Chapter 12
Defining and Assessing Students' Computational Thinking in a Learning by Modeling Environment

Ningyu Zhang and Gautam Biswas

Abstract Learning sciences researchers have hypothesized the connections between STEM learning and computational thinking (CT), and this has been supported by studies that have demonstrated the reciprocal relationships between CT and STEM subjects. However, not much work has been done to establish the fundamental set of CT knowledge and practices that need to be taught to enhance STEM and CT learning in K-12 curricula. Therefore, many important aspects of CT are underrepresented in K-12 classrooms. We believe that CT concepts and practices are not only important for computing education but also play an important role in helping students develop modeling and problem-solving skills in STEM domains. In this chapter, we build on our existing framework (Sengupta et al in Educ Inf Technol 25(1):127–147, 2013) to promote students' synergistic learning of science content and CT practices in middle school classrooms. We discuss the primary STEM and CT concepts and practices that we have introduced through our curricular units into science classrooms and discuss how students can learn these practices in CTSiM, a computer-based learning environment developed in our lab (Basu et al in User Model User-Adapt 27, 2017). We present results from studies in middle school classrooms with CTSiM and show that students have strong learning gains in Science and CT concepts. Our assessments also help characterize and understand students' learning behaviors and their link to learning and practicing STEM and CT concepts.

Keywords Computational thinking · Domain content knowledge · Theoretical framework · Learning by modeling · Assessments

N. Zhang · G. Biswas (✉)
Department of Electrical Engineering and Computer Science, Institute for Software Integrated Systems, Vanderbilt University, 1025 16th Avenue South, Nashville, TN 37212, USA
e-mail: gautam.biswas@vanderbilt.edu

N. Zhang
e-mail: ningyu.zhang@vanderbilt.edu

S.-C. Kong and H. Abelson (eds.), *Computational Thinking Education*,
https://doi.org/10.1007/978-981-13-6528-7_12

12.1 Introduction

Recently, Computational Thinking (CT) has been recognized as a framework for developing computer literacy and computing skills among the K-12 computer science (CS) and STEAM (Science, Technology, Engineering, Arts (and Humanities) and Mathematics) communities (Barr & Stephenson, 2011; Grover & Pea, 2013). Industry and governments also consider CT to be one of the primary drivers of the twenty-first--century workforce (Barr, Harrison, & Conery, 2011). In general, CT encompasses a wide range of abilities and practices, such as problem decomposition and composition, algorithm development, reasoning at multiple levels of abstraction, and creating and reusing modular solutions (Wing, 2006). Researchers and practitioners believe that by drawing from the fundamental skills and practices of CT, students can develop analytical and problem-solving skills and practices. These CT practices go beyond the learning of CS and benefit students' understanding of scientific processes, systems design, and human behaviors (Wing, 2006; NRC, 2010; Barr & Stephenson, 2011; NRC, 2011). In other words, CT can benefit K-12 students' learning in other domains such as mathematics and science as they solve problems while *thinking like a computer scientist* (Wing, 2011; Barr & Stephenson, 2011).

A series of studies have shown that applying CT in STEM domains helps students' learning (Basu & Biswas, 2016; Basu et al., 2017; García-Peñalvo, Reimann, Tuul, Rees, & Jormanainen, 2016; Weintrop et al., 2016a). Additionally, CT skills and practices can transfer to other learning and problem-solving contexts (Basawapatna et al., 2011; Grover, 2015), as CT requires a deep understanding of problem-solving when compared against rote learning (Wing, 2006). Therefore, it provides an essential framework for preparing students for future learning (Bransford, Brown, & Cocking, 2000). In addition, Brennan and Resnick (2012) point out that CT includes the practice of designing artifacts (e.g., building models of STEM phenomena), which helps students develop perspectives of the world around them (e.g., a deeper understanding of the role of vegetation in reducing greenhouse gases). Therefore, CT provides a synergistic framework for learning of computational and science concepts and practices (Sengupta et al., 2013; Basu et al., 2017).

The acknowledgment of these potential benefits of CT has led to the inclusion of CT into the K-12 STEM curricula; for example, the Next Generation Science Standards (NGSS) in the United States includes CT as a core scientific practice (The NGSS Lead States, 2013; Barr & Stephenson, 2011). Researchers have also stressed the urgency of introducing CT into K-12 classrooms. However, there has been insufficient effort to establish a key set of CT concepts and practices across K-12 classrooms, and little effort has been made to include CT concepts and practices into other disciplinary curricula.

In this chapter, we extend our earlier framework (Sengupta et al., 2013; Zhang & Biswas, 2017) by

1. Introducing CT with an emphasis on STEM practices through an open-ended learning environment (OELE); and

2. Evaluating students' synergistic learning of science content and CT practices in middle school classrooms.

In Sect. 12.2 of this chapter, we review existing frameworks for CT learning and assessment. In Sect. 12.3, we introduce our STEM + CT framework, the primary CT skills and practices, and our OELE to promote the synergistic learning of domain content and CT. We also discuss how these learning constructs can be assessed in our STEM + CT framework. Finally, in Sect. 12.4, we use the results from our middle school classroom studies to discuss the design of assessments that help us analyze how students develop and exploit the synergy between the STEM and CT concepts.

12.2 Related Work

Wing's seminal paper describes CT as "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (Wing, 2006, p. 33). Another popular definition states that CT is a problem-solving process that includes characteristics such as problem formulation, data analysis, abstraction, algorithmic automation, solution refinement, and transfer (ISTE & CSTA, 2011). Given the wide scope of CT, there are different ideas of what constitutes CT, and how to integrate CT into the K-12 curricula, especially for STEM topics (NRC, 2010; Brennan & Resnick, 2012). In addition, the close relationships between CT, mathematics, algorithmic thinking, and problem-solving can sometimes veil the core ideas in computation that CT encompasses (García-Peñalvo et al., 2016; Weintrop et al., 2016a). Nevertheless, the various definitions of CT provide the research community and educational stakeholders with many insights into how its various concepts and practices can benefit learning.

