanced.

Cognitive load and prior knowledge In addition to key CT competencies, other cognitive attributes can also be considered for data collection: cognitive load and prior knowledge (skills). Whereas cognitive load indicates students' either mental exertions or extraneous loads in their computerized programming tasks, students' prior knowledge or skills on computer programming estimates how students already become familiar with their programming environments.

Emotional profile

Emotional states in this framework refer to learners' engagement and motivation when using OERs. Analyzing learners' emotional states is to indicate how learners' task perseverance changes during a series of programming exercise modules. Specifically, an increasing level of learners' interest indicates their enhancement of mental exertions during their programming exercises, and frustration is indicative of learners' decreasing motivation. For example, Vail, Grafsgaard, Boyer, Wiebe, and Lester (2016) evidenced how multimodal LA can predict their emotions in an intelligent tutor-based OER. For emotion classification, this study used multiple multimodal data features: facial expression, gesture and physical distance from the workstation. Moreover, using a bag-of-word and latent-semantic indexing models, Colneriĉ and Demsar (2018) proposed the emotion recognition algorithm that detects key emotional states from verbal utterances.

Data fusion and processing

In addition to collecting profile information, it is necessary to integrate different types of data as data fusion. This framework envisions the low-level data fusion that combines all raw data, which has similar epistemological features. In comparison to naïve data fusions using unsupervised dimensionality-reduction techniques, we already define major cognitive and emotional profiles and associated data features. Hence, we do not run data explorations to confirm a collection of similar data features. Instead, for data fusion, this step normalizes all raw data from multiple data channels with the same

profile category. Benchmarked by the idea of the JDL data-fusion classification model (Steinberg, Bowman, & White, 1999), this framework stores the fused data in database management systems that can be retrieved for further inferential data mining, as well as module adaptations. Table 1 shows how the proposed framework includes multiple data channels that represents observable variables associated target attributes of the competency assessment.

Inferential data mining

Inferential data mining investigates how collected learners' profile data is interpreted to provide future learning modules. This step needs automatic decision-making based on learners' profile data (i.e., cognitive and emotional profiles) depending on the analytics' purposes. First, sequential data analytics (SDA) (e.g., sequential analysis and sequential pattern mining) can be implemented to measure how students' programming actions resemble those of experts (Moon & Liu, 2019). As an example, sequential pattern mining (SPM) is useful to indicate how closely students' procedural behaviors for programming are similar to those from experts. This result may indicate students' skill acquisition level. Evidence supports that prior constructivist pedagogical interventions aimed to capture how expert-like scientific understanding emerges by students' in-process actions (Sengupta et al., 2013). Second, machine-learning-based prediction modeling, such as *recurrent neural network* (RNN: Mao et al., 2019) and *Bayesian knowledge tracing* (BKT: Jiang, Ye, & Zhang, 2018), enables researchers to estimate how likely individuals improve their competencies. Whereas SDA aims to disclose students' action patterns and consequences, prediction modeling approaches aim at their future improvements of target competencies.

Results from those approaches are evidence of how to deliver personalization to individual learners. To run all approaches aforementioned for determining adaptation types, the accurate estimation of the threshold-based model is important. To set the accurate threshold range for the competency states for automatic decision-making (i.e., high. Intermediate, and low), supervised dimensionality-reduction techniques can be used—such as linear discriminant analysis (LDA).

Module adaptations and data visualizations

Task modeling

Task modeling refers to the design and development of a collection of learning activities that promote learners' performance. We propose task modeling for adaptive presentations of tasks based on learners' improvements in target competencies. In this framework, we contextualize our task modeling approach guided by Corbalan, Kester, and van Merriënboer (2006). The original model consists of three main components: *characteristics*, *personalization*, and *learning-task database*. First, in the component characteristics, learner profile information (cognitive and emotional profile) and task characteristics are documented through learners' activity artifacts. Learners' artifacts are automatically logged to compute learners' mental effort level (cognitive load) and task performances (competencies). Second, as personalization, based on the activity logs from artifacts, the difficulty of a task, as well as learning support types change. Third,

all the activity logs are archived in the learning-task database to determine the level of personalization—indicating which tasks should be presented.

