




# Domain-Specific Modeling Languages in Computer-Based Learning Environments: a Systematic Approach to Support Science Learning through Computational Modeling

Nicole M. Hutchins<sup>1</sup>  · Gautam Biswas<sup>1</sup> · Ningyu Zhang<sup>1</sup> · Caitlin Snyder<sup>1</sup> · Ákos Lédeczi<sup>1</sup> · Miklós Maróti<sup>2</sup>

Published online: 9 September 2020

© International Artificial Intelligence in Education Society 2020

## Abstract

Driven by our technologically advanced workplaces and the surge in demand for proficiency in the computing disciplines, it is becoming imperative to provide computational thinking (CT) opportunities to all students. One approach for making computing accessible and relevant to learning and problem-solving in K-12 environments is to integrate it with existing Science, Technology, Engineering, and Math (STEM) curricula. However, novice student learners may face several difficulties in trying to learn STEM and computing concepts simultaneously. To address some of these difficulties, we present a systematic approach to learning STEM and CT by designing and developing *domain-specific modeling languages* (DSMLs) to aid students in their model building and problem-solving processes. The paper discusses a theoretical framework and the design principles for developing DSMLs, which is implemented as a four-step process. We apply the four-step process in three domains: Physics, Marine Biology, and Earth Science to demonstrate its generality, and then perform case studies to show how the DSMLs impact student learning and model building. We conclude with a discussion of our findings and then present directions for future work.

**Keywords** Learning-by-modeling · Stem+CT · Synergistic learning · Evidence-centered design · Domain-specific modeling language

---

**Electronic supplementary material** The online version of this article (<https://doi.org/10.1007/s40593-020-00209-z>) contains supplementary material, which is available to authorized users.

---

✉ Nicole M. Hutchins  
nicole.m.hutchins@vanderbilt.edu

<sup>1</sup> Vanderbilt University, Nashville, TN, USA

<sup>2</sup> University of Szeged, Szeged, Hungary

## Introduction

Computational modeling is fundamental to the learning and practice of science (Wing 2011; Grover and Pea 2018). This multifaceted process includes the building, evaluating, and revising of models based on the learners' underlying understanding of the domain theory and relations that govern the behavior of the model (Schwarz and White 2005). *Learning-by-modeling* encompasses these processes, facilitating model building, simulation, and analysis supported by the use of a *modeling language*. Adopting *learning-by-modeling* approaches in K-12 STEM (Science, Technology, Engineering, and Math) classrooms has proven to be an effective vehicle for integrating the teaching and learning of STEM and computational thinking (CT), helping students become active constructors as opposed to passive consumers of scientific knowledge and practices (Lehrer and Schauble 2015; Clark et al. 2009; Sun and Looi 2013; Wieman et al. 2008). Technology-enhanced environments can be productive avenues for scaffolding and engaging students in modeling processes that support STEM inquiry and problem-solving (e.g., Jonassen et al. 2005; Keating et al. 2002; Sengupta et al. 2013; Bredeweg et al. 2013; Shen et al. 2014; Weintrop et al. 2016).

In the context of a K-12 STEM classroom, *learning-by-modeling* approaches have adopted multiple representations that include: (1) constraint-based (Leelawong and Biswas 2008; Bredeweg et al. 2013); (2) system-dynamics (van Joolingen et al. 2005; Metcalf et al. 2000; VanLehn et al. 2015); and (3) agent-based (Basu et al. 2013; Nikolai and Madey 2009; Wilensky et al. 2014; Wilensky and Reisman 2006) representations to support model building. These representations are translated into an executable form by mapping them on to programming constructs that express the dynamic behavior of the target STEM phenomena. To create an interpretable and transparent representation that students can work with, it is important to develop a seamless interweaving of the STEM and computing concepts in developing the *modeling language*. However, it is also well-known that the advantages provided by these *modeling languages*, i.e., the support for *synergistic learning* of STEM and computational concepts and practices (Sengupta et al. 2013; Snyder et al. 2019), can pose significant challenges for students who are novices in the science and CT domains (Basu et al. 2016b; Chi 2005; Sengupta and Farris 2012; Araujo et al. 2008) and unfamiliar with the learning environment and its associated components (e.g., the underlying *modeling language*) (VanLehn 2013).