Some effort has been made to establish operationalizable frameworks for CT. These frameworks either focus on the constructs that emerge from existing game-based and media-narrative programming environments, or they emphasize STEM concepts and practices. The CT frameworks associated with AgentSheets (Repenning et al., 2000) and Scratch (Brennan & Resnick, 2012) are examples of the former type, and the CT taxonomies proposed by Weintrop et al. (2016a) are examples of the latter type.

The CT framework (Basawapatna et al., 2011) in AgentSheets focuses on the aspects of CT that can be transferred to other learning contexts. In their framework, CT is defined as the recurring program patterns that students acquire while programming games and later reuse in other simulation contexts (Basawapatna et al., 2011). Brennan and Resnick (2012) defined CT as a framework of three components related to programming in Scratch: *what* students should know about programming, *how* to program, and the *sociocultural* aspects of programming. More specifically, computational concepts of their framework refer to the fundamental knowledge of computing, such as how loops and conditionals work in a Scratch program. Computational practices are defined as programming related actions, such as building and debugging. Finally, computational perspectives describe the learner's computation-related world-view (Brennan & Resnick, 2012).

These frameworks operationalize programming-centered CT constructs in existing environments, but they do not provide explicit evidence of how CT is linked to STEM learning. On the other hand, the CT taxonomies proposed in Weintrop et al. (2016a) emphasize the application of CT in STEM classrooms. Weintrop et al. proposed key CT practices that are naturally linked to STEM concepts and practices (NGSS, 2013), including (1) data, (2) modeling and simulation, (3) problem-solving, and (4) systems thinking. In addition to focusing on *what* students learn about CT, these CT practices define *how* students can learn and apply CT, thus providing a theoretical foundation for integrating CT in STEM classrooms.

Some CT frameworks also include the assessment of CT. For example, computational thinking pattern analysis (CTPA) in AgentSheets matches the recurring program patterns in game design and science simulation contexts to evaluate students' understanding of CT (Basawapatna et al., 2011). Additionally, Scratch uses multimodal assessments including project analyses and interviews to assess its main CT constructs (the CT concepts, practices, and perspectives) (Brennan & Resnick, 2012). These CT frameworks provide important information on students' understanding of CT, but they have their shortcomings as well. For example, students' expression of CT is demonstrated at the program level, assuming that the student understands CT if they use a CT pattern. On the other hand, the *used = learned* assumption poses peril because a student writing correct programs nevertheless may not have made the necessary conceptual connections (NRC, 2010). In addition, these analyses of snapshots of completed programs lose the temporal information and the subtlety to understand students' developmental process of CT. As a result, many fundamental aspects of CT have not received sufficient attention especially in the context of block-based programming environments, except for a few successful assessments (e.g., the Fairly Assessment in Alice, Werner, Denner, Campe, & Kawamoto, 2012; Grover & Pea, 2013). Therefore, more detailed, reliable and formative test instruments need to be developed to enrich CT assessments.

12.3 The STEM + CT Framework

Studies have shown that CT and STEM subjects shared a reciprocal relationship. There is evidence in the literature that students improved their understanding of STEM topics when they are studied in a CT framework (e.g., Basu et al., 2017; Sengupta et al., 2013; Weintrop et al., 2016a, b). Similarly, developing CT concepts and practices in a science learning framework provides a context and a perspective for the better understanding of CT. For example, the NRC (2010) report states that CT concepts and practices are best acquired when studying them within domain disciplines. If students were introduced to CT in programming contexts only, they might not develop the skills to apply the generalized CT concepts across disciplines because of the difficulties in the transfer of learning (NRC, 2011). Additionally, learning CT concepts and practices in a STEM modeling and simulation framework provides students with the real-world perspective (Brennan & Resnick, 2012) they may need to develop a good understanding of the STEM and CT concepts in context. Therefore, our CT framework uses an integrated STEM + CT approach to foster students' synergistic learning of domain content knowledge and CT through computational modeling and simulation. In the rest of the section, we introduce our STEM + CT framework, the CTSiM learning environment, and the assessment schemes.

12.3.1 The STEM + CT Framework

Figure 12.1 gives an overview of our STEM + CT framework. Central to the framework is the **Practices** applied across STEM domains that use CT methods. There are four types of practices: *Systems Thinking, Problem-solving, Modeling and Simulation,* and *Data and Inquiry*. The STEM + CT practices are the means to support students' synergistic learning and understanding of **Domain Content** and **CT concepts**. The CT concepts include *variables and assignments, sequential execution of statements, loop structures, conditionals, functions*, and *events*. These CT concepts are fundamental to most programming environments.