Task module adaptation and visualization

Derived from the threshold-based model, which specifies how task modules are presented, this framework envisions two personalization approaches: sequencing and feedback. First, the module adaptations can be executed by sequencing design (e.g., topical sequencing, whole-to-part sequencing, and part-to-whole sequencing) (Reigeluth, 1999). Either individual sequencing or its combinations can be taken into consideration depending on learners' CT improvement during programming exercises. For example, in learners' modeling and simulation tasks, if their emotional profile is lower than the threshold range, different contexts or scenarios of modeling tasks would be displayed for individual learners. Second, in-situ feedback during learners' practices is given. Delivering just-in-time (JIT) feedback indicating rules and grammar is considered to promote learners' modifications of programming codes effectively. For example, using intelligent tutors gives JIT cues that correct students' errors in their exercises. If the results of students' cognitive profile data are lower than the threshold range, more hints and clues can be delivered for the target individuals.

In addition to module adaptations, data visualizations provide information on learners' skill progression during programming exercises. As an assessment for learning (Black, Harrison, & Lee, 2003), this step aims to enhance students' CT by promoting their self-monitoring of learning outcomes. For instance, data visualizations can include interactive diagrams that can be accessed by learners to obtain information on which knowledge and skills should be improved.

Discussion

The proposed framework in this study provides a guideline for using LA as one of the ways to support personalized learning in OER. Prior research has mainly focused on producing and sharing OER (Wiley, Bliss, & McEwen, 2014). Although it has been explained that OERs have the potentials to support personalized learning (Yuan et al., 2008), specific guidelines or frameworks that inform the implementation are lacking. This framework can be a starting point for discussions of such guidelines and frameworks.

In addition, further LA designs and implementations could stem from this conceptualization. Specifically, research needs to explore how integration of LA in OER systems can better predict learners' competencies in programming which can contribute to learners' computational thinking abilities. Future empirical studies would help educational researchers in identifying which types of LA designs and implementations better support the personalized OER. For example, emerging multimodal LA techniques are an approach to deliver comprehensive information on learners' meaningful competency progressions (Andrade, Delandshere, & Danish, 2016). Multimodal LA refers to the data technique which combines computer-interaction logs and real-world signals from learners. A personalized OER combined with multimodal LA techniques would consider observing learners' multi-channel data to indicate and recommend optimized learning modules tailored to learners' competency level.

In terms of content designs in OERs, future research needs to consider the integration of instructional design (ID) elements for personalization. First, discussions on contextualized instruction designs for teaching CT in OERs are essential. Previous research on MOOCs have focused on accessing the platform's benefits, such as free and open access to learning materials, self-paced learning with time and place-independent online courses, as well as freedom in the selection of learning materials (Gao, 2016; Mullen et al., 2017). As shown in Sunar, Abdullah, White, and Davis's (2015) review, there have been efforts to propose and implement personalization in MOOCs. For example, researchers have introduced designing learners' personalized repositories in MOOCs, which used as book collections and Web 2.0 tools with bibliographical indices that enable learners to better search for their preferred learning materials (Cohen, Reisman, & Sperling, 2015; Mikroyannidis & Connolly, 2015). However, existing OER designs and implementations are still limited in enhancing learners' CT during web-based programming exercises. Future design-based research is necessary to explore how OERs can be tailored to teaching and learning CT. Moreover, while the present study stated the role of content-sequencing and feedback designs in the OER framework, we could not thoroughly discuss how those design implementations can be better coordinated in teaching CT. There is still a gap in coordinating between data-mining approaches from LA and personalized ID implementations during programming exercises. For example, novice learners in programming exercises are likely to experience high cognitive load as they interpret and create programming codes, but implications on how to reduce novice learners' cognitive load.

The purpose of the study was to propose a conceptual framework that illustrates how to support learners' CT developments through personalized designs of OERs. With this in mind, study aimed to provide a groundwork for designing personalized OERS that support learners in developing CT competencies. In order to achieve the goal, several foundational literatures have been reviewed. First, through the review of literatures on computational thinking, we have identified essential elements of computational thinking, introduced computing platforms and learning environments that are designed and built to support and enhance computational thinking, and explored cases of formative assessments and adaptations of teaching computational thinking. Second, we also reviewed literatures on OERs, personalized learning, and learning analytics and their implications in enhancing CT for learners.