To support the learning of science with understanding, the designed modeling environments must incorporate appropriate and intuitive representational mechanisms linked to interpretable computational (i.e., language) constructs that can be easily adopted by classroom teachers and students, and are situated in the relevant domain concepts and practices. Designing *modeling languages* whose component structures clearly imply their associated semantics provides a method for accomplishing these links (VanLehn 2013).

*Domain-specific modeling languages* (DSMLs), prevalent in the software engineering community for application-specific modeling and analysis (Ledeczi et al. 2001), may provide the needed framework to establish the simultaneous '*situating*' in domain modeling and CT constructs seamlessly to better support the synergistic learning of STEM and CT through computational modeling. DSMLs are categorized by their "*focused, expressive power in the problem domain*" (Van Deursen et al. 2000, p. 26).

They support modeling, analysis, and verification of models, while also providing the needed *abstractions* for model building and problem solving by domain experts who are not computer science researchers and practitioners in their domain-specific tasks (Karsai et al. 2014; Ledeczi et al. 2001).

In this paper, we target the systematic design and development of *educational DSMLs* for STEM domains. We develop the framework and conduct case study analyses to address the following research questions:

1. *What are the design steps necessary for ensuring a proper and transparent mapping from STEM (and specifically, science) curriculum learning objectives to the required DSML constructs for model building?*

We address this question in the “[Establishing an Educational DSML Design Process](#)” section of the paper.

2. *How do the DSML-created abstractions support learning of STEM and CT?*

We address this question in three parts in the sections “[Impact on Student Learning and Performance](#)” and “[DSMLs Support Learning-by-Modeling in STEM Classrooms](#)”: (1) how DSMLs help students build correct computational models of scientific processes; (2) how the DSML structures we create help students to debug their models and correct misunderstanding and errors; and (3) how DSMLs support the classroom teacher, and help the grounding of *modeling language* semantics in science domains.

To characterize and situate our approach, we develop a theoretical framing for our *learning-by-modeling* approach, compare it to alternate modeling approaches presented in the literature, and discuss the pros and cons of each of these approaches in the context of K-12 science education. We then provide background on DSMLs developed in software engineering, including design frameworks for developing DSMLs. We provide an alternate framing for DSMLs, linking them to knowledge representation and reasoning mechanisms in Artificial Intelligence. The resulting design framework is based on three design principles:

1. An *evidence-centered design* approach (Mislevy and Haertel 2006) to ensure coherence and coverage of domain concepts, relations, and practices in the design of the DSML constructs;
2. Opportunities for *exploratory learning of dynamic processes* by creating a discrete-time *step-by-step* model of continuous dynamic processes (diSessa 2001; Redish and Wilson 1993; Wilensky and Reisman 2006), and
3. A *block-based programming language* (Brown et al. 2016) to lessen programming difficulties among novice learners, while supporting the effective use of CT principles (e.g., conditional logic, initializing and updating variables).

We apply this design process to develop curriculum modules that we have designed in different K-12 science domains (Physics, Marine Biology, and Earth Sciences) for elementary, middle, and high school students. We then demonstrate the effectiveness of our DSML approach using case studies that target STEM and CT learning for students at different levels, while also demonstrating how DSMLs support classroom

instruction. We conclude by summarizing the effectiveness of our DSMLs in supporting *learning-by-modeling* in K-12 classrooms and outline directions for future research.

## Background

This section presents our approach to learning science by constructing computational models and the benefits of this approach. We also discuss the difficulties students face when working in this framework in K-12 science classrooms.