We use Computational Thinking using Simulation and Modeling (CTSiM) as the **Learning Environment** in our framework to help students foster the key Domain and CT concepts and practices. CTSiM (Basu et al., 2017; Sengupta et al., 2013) is an open-ended learning environment (OELE) that helps students achieve the syner-

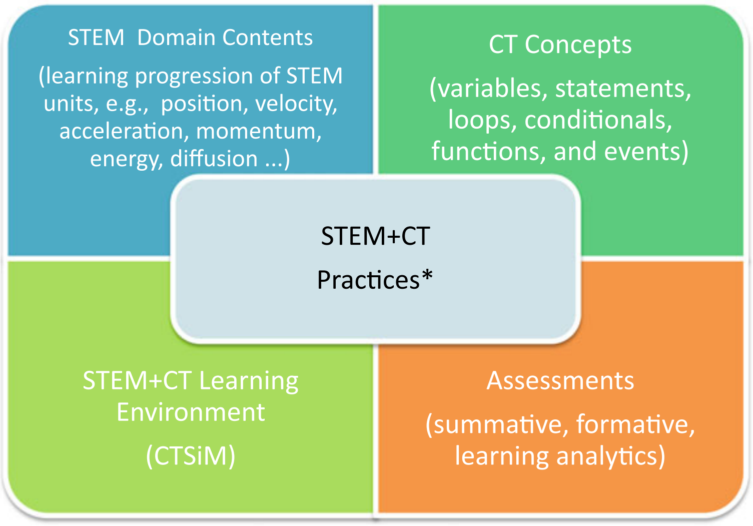

STEM Domain Contents
(learning progression of STEM units, e.g., position, velocity, acceleration, momentum, energy, diffusion ...)
CT Concepts
(variables, statements, loops, conditionals, functions, and events)
STEM+CT
Practices*
STEM+CT Learning Environment
(CTSiM)
Assessments
(summative, formative, learning analytics)

Fig. 12.1 The STEM + CT framework

gistic learning of science knowledge and CT practices using a learning by modeling approach. Additionally, the CTSiM Learning Environment encapsulates **STEM Domain Content** in a progression of units. Finally, the summative and formative **Assessments** evaluate students' improvement in the STEM content and CT concepts, as well as their model-building and problem-solving behaviors and performance.

In this framework, students can synergistically learn STEM and CT concepts and practices as they apply them to develop computational (specifically, simulation) models of the scientific phenomena. Meanwhile, the domain content provides the learning context for how to apply the CT concepts through opportunities for model building and simulation of specific scientific phenomena that students can easily relate to because of their links to real-world phenomena.

Table 12.1 presents the set of STEM + CT practices defined in our framework. It includes four categories of practices commonly applied in STEM domains and 14 practices instantiated in the CT learning environment. Among the four categories, Systems Thinking is the overarching practice as the deep understanding of systems with emergent behaviors is essential for students to prepare for future scientific endeavors (Cheng et al., 2010). However, students can develop misunderstandings when they attempt system-level analyses in an agent-based modeling framework (Wilensky & Resnick, 1999). On the other hand, developing systems understanding

Table 12.1 Related STEM + CT practices

Systems thinking
(ST1) identify the relevant components that make up a system
(ST2) understand component relationships and interactions
(ST3) understand the system at different levels of abstraction, and the behaviors that emerge from these interactions
Modeling and simulation
(MS1) conceptual modeling (formal and abstract thinking)
(MS2) computational modeling (algorithmic thinking and implementation)
(MS3) simulation and execution of models
(MS4) Interpreting and understanding system behaviors (emergence)
Problem-solving
(PS1) dividing into sub-problems
(PS2) modularity and reuse
(PS3) debugging and error-checking
Data and inquiry
(DI1) acquire information from texts
(DI2) set up and run experiments to collect and analyze data
(DI3) organize and display data (table, graphs)
(DI4) interpret plots and other data representations

by applying CT practices provides an approach to help students overcome these problems.

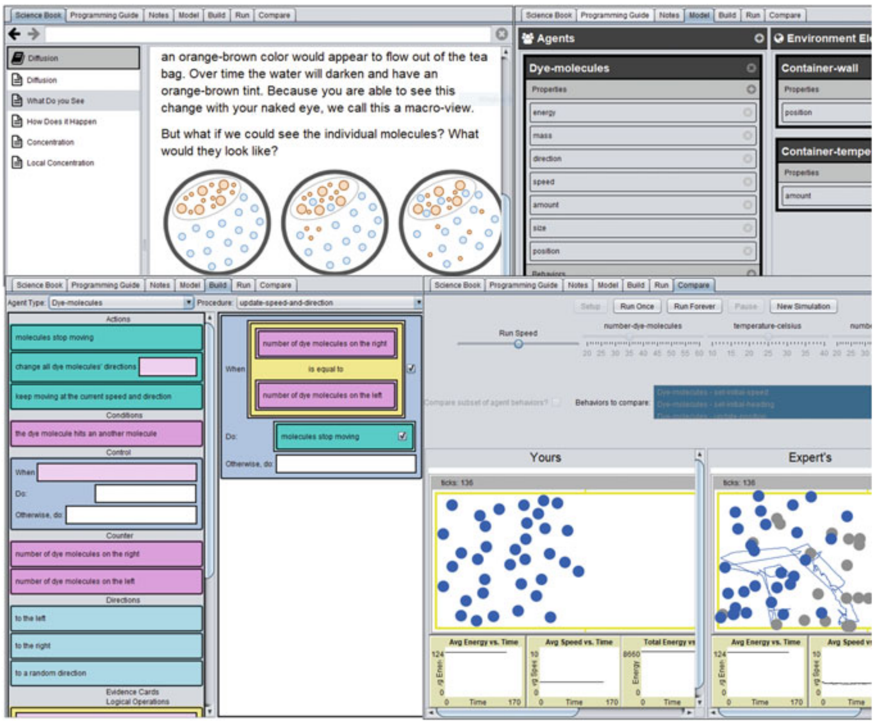
Within our proposed framework, students are expected to study the interactions among agents and between agents and the environment. They can then translate this understanding to identify variables and the relations between relevant variables, and, in this process, study emerging behaviors at different levels. Additional STEM practices, such as *Data collection, Analysis*, and *Inquiry*, *Modeling and Simulation*, and *Problem-solving,* help students develop a deeper understanding of Systems Thinking. Together, these set of fundamental STEM + CT practices serve as the methodology to help learners synergistically acquire the required CT and STEM concepts.