Based on the conceptualizations based on these reviews, we formulated a personalized OER framework and illustrated how such a framework is capable of observing and supporting learners' CT. Our OER design framework consists of data extractions, data fusion and processing, inferential data mining, and module adaptations and data visualizations. Moreover, as a way to observe learners' CT skills in OERs, this study suggested the execution of LA to assess the evidence of learners' CT capabilities. We proposed that this conceptualization of a personalized OER framework requires further empirical studies to confirm how personalized OER designs can improve in identifying and supporting learners' CT competencies. While we acknowledge the need for our framework to be validated by an empirical study, we believe that the study can play a role in providing a groundwork for designing and developing a theory-informed personalized OER that can improve learners' CT competencies.

Acknowledgements
Not applicable.

Authors' contributions
All authors read and approved the final manuscript. JM drafted the overall manuscripts. Specifically, he designed and developed a conceptual framework for this study. JD supported drafting this manuscript and then participated in the review of OER environments. DL helped in developing the concept of personalization and contributed to the design of a conceptual framework in this study. GWC participated in the reviews and designs of learning analytics in this paper.

Authors' information
Jewoong Moon is a Ph.D. Candidate in the Department of Educational Psychology and Learning Systems at Florida State University. His research interests include digital game-based learning, inclusive e-learning design, learning analytics, and educational data mining.
Jaewoo Do is an associate research fellow at Korean Educational Development Institute. His research interests include synchronous online course design, design thinking, equity in education and qualitative research.
Daeyeoul Lee is a Ph.D. Candidate at the Department of Curriculum and Instruction at Purdue University. His main research interest is self-regulated learning, motivation, online learning, and MOOCs.
Gi Woong Choi is an assistant professor of human-computer interaction at State University of New York at Oswego. His current research interests include educational affordances of emerging technologies, mobile learning, problem-solving, AR/VR, and AI in education.

Funding
This research was not supported by any funding institutions.

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Educational Psychology and Learning Systems, Florida State University, Tallahassee, United States. [2]Korean Educational Development Institute(KEDI), 7, Gyohak-ro, Deoksan-eup, Jincheon-gun, Chungcheongbuk-do, Republic of Korea. [3]Department of Curriculum and Instruction, Purdue University, West Lafayette, United States. [4]HCI Program, Department of Computer Science, State University of New York at Oswego, Oswego, United States.

Received: 13 October 2019 Accepted: 5 December 2019
Published online: 13 February 2020

References

Abelson, H., Wolber, D., Morelli, R., Gray, J., & Uche, C. (2012). Teaching with app inventor for android. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (p. 681). New York: ACM.
Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832–835.
Aljohani, N. R., Daud, A., Abbasi, R. A., Alowibdi, J. S., Basheri, M., & Aslam, M. A. (2019). An integrated framework for course adapted student learning analytics dashboard. *Computers in Human Behavior, 92*, 679–690. https://doi.org/10.1016/j.chb.2018.03.035.
Andrade, A., Delandshere, G., & Danish, J. A. (2016). Using multimodal learning analytics to model student behavior: A systematic analysis of epistemological framing. *Journal of Learning Analytics, 3*(2), 282–306.
Author (2019). Blinded for peer review.
Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology, 13*(1), 20–29.
Bill & Melinda Gates Foundation, A. P, Eli and Edythe Broad Foundation, CEE Trust, Christensen Institute for Disruptive Innovation, Charter School Growth Fund, . . . Silicon Schools. (2014). Personalized learning: A working definition. Education week. Retrieved from https://www.edweek.org/ew/collections/personalized-learning-special-report-2014/a-working-definition.html
Black, P., Harrison, C., & Lee, C. (2003). *Assessment for learning: Putting it into practice*. UK: McGraw-Hill Education.
Blikstein, P. (2011). Using learning analytics to assess students' behavior in open-ended programming tasks. In *Proceedings of the 1st international conference on learning analytics and knowledge* (pp. 110–116). New York: ACM.
Bonk, C. J., Lee, M. M., Reeves, T. C., & Reynolds, T. H. (2015). *MOOCs and open education around the world*. New York: Routledge.
Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Paper presented at the proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*.
Brusilovsky, P. (1999). Adaptive and intelligent technologies for web-based eduction. *Ki, 13*(4), 19–25.
Butoianu, V., Vidal, P., Verbert, K., Duval, E., & Broisin, J. (2010). User context and personalized learning: A federation of contextualized attention metadata. *Journal of Universal Computer Science, 16*(16), 2252–2271.
Chen, C.-M. (2008). Intelligent web-based learning system with personalized learning path guidance. *Computers & Education, 51*(2), 787–814.
Chen, C.-M., Lee, H.-M., & Chen, Y.-H. (2005). Personalized e-learning system using item response theory. *Computers & Education, 44*(3), 237–255.