### *Learning-by-Modeling*

The *learning-by-modeling* framework, illustrated in Fig. 1 (adapted from the Common Core Mathematics Standards (CCSSO 2011)), highlights the role that different sub-processes play in acquiring, interpreting, and refining one’s knowledge when performing computational modeling tasks. The sub-processes illustrated match the Next Generation Science Standards (NGSS) on “*Developing and Using Models*” (NGSS 2013) and define the key processes that a DSML developed for educational purposes must support. We investigate and analyze the subprocesses marked with a “\*” in our educational DSML case studies that we discuss later.

Our work focuses on developing comprehensive *agent-based* models, that capture the emergent behavior of relevant scientific phenomena expressed using computational constructs and a discrete-time (*step-by-step*) simulation framework. Agent-based modeling has received significant attention as a means for supporting STEM learning adopting a multi-representational approach. To gain a deeper understanding of the science phenomena, students can link the behaviors generated by their agent models to animations and plots to depict the generated behaviors. NetLogo (Wilensky and Reisman 2006), CTSiM (Basu et al. 2013), AgentSheets (Repenning, et al. 2010), and Scratch (Resnick et al. 2009) are examples of environments that adopt agent-based modeling approaches. Agent-based modeling can be contrasted from *constraint systems* (e.g., Betty’s Brain (Leelawong and Biswas 2008) and DynaLearn (Bredeweg et al. 2013)), which use sequences of causal relations to model system behaviors, and *systems dynamics models* (e.g., Dragoon (VanLehn et al. 2015)), which use simplifications of differential equations to represent dynamic system behaviors. Table 1 outlines the key benefits and difficulties experienced in the three major modeling

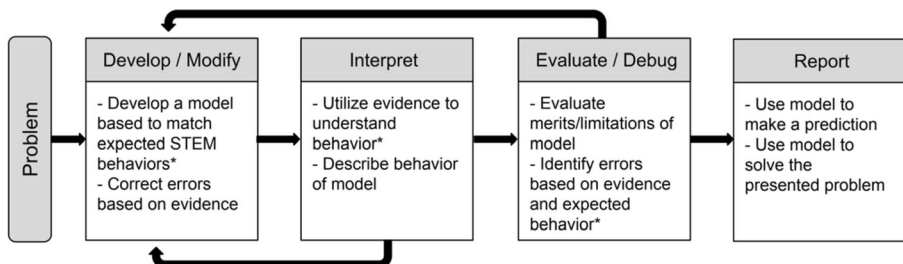


Fig. 1 Processes and subprocesses integral for learning-by-modeling

**Table 1** Comparison of *modeling language* approaches for STEM learning-by-modeling

Language	Benefits	Difficulties
Agent-based	Engaging programming context with automatic visual representation from running code - Relationship to object-oriented programming supports mapping to real world objects	Understanding emergent system behavior from individual agent behavior Designing and implementing mathematical and logical expressions - Lack of grounding in custom variable and block definitions
Constraint-based	Explicit representation of relations between variables in a domain - Ability to analyze relations with simplified qualitative representations	Difficult to extend qualitative reasoning to quantitative representations - Not easy to describe the dynamic nature of processes as a simulation
Systems dynamics	- Matches traditional representations in science and engineering for modeling dynamic processes	Translation of mathematical equations to model form, especially when students lack knowledge of calculus and higher algebra - Grounding of symbolic representations

approaches. The shortcomings described, such as student difficulties in understanding the simplified forms of differential equations representing dynamic processes in systems dynamics models (Wetzel et al. 2017), provide a rationale for our work for reducing student difficulties and leveraging learning-by-modeling as a tool for the *synergistic learning* of STEM and CT.

Bolstered by the contributions of the Netlogo environment and systems that extend this approach, agent-based modeling has demonstrated the benefits of learning systems behavior by utilizing a modeling language approach, especially when the modeling language is implemented using a block-based representation (Sengupta et al. 2013, 2018). However, to our knowledge, not much research has established systematic design processes for developing agent-based *modeling languages* that focus on abstractions and visual block-structured constructs to reduce the modeler's programming burdens while keeping the focus of learning on the STEM domain concepts.