12.3.2 The Learning Environment

In CTSiM, students can model scientific phenomena using a block-structured language that is framed in an agent-based modeling approach (Sengupta et al., 2013). Learners can leverage six types of primary learning tasks in CTSiM. More specifically, students can (1) acquire information of the domain content and CT-related programming concepts from the built-in hypertext resource libraries; (2) represent their system in terms of agents, their properties, and their behaviors using an abstract conceptual model representation; (3) construct runnable computational (i.e., simulation) models that define the agents' behaviors and interactions in the environment; (4) execute the computational models as NetLogo (Wilensky, 1999) simulations to examine the agents' behaviors; (5) compare the behaviors of their computational models to the behaviors generated by an expert model; and (6) conduct science inquiry to understand the relationship between components in a system and the behaviors generated by the components. These learning activities are closely related to and can help students' foster STEM + CT practices. For example, reading the hypertext libraries supports the *Data and Inquiry* practice of acquiring information from texts. The *Data and Inquiry* practice also involves interpreting plots and other data representations. Similarly, constructing the conceptual and computational models directly relates to *Modeling and Simulation* practices. In addition, running the computational models and comparing their results to ones generated from an expert simulation support students *Problem-solving* practices, such as debugging and error-checking. We will discuss in more detail the affordances of the environment in supporting STEM + CT concepts and practices in the introduction to the *Assessment* framework.

The units in CTSiM are developed to complement the middle school STEM curricula with modeling and simulation activities. CTSiM offers a number of units, forming a STEM-learning progression from modeling motion of objects using turtle geometry to investigating complex, emergent systems, such as diffusion of particles in a liquid medium. The curricular modules of CTSiM include (1) turtle geometry, (2) kinematics, (3) mechanics, (4) collision and momentum, (5) particle collision and energy, (6) fish tanks and the carbon cycle, and (7) role of bacteria and the nitrogen cycle in fish tanks (Basu et al., 2017; Sengupta et al., 2013; Zhang, Biswas, & Dong, 2017; Zhang & Biswas, 2018).

Figure 12.2 shows the user interface of four primary activities in CTSiM: (1) reading the science library, (2) editing the conceptual model, (3) building the computational model, and (4) comparing the behaviors generated by the students' models to those generated by an expert model. In this example, the student used computational blocks from a domain-specific block-based modeling language to implement the update-speed-and-direction behavior of dye molecule agents. When executed, the animation on the left depicts the motion of the molecules as modeled by the student. Students can compare the motion of molecules in their simulation to the motion of molecules in the expert simulation on the right side of the compare tab. In addition, students can also compare the changes in key simulation variables over time to the expert model. Note that students cannot see the implementation of the expert simulation model; they can only observe the behaviors it generates.


an orange-brown color would appear to flow out of the tea bag. Over time the water will darken and have an orange-brown tint. Because you are able to see this change with your naked eye, we call this a macro-view.

But what if we could see the individual molecules? What would they look like?

Agent Type: Dye-molecules

Procedure: update-speed-and-direction

Fig. 12.2 The user interface of CTSiM

12.3.3 The Assessment Framework

Assessments provide information about students' understanding of the content knowledge in a particular modeling unit, and how well they can apply this knowledge to other problems. This information, therefore, can help teachers, researchers, and other stakeholders evaluate the effectiveness of the teaching and learning processes adopted in the approach (Mislevy, Almond, & Lukas, 2003). The summative and formative assessments, as well as the analytics measures that we can derive from students' models, help us evaluate students' progressions in the STEM content and CT concepts, as well as their model-building and problem-solving behaviors and performance. Students' improvement of understanding the STEM domain content and CT concepts are primarily evaluated by the Domain and CT pre- and post-tests. In addition, the STEM + CT practices are assessed in a summative and formative way, using a range of modalities that include: (1) evaluating the correctness of student-generated models and the evolutionary trajectories of these solutions; and (2) analyzing the sequences of actions students perform in CTSiM.

To formally define students' behaviors in the learning environment, we use a hierarchical model to characterize students' actions in CTSiM. Following the combined theory- and data-driven framework developed from Coherence Analysis (CA) (Segedy, Kinnebrew, & Biswas, 2015; Kinnebrew et al., 2017), we define three types of primary sub-tasks in CTSiM: *Information Acquisition* (IA), *Solution Creation* (SC), and *Solution Assessment* (SA). These primary sub-tasks can be further decomposed. Figure 12.3 shows that the lowest level of the hierarchy represents the different actions that students can perform in the CTSiM environment. For example, SA tasks consist of running, checking, comparing, and testing models.

The conceptual models of CTSiM provide an abstract representation for defining agents and the properties of the environment that affect system behaviors. While constructing the conceptual model, students select the properties and behaviors of the agents that they believe are required to construct the model. The set of properties and behaviors that students choose from may include some that are unrelated to the specific modeling task, and being able to identify and reason about whether or not to include a particular property or behavior is an important modeling practice. We have used the "bag of words" (Piech, Sahami, Koller, Cooper, & Blikstein, 2012) methods to examine the correctness of students' conceptual model compared to the