Chookaew, S., Panjaburee, P., Wanichsan, D., & Laosinchai, P. (2014). A personalized e-learning environment to promote student's conceptual learning on basic computer programming. *Procedia-Social and Behavioral Sciences, 116*, 815–819.
Cohen, A., Reisman, S., & Sperling, B. B. (2015). Personal spaces in public repositories as a facilitator for open educational resource usage. *The International Review of Research in Open and Distributed Learning, 16*(4), 156–176.
Colnerič, N., & Demsar, J. (2018). Emotion recognition on twitter: Comparative study and training a unison model. *IEEE Transactions on Affective Computing*. https://doi.org/10.1109/TAFFC.2018.2807817.
Corbalan, G., Kester, L., & van Merriënboer, J. J. (2006). Towards a personalized task selection model with shared instructional control. *Instructional Science, 34*(5), 399–422.
Dawson, S., Heathcote, L., & Poole, G. (2010). Harnessing ICT potential: The adoption and analysis of ICT systems for enhancing the student learning experience. *International Journal of Educational Management, 24*(2), 116–128.
Eckerdal, A. (2009). *Novice programming students' learning of concepts and practise*. Acta Universitatis Upsaliensis Retrieved from https://pdfs.semanticscholar.org/bccc/7fb6b4080d8c01aedf2b0f701989f7b9841d.pdf.
Educause Learning Initiative (ELI) (2015). 7 things you should know about personalized learning. Retrieved from https://library.educause.edu/resources/2015/9/7-things-you-should-know-about-personalized-learning
Gao, Q. (2016). Computational thinking and MOOC-oriented computer courses teaching mode for non-computer major. In *3d international conference on applied social science research (ICASSR 2015)*. Atlantis Press.
Gong, L., & Liu, Y. (2019). Design and application of intervention model based on learning analytics under blended learning environment. In *Proceedings of the 2019 7th international conference on information and education technology - ICIET 2019* (pp. 225–229). https://doi.org/10.1145/3323771.3323825.
Griffin, P., & Care, E. (2014). *Assessment and teaching of 21st century skills: Methods and approach*. Dordrecht: Springer.
Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12. In *Paper presented at the proceedings of the 2014 conference on innovation & technology in computer science education*.
Gunathilaka, T. A. U., Fernando, M. S. D., & Pasqual, H. (2017). Identification of the learning behavior of the students for education personalization. In *2017 international conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)* (pp. 364–370). IEEE.
Harel, I. E., & Papert, S. E. (1991). *Constructionism*. Norwood: Ablex Publishing.
Hoppe H., & Werneburg S. (2019). *Computational Thinking—More Than a Variant of Scientific Inquiry!. In: Kong SC., Abelson H. (eds) Computational Thinking Education*. Singapore: Springer.
Hosseini, R. (2017). Assessing programming behaviors through evidence-centered design. Analytics for Learning (A4L). Retrieved from https://a4li.sri.com/archive/papers/Hosseini_2017_Problem_Solving.pdf.
Ifenthaler, D., & Widanapathirana, C. (2014). Development and validation of a learning analytics framework: Two case studies using support vector machines. *Technology, Knowledge and Learning, 19*(1–2), 221–240. https://doi.org/10.1007/s10758-014-9226-4.
Informatics Europe. (2017). Class'Code wins 2017 best practices in education award. Retrieved from https://www.informatics-europe.org/news/395-class-code.html