### Students' Difficulties with *Learning-by-Modeling*

The need to prepare students to utilize computational tools as vehicles for problem-solving and professional advancement (Dede 2010; Wing 2011; Hilton 2010) conforms to the NGSS call for engaging students in authentic modeling practices in science and integrating computational modeling into the K-12 science curriculum. However, research studies show that this integration can create difficulties in student learning. For example, students often face difficulties in:

1. Translating learnt domain knowledge into computational forms for model building (Sengupta et al. 2013; Basu et al. 2016b)
2. Integrating key aspects of programming and CT (e.g., programming language syntax), then identifying appropriate abstractions and developing iterative structures to model the temporal dynamics of the scientific processes (Hutchins et al. 2020)

3. Relating the behavior of individual entities to aggregated or emergent system behaviors (Chi 2005; Wilensky and Resnick 1999)
4. Understanding the mathematical relations between variables and interpreting graphs in relation to generated simulation behaviors (Sengupta and Farris 2012; Araujo et al. 2008)
5. Debugging the behaviors (results) generated by the abstract modeling representations and interpreting them in terms of scientific principles and theories (Basu et al. 2016b).

These difficulties can be mapped on to the subprocess illustrated in Fig. 1, and they need to be addressed in the context of these subprocesses to make the *learning-by-modeling* approach a productive experience for novice learners.

Additional concerns about computational modeling in science arise from a teaching and classroom perspective. These include developing a shared understanding between the teacher and the students of the *modeling language* used for constructing the science models and understanding the behaviors generated when the model is executed (VanLehn 2013). We hypothesize that this may be exacerbated when students create their own modeling structures (e.g., creation of custom blocks in Scratch or Snap!), especially when students come in with STEM and CT domain misunderstandings prior to their model building activities (Sengupta et al. 2013). In addition, these approaches may increase the training requirements for teachers and students (e.g., class time spent on learning the *modeling language*) to establish a sufficient understanding of the computational constructs needed to build meaningful science models and to communicate results. This is especially true when text-based programming languages are used (e.g., Hashem and Mioduser 2011; Sherin et al. 1993).

It is important that *learning-by-modeling* systems support synergistic STEM + CT learning, as opposed to dealing with the dual tasks of STEM learning and CT learning. This may require establishing a tight coupling and a shared semantics between the STEM *modeling language* and the CT constructs needed to build the models, and serve as a basis for developing educational DSMLs to support *learning-by-modeling* in K-12 STEM classrooms. A DSML approach to designing *learning-by-modeling* environments may provide the necessary scaffolding that overcomes the difficulties discussed above, and therefore, provides a systematic approach to synergistic STEM + CT learning. In this paper, we present a structured design process for developing educational DSMLs.

## What Are Domain-Specific Modeling Languages?

Abstraction is fundamental to developing good software systems (Hudak 1996); however, determining the right level of abstraction is often application- and task-dependent. Therefore, it becomes important to involve domain experts when constructing task-specific environments for specific application domains. An approach to developing computational task-specific environments has been domain-specific languages (DSLs), which are “a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain” (van Deursen et al.



2000, p. 26). General-purpose programming languages like C and Java, and even languages defined for specific purposes, such as Fortran for scientific problem-solving and Lisp for symbol processing, include general language constructs that can be applied to multiple problem domains. While generality increases expressiveness, the complexities that come with these generalities can overwhelm practitioners who are not well-versed in the programming language. In comparison to general-purpose *modeling languages*, DSLs help to reduce common programming difficulties linked to syntax and logic errors, as well as errors related to applying domain constraints and rules that govern model construction processes (Kelly and Tolvanen 2008).