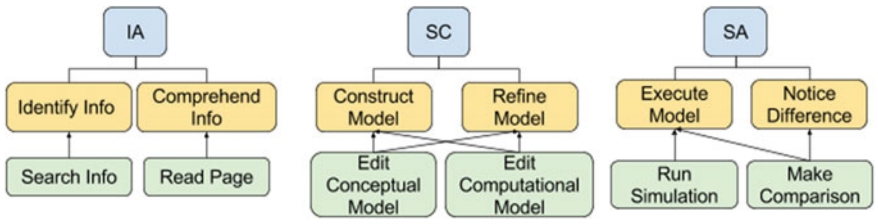


Fig. 12.3 The task model of CTSiM

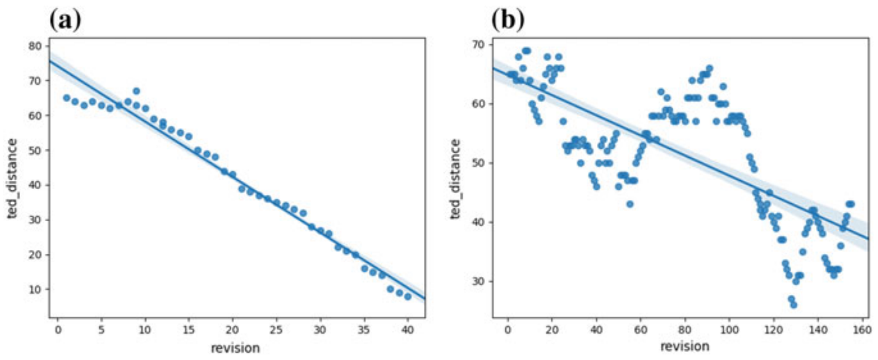


Fig. 12.4 Following two students model building progression using the TED measure

expert models. Another type of summative assessment is based on students' model building performance. We use two measures: (1) model distance, and (2) model trajectory. We calculate the distance between a student-created model and an expert model using the minimal tree edit distance (TED, Bille, 2005) metric, where the models are encoded as abstract syntax trees. In addition, we analyze a student's model building trajectory by computing the slope of distance between the student's models and the expert model as the student performs a sequence of model building actions.

A large model distance indicates an incorrect model, whereas a steeper negative slope indicates faster convergence to the final model. Figure 12.4 shows two students: A's (left) and B's (right) trajectories in building their computational model for the diffusion unit. Both students started with the same empty computational model that had a TED score of 65. Figure 12.4 indicates that student A used much fewer edit actions (~40 edit actions) to build the model compared to student B (~160 edit actions), in building a final model, and the model also had a much smaller final TED (~5 compared to ~45). In summary, Table 12.2 lists the performance expectations and the modality and evidence of the assessment methods for each corresponding practice.

12.4 Results and Discussion

We use results from a recent classroom study to demonstrate students' synergistic learning of STEM and CT concepts and practices with our STEM + CT framework. In the study, 52 6th-grade students from a middle school in the South-Eastern U.S. participated over 14 school days. The students worked on the five units of CTSiM: the two training units, the acceleration unit, the collision unit, and the diffusion unit. Table 12.3 summarizes the daily schedule during the students' hour-long science block. In the rest of the section, we discuss the three types of analyses: (1) the

Table 12.2 The assessment modality and evidence for the STEM + CT practices

	Assessment modality and evidence
ST1	Summative: the correctness of the conceptual models;
ST2	Summative: the domain and CT pre-post questions and the correctness of the computational models;
ST3	Summative: the domain and CT pre-post questions; Formative: average computational edit chunk size
MS1	Summative: the correctness of the conceptual models; Formative: SC-related CA metrics;
MS2	Summative: the correctness of the computational models Formative: SC-related CA metrics;
MS3	Formative: SA-related CA metrics;
MS4	Formative: CA metrics associated with SA $\rightarrow$ SC, and SA $\rightarrow$ IA transitions
PS1	Formative: aggregated IA-, SC- and SA-related CA metrics;
PS2	Formative: computational model evolution and SC-related CA metrics;
PS3	Formative: aggregated SA and SC-related CA metrics; CA metrics associated with SA $\rightarrow$ SC (SA actions followed by SC actions) transitions
DI1	Formative: IA-related CA metrics (supporting action percentage, duration, etc.);
DI2	Summative: domain and CT pre-post questions; Formative: inquiry learning activities in the system;
DI3	Summative: domain and CT pre-post questions; Formative: inquiry learning activities in the system; SA- and SC-related CA metrics
DI4	Summative: answers to the problem-solving inquiry questions; Formative: inquiry learning activities;

learning gains in the STEM domain content knowledge and key CT concepts; (2) the synergy between STEM and CT learning; and (3) students' application of STEM + CT practices as well as their links to the learning gains and model-building performance.

12.4.1 Overall Learning Gains

The pre-post-test results demonstrate students' improved understanding of the STEM domain content and CT concepts. Table 12.4 summarizes the students' pre- and post-test scores, and test statistics in the Mechanics unit, Diffusion unit, and CT. The max scores for the tests were 28, 12 and 34 points, respectively. We ran the non-parametric Wilcoxon Rank Sum Test to assess the statistical significance of the post-to-pre learning gains. The results in Table 12.4 show that students had significant learning gains in domain knowledge (mechanics and diffusion) and CT concepts with moderate effect sizes.