Jiang, B., Ye, Y., & Zhang, H. (2018). Knowledge tracing within single programming exercise using process data. In *Proceedings of the 26th international conference on computers in education* (pp. 89–94).
Kafai, Y. B. (2006). Playing and making games for learning: Instructionist and constructionist perspectives for game studies. *Games and Culture, 1*(1), 36–40.
Kafai, Y. B., & Burke, Q. (2013). The social turn in K-12 programming: moving from computational thinking to computational participation. In *Paper presented at the proceeding of the 44th ACM technical symposium on computer science education*.
Keefe, J. W. (2007). What is personalization? *Phi Delta Kappan, 89*(3), 217–223.
Koh, K. H., Basawapatna, A., Nickerson, H., & Repenning, A. (2014). Real time assessment of computational thinking. In *2014 IEEE symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 49–52). Melbourne: IEEE.
Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., et al. (2011). Computational thinking for youth in practice. *Acm Inroads, 2*(1), 32–37.
Lee, T. Y., Mauriello, M. L., Ingraham, J., Sopan, A., Ahn, J., & Bederson, B. B. (2012). CTArcade: learning computational thinking while training virtual characters through game play. In *Paper presented at the CHI'12 extended abstracts on human factors in computing systems*.
Liu, M., Kang, J., Zou, W., Lee, H., Pan, Z., & Corliss, S. (2017). Using data to understand how to better design adaptive learning. *Technology, Knowledge and Learning, 22*(3), 271–298. https://doi.org/10.1007/s10758-017-9326-z.
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*, 51–61.
Mao, Y., Zhi, R., Khoshnevisan, F., Price, T. W., Barnes, T., & Chi, M. (2019). One minute is enough: Early prediction of student success and event-level difficulty during a novice programming task. In *Proceedings of the 12th international conference on educational data mining* (pp. 119–128).
Mikroyannidis, A., & Connolly, T. (2015). Case study 3: Exploring open educational resources for informal learning. In *Responsive open learning environments* (pp. 135–158). Cham: Springer.
Mislevy, R. J., & Haertel, G. D. (2006). Implications of evidence-centered design for educational testing. *Educational Measurement: Issues and Practice, 25*(4), 6–20.
Mojarad, S., Essa, A., Mojarad, S., & Baker, R. S. (2018). Studying adaptive learning efficacy using propensity score matching. In *Companion proceedings 8th international conference on learning analytics & knowledge* (pp. 1–8). Sydney: ACM.
Mullen, J., Byun, C., Gadepally, V., Samsi, S., Reuther, A., & Kepner, J. (2017). Learning by doing, high performance computing education in the MOOC era. *Journal of Parallel and Distributed Computing, 105*, 105–115.
Ota, G., Morimoto, Y., & Kato, H. (2016). Ninja code village for scratch: Function samples/function analyser and automatic assessment of computational thinking concepts. In *Paper presented at the 2016 IEEE symposium on visual languages and human-centric computing (VL/HCC)*.
Panjaburee, P., Hwang, G.-J., Triampo, W., & Shih, B.-Y. (2010). A multi-expert approach for developing testing and diagnostic systems based on the concept-effect model. *Computers & Education, 55*(2), 527–540.
Papert, S. (1999). What is logo? Who needs it. Logo philosophy and implementation. Retrieved from http://www.microworlds.com/support/logo-philosophy-papert.html.
Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism, 36*(2), 1–11.