*Domain-specific modeling languages* (DSMLs) have evolved from DSLs. They are designed to enhance the readability of constructs (van Deursen 1997), increase the level of abstraction to enable a large number of potential users (Karsai et al. 2014), support application development by experts (Kelly and Tolvanen 2008), and provide domain-specific documentation for subsequent analysis and use (Kelly and Tolvanen 2008). It is important to note that guidelines for developing DSMLs start with establishing a thorough understanding of the application domain, including clearly defining the domain boundaries and constraints, and collecting relevant conceptual knowledge for developing the programming constructs. DSMLs have added benefits, providing for systematic verification methods for model performance against domain-specific criteria, expressing and automatically enforcing integrity constraints, and providing “integrated models as opposed to relying merely on source code” (Ledeczi et al. 2001, p.44). For educational purposes, these characteristics are particularly useful as students can directly express model behaviors in terms of the laws of the domain thus making it easier to interpret and verify model performance. DSML applications maintain the ability to represent data in different formats (Kelly and Tolvanen 2008) and allow for representations other than text - thus providing for different views and allowing for additional scaffolding in terms of information provided by the models.

Initially outlined by Paige, Ostroff, and Brooke (2000) and later refined by Karsai et al. (2014), the general principles for developing *modeling languages* include simplicity, uniqueness, consistency, and scalability. The principles are actualized in the implementation of Karsai et al.’s (2014) DSML development guidelines that outline key categories, including Language Purpose and Consistency, Language Realization, and Language Content and Scalability. Consistency is established by clearly defining the target audience of the language (e.g., upper middle school students), associated models (e.g., for different science topics), and language characteristics (e.g., graphical or text-based) (Karsai et al. 2014). When considering the Language Content, simplicity is emphasized through recommendations, such as avoiding unnecessary generality and conceptual redundancy, as well as utilizing only the necessary domain concepts (constrain concept selection to those needed for modeling and problem-solving in the domain). Simultaneously, a core aim of DSMLs is to raise the level of abstraction to gain simplicity and the increased efficiency of language use. This influences the Language Content, i.e., avoid the creation of unnecessary language elements and impose modularity in the DSML design process. Scalability is impacted by the possibility of extending an existing language whilst adhering to the seamless principle - concepts of the language need to be consistent (Karsai et al. 2014). Adopting this framework, we will focus on embedding a DSML into an existing modeling framework (van Deursen et al. 2000); specifically, a *block-based environment* in which

computational block constructs are derived from domain principles. We will discuss this further in the “Framing the Design of DSMLs for *Learning-by-Modeling* Environments” section.

From another viewpoint, DSMLs are analogous to *knowledge representation* (KR) schemes in AI with their associated *reasoning mechanisms* (Levesque 1986; Niwa et al. 1984). Much like the formulation of DSMLs, KR research addresses the problem of finding compact and expressive languages to represent domain knowledge in a form that computer algorithms can be designed to make inferences using the knowledge structures created. Similar to the DSML structures developed in software engineering, good KR schemes must be complete in terms of the domain of interest (e.g., laws of kinematics), they must be computable, the important concepts and relations must be made explicit and accessible, they should be transparent, and the scheme must suppress irrelevant detail (Brodie et al. 2012). Given this significant overlap, we incorporate methods from KR into our design of DSMLs for educational applications, paying particular attention to the reasoning or inference mechanisms that accompany the DSML structures, and ensuring that the computational algorithms are easy to follow.

The users of DSMLs in engineering applications are generally domain experts. However, in our *learning-by-modeling* environments, the focus of DSML design is to support novice students in learning science and CT concepts and practices. Traditionally, the DSML constructs should “adopt existing notations domain experts use” (Karsai et al. 2014). In the educational context, this equates to representing the scientific laws as relations between variables expressed using computational structures at the appropriate levels of abstraction. The syntax (i.e., representation) chosen for a particular domain may include additional notations that are tailored to the type of problem solving being addressed in the domain. Karsai et al. (2014) recommend using descriptive notations, distinguishing between language elements, organizational structures of models, and supporting documentation of the model. Such approaches also support the developer’s ability to maintain consistency through the design and development of the DSML. For example, designers of the Snap! programming environment (Harvey et al. 2013) provided two distinct blocks: (1) reporter blocks that report a value of a variable, and (2) command blocks that carry out actions in the environment governed by the laws of the domain. To make it easier for novice programmers, distinct block types are identified by different shapes and colors.