Table 12.3 Summary of daily activities in the CTSiM study

Day(s)	Description of activities
1	Domain and CT pre-tests
2	Agent-based modeling mini-lecture, tutorial of the CTSiM user interface
3–4	Training units, build software agents that draw turtle geometry shapes, practice assigning values and using loops
5–7	Acceleration unit, model the relationship of force, mass, and acceleration of cars, understand acceleration in the Newtonian physics
8–10	Collision unit, model inelastic collision of spheres, momentum, and conservation of momentum in a system
11–13	Diffusion unit, model diffusion of microscopic particles, understand how temperature influences the speed of particles and that diffusion is an emergent behavior of colliding particles
14	Domain and CT post-test

Table 12.4 Pre-post-test results and aggregated test statistics

Test category	Pre mean (std)	Post mean (std)	p-value	Z-stat	Effect size
Acceleration	13.61 (3.40)	15.24 (4.08)	0.003	2.69	0.26
Diffusion	3.81 (2.24)	5.88 (2.40)	<0.0001	4.27	0.42
CT	14.55 (6.27)	18.11 (6.37)	0.0002	3.52	0.35

12.4.2 *The Correlations and Synergies in STEM and CT Learning*

We then computed pairwise correlations between the learning gains in CT, Acceleration, and Diffusion units, as well as the accuracy of the students' computational models in the Acceleration, Collision, and Diffusion units. Table 12.5 presents the correlation coefficients (Spearman's ρ) between all pairs of performance metrics. The asterisks (*) indicate statistically significant correlations ($p < 0.05$).

Students' learning gains in CT showed moderately high and statistically significant correlations with the domain gains in the Acceleration ($\rho = 0.32$) and Diffusion units ($\rho = 0.27$), providing some evidence that there was synergistic learning of STEM content knowledge and CT concepts. The fact that students who improved more in their understanding of CT also achieved larger learning gains in the STEM content supports the notion for synergistic learning through the CTSiM intervention.

Table 12.5 also shows that all of the computational model distances were negatively[1] correlated with the CT gains. Two out of three units (Collision and Diffusion) had statistically significant correlations with the learning gains ($\rho = -0.34$ and $\rho =$

[1]A larger model distance from the correct model indicates a more incorrect model. Therefore, the negative correlations between the model distance and learning gains indicates better model building ability is related to better performance in the domain and CT learning gains.

Table 12.5 Correlation analysis

	CT gain	Acc. gain	Diffusion gain	Acc. distance	Collision distance	Diffusion distance
CT gain	–					
Acceleration gain	0.32*	–				
Diffusion gain	0.27*	0.18	–			
Acc. distance	−0.14	−0.07	−0.10	–		
Collision distance	−0.34*	−0.22	−0.13	0.21*	–	
Diffusion distance	−0.28*	−0.19	−0.31*	0.33*	0.32*	–

−0.28), respectively. The relation between the acceleration learning gain and acceleration test score had a low correlation value and was not significant. In addition, students' CT pre-test scores showed low correlations with model-building performance, therefore, it is unlikely that students' prior CT knowledge was a significant factor in their model building abilities. Overall, the results provide evidence of synergistic learning of STEM domain and CT concepts as students worked on their model building tasks, and it seemed to improve as students worked through different units. The students' computational model building performance was consistent across the three units, as all computational model distances showed moderate correlation values (and the correlations were statistically significant).

12.4.3 The Use of STEM + CT Practices

Finally, we show how the Coherence Analysis-derived (Kinnebrew et al., 2017) metrics can help characterize students' application of STEM + CT practices. Coherence analysis provides a framework for defining a number of metrics related to individual tasks students perform in the system (e.g., seeking information, building models, and checking models) (Segedy et al., 2015). An introduction to the CA-derived metrics was provided in Sect. 12.3.3. In previous work, we have developed a number of these measures to analyze students' work in CTSiM (Zhang et al., 2017). In this chapter, we extend the collection of CA-derived measures to characterize the students' use of STEM + CT practices. Due to limited space, we only report the analyses on the Diffusion unit. We first ran a feature selection algorithm to select features that produced higher percentages of the total variance in the feature space. We assumed these features would better distinguish students who used the STEM + CT practices from those that did not. This approach also helped reduce the dimensions of the feature space for clustering. The eight features obtained after feature selection and their descriptions are summarized in Table 12.6. The features are described as percentages and were computed with respect to the total number of actions performed by a student. For example, the feature "Conceptual Model Edit Effort" of a student accounts for the percentage of computational model edit actions among all her actions in CTSiM. Table 12.6 also lists the STEM + CT practices aligned with the chosen features.

We then clustered the student data using the Expectation-Maximization (Gaussian Mixture Model) algorithm to generate probabilistic models corresponding to each cluster. The Calinski-Harabasz information criteria were applied to select the number of clusters, and applying this measure produced three clusters (Zhang et al., 2017). Table 12.7 reports the mean values and the standard deviations of the eight features for the three clusters. We assumed the feature value probabilities for each cluster would explain the differences in the use of STEM + CT practices among students. We also report the results of single-factor analysis of variance (ANOVA) for each

Table 12.6 Selected features for cluster analyses

Feature	Description	STEM + CT practice category
Domain read time	The total time (in seconds) spent on reading domain content	Data and inquiry (DI1)
Conceptual model edit effort	The percentage of conceptual model edit actions	Systems thinking (ST1) Modeling and simulation (MS1)
Computational model edit effort	The percentage of computational model edit actions	Systems thinking (ST2) Modeling and simulation (MS2)
Model test effort	The percentage of testing actions	Modeling and Simulation (MS3) Problem-solving (PS3)
Model comparison effort	The percentage of comparison actions	Modeling and simulation (MS3) Problem-solving (PS3)
Supported comp. model edit	The percentage of supported computational model edits	Problem-solving (PS1) Modeling and simulation (MS2)
Average computational edit chunk size	The average number of consecutive computational edit actions	Modeling and simulation (MS2)
IA $\rightarrow$ SC rate	The transition probability from IA actions to SC actions	Data and inquiry (DI1) Modeling and simulation (MS1) Modeling and simulation (MS2)