Reigeluth, C. M. (1999). The elaboration theory: Guidance for scope and sequence decisions. *Instructional design theories and models: A new paradigm of instructional theory, 2*, 425–453.
Repenning, A., & Sumner, T. (1995). Agentsheets: A medium for creating domain-oriented visual languages. *Computer, 28*(3), 17–25.
Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67.
Saines, G., Erickson, S., & Winter, N. (2013). *Codecombat*. Silicon Valley: CodeCombat.
Schumacher, C., & Ifenthaler, D. (2018). Features students really expect from learning analytics. *Computers in Human Behavior, 78*, 397–407. https://doi.org/10.1016/j.chb.2017.06.030.
Sengupta, P., & Farris, A. V. (2012). Learning kinematics in elementary grades using agent-based computational modeling: A visual programming-based approach. In *Proceedings of the 11th international conference on interaction design and children* (pp. 78–87). Bremen: ACM.
Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351–380.
Settle, A., Franke, B., Hansen, R., Spaltro, F., Jurisson, C., Rennert-May, C., & Wildeman, B. (2012). Infusing computational thinking into the middle-and high-school curriculum. In *Paper presented at the proceedings of the 17th ACM annual conference on innovation and technology in computer science education*.
Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158.
Song, Y., Wong, L.-H., & Looi, C.-K. (2012). Fostering personalized learning in science inquiry supported by mobile technologies. *Educational Technology Research and Development, 60*(4), 679–701.
Steinberg, A. N., Bowman, C. L., & White, F. E. (1999). Revisions to the JDL data fusion model. In *Sensor fusion: Architectures, algorithms, and applications III* (Vol. 3719, pp. 430–441). Orlando: International Society for Optics and Photonics.
Sunar, A. S., Abdullah, N. A., White, S., & Davis, H. C. (2015). Personalisation of MOOCs – The state of the art. In M. Helfert, M. T. Restivo, S. Zvacek, & J. Uhomoibhi (Eds.), *Proceedings of the seventh international conference on computer supported education (CSEDU'15)* (pp. 88–97). Setúbal: Science and Technology Publications.
Sung, W., Ahn, J., & Black, J. B. (2017). Introducing computational thinking to young learners: Practicing computational perspectives through embodiment in mathematics education. *Technology, Knowledge and Learning, 22*(3), 443–463.
U. S. Department of Education (2017). Reimaging the role of technology in education: 2017 national education technology plan update. Retrieved from https://tech.ed.gov/files/2017/01/NETP17.pdf
Vail, A. K., Grafsgaard, J. F., Boyer, K. E., Wiebe, E. N., & Lester, J. C. (2016). Predicting learning from student affective response to tutor questions. In *International conference on intelligent tutoring systems* (pp. 154–164). Cham: Springer.
Vesin, B., Mangaroska, K., & Giannakos, M. (2018). Learning in smart environments: User-centered design and analytics of an adaptive learning system. *Smart Learning Environments, 5*(1), 5–24.
Werneburg, S., Manske, S., & Hoppe, H. U. (2018). ctGameStudio–A game-based learning environment to foster computational thinking. In *26th international conference on computers in education, Philippines*.
Wiggins, J. B., Boyer, K. E., Baikadi, A., Ezen-Can, A., Grafsgaard, J. F., Ha, E. Y., et al. (2015). JavaTutor: an intelligent tutoring system that adapts to cognitive and affective states during computer programming. In *Proceedings of the 46th acm technical symposium on computer science education* (p. 599). Kansas City: ACM.
Wiggins, J. B., Grafsgaard, J. F., Boyer, K. E., Wiebe, E. N., & Lester, J. C. (2017). Do you think you can? The influence of student self-efficacy on the effectiveness of tutorial dialogue for computer science. *International Journal of Artificial Intelligence in Education, 27*(1), 130–153.
Wilensky, U. (1999). NetLogo (and NetLogo user manual). Center for connected learning and computer-based modeling, Northwestern University. http://ccl.northwestern.edu/netlogo.
Wiley, D., Bliss, T.J., McEwen, M. (2014). Open Educational Resources: A Review of the Literature. In: Spector J., Merrill M., Elen J., Bishop M. (eds). Handbook of research on educational communications and technology. New York: Springer.
Wilkerson-Jerde, M., Wagh, A., & Wilensky, U. (2015). Balancing curricular and pedagogical needs in computational construction kits: Lessons from the DeltaTick project. *Science Education, 99*(3), 465–499.
Wilson, A., Hainey, T., & Connolly, T. (2012). Evaluation of computer games developed by primary school children to gauge understanding of programming concepts. In *6th European conference on games-based learning (ECGBL)* (pp. 4–5).
Wilson, C. (2014). Hour of code: We can solve the diversity problem in computer science. *Inroads, 5*(4), 22.
Wing, J. (2011). Research notebook: Computational thinking—What and why? The Link Magazine, Spring. Carnegie Mellon University, Pittsburgh. Retrieved from http://link.cs.cmu.edu/article.php?a=600
Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.
Wohl, B., Porter, B., & Clinch, S. (2015). Teaching computer science to 5–7 year-olds: An initial study with scratch, cubelets and unplugged computing. In *Paper presented at the proceedings of the workshop in primary and secondary computing education*.
Xie, H., Chu, H.-C., Hwang, G.-J., & Wang, C.-C. (2019). Trends and development in technology-enhanced adaptive/personalized learning: A systematic review of journal publications from 2007 to 2017. *Computers & Education*, 103599.
Yan, Y., Hara, K., Nakano, H., Kazuma, T., & He, A. (2016). A method to describe student learning status for personalized computer programming e-learning environment. In *Paper presented at the 2016 IEEE 30th international conference on advanced information networking and applications (AINA)*.
Yuan, L., MacNeill, S., & Kraan, W. G. (2008). *Open educational resources-opportunities and challenges for higher education*. JISC CETIS Retrieved from http://ubir.bolton.ac.uk/290/.

Publisher's Note