We adopt the general principles of DSML design outlined above along with the property of transparency of the accompanying reasoning (model execution) methods to help students develop models and interpret and analyze the behaviors generated by the models.

## Framing the Design of DSMLs for *Learning-by-Modeling* Environments

Our DSML design process is guided by three key principles for learning by modeling applications: (1) Adopting an *evidence-centered design* approach to determine the set of concepts and practices that need to be included; (2) Supporting *exploratory learning of dynamic processes* by tailoring the simulation model for *step-by-step* analysis; and (3) Utilizing a *block-based programming environment* (BBPE) to simplify the syntax and semantics of the *modeling language*.



## Evidence-Centered Design

Evidence-centered design (ECD) ensures a rigorous framework for structuring domain concepts and practices, and the design of educational assessments to link claims about student learning with evidence from learning products as well as features of the instructional tasks (Mislevy and Haertel 2006; Mislevy and Riconscente 2005; Hutchins et al. 2020). For educational DSMLs, ECD serves as a systematic approach to conducting integrated STEM + CT domain analysis in a way that the principles of completeness, simplicity and consistency discussed in the “[What Are Domain-Specific Modeling Languages?](#)” section can be applied to the design and development of the DSML constructs. Domain analysis in the context of ECD “leads us to understand the knowledge people use in a domain, the representational forms, characteristics of good work, and features of situations that evoke the use of valued domain knowledge, procedures, and strategies” (Mislevy and Haertel 2006, p.7). The completion of this process results in a comprehensive understanding of the STEM and CT domain constructs at the level of abstraction in which they will be utilized for computational modeling in the integrated curriculum. Consequently, this step must:

1. identify the target audience (typically the grade level of the students),
2. establish boundaries on what needs to be part of the learning content,
3. identify the prerequisite knowledge and skills required to achieve proficiency,
4. define known difficulties in acquisition and understanding of the STEM and CT constructs, and
5. describes essential relationships among and between the STEM and CT concepts.

Careful consideration of each must be made to ensure consistency in the DSML design. Discrepancies regarding issues such as boundaries and abstraction levels may result in difficulties defining block semantics (e.g., multiple options for a single construct may result in differences in the meaning of the block structures for different users).

Once the contributions of the ECD process are established, the DSML constructs to support model building tasks can be developed accordingly. For instance, it has been found that students struggle to distinguish velocity from acceleration (Trowbridge and McDermott 1981). Representational structures designed for modeling these kinematics concepts must, therefore, (1) support applications that help target this misunderstanding and (2) provide evidence of appropriate or inappropriate applications of the relations between the variables. We hypothesize that the appropriate use of such structures will support the learning of the velocity-acceleration relationship, and provide a case study in the “[Impact on Student Learning and Performance](#)” section that supports our hypothesis.

In selecting the abstraction level, we have to consider the students’ level of mathematical understanding. Therefore, the mechanics curricular content for middle school may involve concepts and relations expressed as qualitative relations (e.g., a larger force applied to an object creates a greater acceleration), whereas a high school curriculum uses analytic representations (e.g.,  $acceleration = \frac{Force}{mass}$ ). We will elaborate on this in the “[Establishing an Educational DSML Design Process](#)” section. In particular, we will discuss applications of ECD processes in steps one and two of our DSML design process.