Table 12.7 Selected features for cluster analyses

Feature	Cluster 1 (n = 23)	Cluster 2 (n = 9)	Cluster 3 (n = 20)	F-score	P-value
Domain pre-test	7.14 (4.20)	6.56 (4.28)	5.84 (4.86)	0.42	0.66
CT pre-test	14.57 (3.22)	15.83 (1.05)	13.95 (0.97)	0.27	0.76
Domain gain	1.96 (2.4)	1.89 (2.8)	2.30 (2.8)	0.11	0.9
CT gain	3.39 (6.7)	1.50 (4.8)	4.68 (4.7)	0.97	0.38
Final computational model distance	7.9 (3.21)	2.1 (1.05)	3.0 (0.97)	4.36	0.01
Computational modeling distance slope	−0.50 (0.07)	−0.62 (0.05)	−0.58 (0.04)	3.37	0.04
Domain read time (s)	291 (454)	182 (124)	163 (141)	0.94	0.39
Conceptual modeling editing effort	5% (0.01)	6% (0.01)	7% (0.01)	4.00	0.02
Computational modeling editing effort	15% (0.04)	14% (0.06)	17% (0.07)	0.91	0.4
Supported computational model edits	13% (0.07)	12% (0.11)	16% (10)	0.89	0.4
Average computational edit chunk size	4.57 (0.8)	6.83 (3.5)	5.01 (1.1)	5.96	0.004
Model testing effort	14% (0.04)	10% (0.06)	10% (0.03)	5.02	0.01
Model comparison effort	27% (0.09)	16% (0.13)	14% (0.05)	9.47	<0.001
IA → SC rate	2% (0.01)	13% (0.13)	3% (0.06)	4.38	0.01

feature in Table 12.7 to investigate if the clusters produced statistically significant differences between each other in their learning performances and the CA metrics.

We checked to see if students' prior knowledge in the science domain or CT influenced their learning behaviors. For example, those with higher prior knowledge may find it easier to generate the correct solutions to problems in the learning environment. However, our analysis shows that the students in the three clusters had similar pre-test scores and similar domain and CT learning gains, but there were distinct differences in their learning behaviors that reflected their application of the STEM + CT practices.

Students in Cluster 1 spent the largest amount of time among the three groups in reading the domain library; however, their IA → SC (performing model building actions after their read actions) transition rates were among the lowest. This indicates that although these students read the most, they were not successful in translating the information acquired (IA) into building models (SC) (the differences between the groups were statistically significant). This indicates that Cluster 1 did not show good mastery and use of *Data and Inquiry* practices. Meanwhile, Cluster 1 students had the highest number of testing and comparison (SA) actions (42% of their actions were SA actions), and their computational model edit chunk sizes were the smallest (again, the differences between the three groups was statistically significant). The implication is that students in Group 1 tended to use more trial-and-error *Problem-solving* and *Modeling and Simulation* practices. As shown in our previous work (e.g., Basu et al., 2017; Segedy et al., 2015; Zhang & Biswas, 2018), these trial-and-error approaches adversely affect students' modeling tasks. These students also had significantly larger computational model distances (mean distance = 7.9) when compared to the other clusters (mean distance = 2.1 and 3.0, $p = 0.01$). In summary, students in Cluster 1 had the least effective model-building behaviors and they did not achieve high domain learning gains. However, these type of tinkering behaviors were not necessarily negative. Cluster 1 students remained engaged in their model building tasks, and their learning gains in CT were quite strong although not significantly higher than the students in Cluster 2.

Compared to students in Cluster 1, students in Cluster 2 spent less time reading the domain library, yet their IA $\rightarrow$ SC transition rates were high compared to the other groups. Correspondingly, they achieved the best computational model building performance (mean distance = 2.1 and average computational modeling distance slope = -0.62) with the lowest percentages of SC and SA actions. The combined SC (test + comparison) percentages were the lowest (20%), and the combined SA (conceptual model-building + computational model-building) percentages are also quite low (26%). This reflects good debugging and error-checking practices because they did not need excessive SA—actions while debugging because they were able to pinpoint the issues with their computational models quickly. However, their average computational edit chunk size was the largest, indicating that they did not frequently switch from SC to other types of actions. The students in Cluster 2 had the highest IA $\rightarrow$ SC rate (13%) while the other two clusters only had (2 or 3%). As a result, although these students did quite well on the practices related to *Data and Inquiry* and *Modeling and Simulation*, their *Problem-solving* skills needed improvement. This was also reflected by the fact that they had the lowest percentages of supported computational model edit actions.

Students in Cluster 3 had very similar patterns of learning activity to Cluster 2 except in two features (average computational model edit chunk size and IA $\rightarrow$ SC Rates). Post hoc Tukey's Honest Significant Difference Test revealed that the differences in these two features compared to Cluster 2 were statistically significant ($p = 0.02$ and 0.03, respectively). The students in this cluster had high learning gains in the domain and CT tests and were efficient model constructors as seen from the slopes of their computational model TED values over time (they had small final computational modeling distances and steep negative slopes similar to cluster 2). Students in this cluster performed the most computational model edits, and their average model edit chunk sizes were in between those of Clusters 1 and 2, which indicates they successfully applied problem-decomposition practices for the larger model building tasks. They performed the smallest number of SA actions, however, the percentage of supported computational model edits were the highest among all clusters. This reflects the proficient use of debugging and error-checking practices, which was similar to Cluster 2. However, these students' had a low IA $\rightarrow$ SC transition rate, which implies that they could improve in their *Data and Inquiry* practices. As a result, though they showed good learning gains in the STEM and CT concepts, their computational model building performance was not as good as the students in Cluster 2.

In summary, the clustering results revealed the differences in the use of STEM + CT practices by the students. Although not all of the learning performance metrics and the CA metrics showed significant differences as suggested by the ANOVA results, analyzing these differences in learning behaviors provided insights and links to the students' learning gains and model-building performance.

12.5 Conclusions

In this Chapter, we stressed the benefit and importance of the synergistic learning of STEM and CT concepts and practices. We extended our previous work on the use of CT concepts and practices in STEM classrooms and refined the STEM + CT framework to develop students' synergistic STEM- and CT- learning. The STEM + CT framework defines (1) the practices that are frequently applied in both STEM and CT learning contexts, (2) the STEM domain content knowledge, (3) the set of key CT concepts required for computational modeling, (4) the open-ended learning environment, CTSiM, that fosters students' learning of these STEM and CT concepts and practices, and (5) the assessment framework that provide summative and formative measures for evaluating student performance and learning behaviors. We then used results from a recent CTSiM classroom study in the U.S. to demonstrate how learning can be defined and analyzed with our STEM + CT framework. The results show that students' model-building performances were significantly correlated to their STEM and CT learning, and students' distinct model-building and problem-solving behaviors in CTSiM were indicative of the model-building performance and learning gains.

In future work, we will refine the set of key STEM + CT practices and the assessment framework such that it will be compatible with other learning modalities, such as collaborative learning. We will generalize the CTSiM framework and extend it to different STEM domains, such as the C2STEM learning environment for high school physics. Our overall goal is to study the feasibility and effectiveness of the CTSiM learning environment in multiple contexts and domains.

References

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology, 38*(6), 20–23.

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), 48–54.

Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education—SIGCSE '11*, 245. http://doi.org/10.1145/1953163.1953241.

Basu, S., Biswas, G., & Kinnebrew, J. S. (2017). Learner modeling for adaptive scaffolding in a computational thinking-based science learning environment. *User Model. User-Adapt, 27.*

Basu, S. & Biswas, G. (2016). Providing adaptive scaffolds and measuring their effectiveness in open-ended learning environments. In *12th International Conference of the Learning Sciences* (pp. 554–561). Singapore.

Bille, P. (2005). A survey on tree edit distance and related problems. *Theoretical Computer Science, 337*(1–3), 217–239.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (2000). *How people learn*.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*, Vancouver, Canada (pp. 1–25).

Cheng, B., Ructtinger, L., Fujii, R., & Mislevy, R. (2010). *Assessing systems thinking and complexity in science (large-scale assessment technical report 7).* Menlo Park, CA: SRI International.

García-Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A., & Jormanainen, I. (2016). *An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers*. Belgium: TACCLE3 Consortium.

Grover, S. (2015). "Systems of Assessments" for deeper learning of computational thinking in K-12. In *Proceedings of the 2015 Annual Meeting of the American Educational Research Association* (pp. 15–20).

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.

International Society for Technology in Education (ISTE), & Computer Science Teachers Association (CSTA). (2011). Operational definition of computational thinking, 1030054. Retrieved from http://www.iste.org/learn/computational-thinking.

Kinnebrew, J. S., Segedy, J. R., & Biswas, G. (2017, April). Integrating model-driven and data-driven techniques for analyzing learning behaviors in open-ended learning environments. *IEEE Transactions on Learning Technologies 10*(2), 140–153.

Mislevy, R. J., Almond, R. G., & Lukas, J. F. (2003). A brief introduction to evidence-centered design. *ETS Research Report Series, 2003*(1), 1–29.

National Research Council (U.S.). (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington, D.C: National Academies Press.

National Research Council (U.S.). (2011). *Report of a workshop on the pedagogical aspects of computational thinking*. Washington, D.C: National Academies Press.

NGSS Lead States. (2013). *Next generation science standards: For states, by states*. Washington, DC: The National Academies Press.

Piech, C., Sahami, M., Koller, D., Cooper, S., & Blikstein, P. (2012). Modeling how students learn to program. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (pp. 153–160).

Repenning, A., Ioannidou, A., & Zola, J. (2000). AgentSheets: End-user programmable simulations. *Journal of Artificial Societies and Social Simulation, 3*(3), 351–358.

Segedy, J. R., Kinnebrew, J. S., & Biswas, G. (2015). Using coherence analysis to characterize self-regulated learning behaviors in open-ended learning environments. *Journal of Learning Analytics, 2*(1), 13–48.

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.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (2016a). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.

Weintrop, D., Holbert, N., Horn, M. S., & Wilensky, U. (2016b). Computational thinking in constructionist video games. *International Journal of Game-Based Learning, 6*(1), 1–17.

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. In *The 43rd ACM Technical Symposium on Computer Science Education* (pp. 215–220).

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Wing, J. M. (2011). *Research Notebook: Computational Thinking–What and Why*?. Retrieved January 1, 2017 from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why.

Wilensky, U. (1999). *NetLogo*. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston. Retrieved January 1, 2017 from http://ccl.northwestern.edu/netlogo.

Wilensky, U., & Resnick, M. (1999). Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and 489 technology, 8*(1), 3–19.

Zhang, N., & Biswas, G. (2017). Assessing students' computational thinking in a learning by modeling environment. In S.-C. Kong (Ed.), The Education University of Hong Kong, Hong Kong (pp. 11–16).

Zhang, N., Biswas, G., & Dong, Y. (2017). Characterizing students' learning behaviors using unsupervised learning methods. In E. Andre, R. Baker, X. Hu, M. M. T. Rodrigo, & B. du Boulay (Eds.), (pp. 430–441). Cham: Springer.

Zhang, N., Biswas, G. (2018). Understanding students' problem-solving strategies in a synergistic learning-by-modeling environment. In C. Penstein Rosé et al (Eds.), *Artificial intelligence in education. AIED 2018. Lecture Notes in Computer Science* (Vol. 10948). Cham: Springer.