## Exploratory Learning of Dynamic Processes

The principle of Exploratory Learning of Dynamic Processes builds on our efforts for targeting DSML support of modeling processes and linking domain principles and laws to behaviors generated by these models (Fig. 1). Technology-enhanced, modeling-based instruction may have particular added benefits for learning because it allows students to explore science phenomena and processes that are complex, abstract, too small, or dangerous to investigate in real-world contexts. For example, studying a dynamic phenomenon as a progression or a sequence in discrete time steps makes it easier for students to understand system behavior in contrast to studying continuous-time processes derived from equations (Sengupta et al. 2013; diSessa 2001; Sherin 2001b). Decomposing complex behaviors (e.g., into the up and down motion of an object thrown up) also helps students reason about boundary conditions, such as: *during upward motion does velocity decrease with time? What happens when it becomes zero?* Similarly, when the ball falls down to earth, its velocity increases, but *what happens when it hits the ground?*). In other words, the modeling process includes assumptions that can be made explicit and linked to observed behaviors (diSessa 2001; Sengupta et al. 2013; Sherin 2001a). Visualizations afforded by the computational environment (both animation of behaviors and graphs of variables) make it easier for learners to observe and analyze whether the behaviors generated by the constructed model seems reasonable (Sherin 2001a). Further, computational modeling supported by simulations, animations, and plots can facilitate learning by providing rapid feedback through adaptive scaffolding (Shen et al. 2014), and exposing students to multiple representations of knowledge (Basu et al. 2016a; Jonassen et al. 2005).

As an extension to the pedagogical benefit of studying a phenomenon as a discrete process of steps in time (Sengupta et al. 2013; diSessa 2001), the environment used in our approach is designed for students to build their computational models using a *step-by-step* simulation approach. This includes two primary considerations. First, the temporal evolution of a moving object's behavior can be decomposed into a set of fundamental and discrete processes; for example, the motion of an accelerating object is modeled in two steps: (1) an update in the object's velocity based on acceleration, and (2) an update in the object's position based on its velocity. Second, particular attention is given to  $\Delta t$ , the simulation time step. By changing values of  $\Delta t$ , students can study its effects on behavior generated for the object, thus gaining an understanding of how the choice of the discrete-time step affects the continuous behaviors generated by their simulation model. Typically, this concept can be discussed in-depth with students who have had some background in calculus. For upper middle school students, this concept can be abstracted in a way that students work with fixed rates. For lower middle school students, the concept of rate may not be made explicit in the model building process, and students can express relations between variables qualitatively, e.g., velocity increases by acceleration amount (Basu et al. 2013), or the qualitative process theory models used in DynaLearn (Bredeweg et al. 2013). In addition to the DSML constructs, the *step-by-step* simulation approach also provides a transparent and easy to understand reasoning scheme for execution of the models created by the DSML constructs. Much like an AI reasoning scheme, this *step-by-step* approach is well-defined and general, and applies across all science domain models created using the DSML constructs.

In summary, this approach allows students to apply the decomposition process during model building, i.e., to build their model in parts (Basu et al. 2017), visualize the behaviors of their modeled entities through simple animations, and observe the evolution of variable values over time, using a watch function or by generating plots (Hutchins et al. 2020; Zhang et al. 2017). In previous studies, students building kinematics models with well-defined DSML block structures found the decomposition and *step-by-step* simulation to be particularly beneficial for understanding and debugging the motion of objects (Hutchins et al. 2020). In contrast, we have found that students who are not able to grasp the *step-by-step* simulation algorithm may have difficulties with key modeling processes, such as behavior interpretation and using evidence to verify the behavior of the model. Further analysis of this issue is warranted and will be discussed in the “[Impact on Student Learning and Performance](#)” section.

### DSML Development Utilizing a Block-Based Programming Environment

Visual programming environments play a key role in introducing K-12 students to computer science curricula (Bau et al. 2017, Grover and Basu 2017). These environments typically provide a set of block constructs that can be selectively dragged onto a canvas and assembled in a puzzle-like, connected computational structure. When executed, the blocks execute sequentially and produce visible results that are easier to interpret than conventional programs. Examples of such environments include Scratch (<https://scratch.mit.edu>), Snap! (<http://snap.berkeley.edu>), App Inventor (<http://appinventor.mit.edu>), and App Lab (<http://code.org>). Block-based programming environments (BBPEs) simplify program syntax and reduce program creation time (very little typing is needed to build the program). They also make it easier for students to create and visualize the algorithmic structure and flow of the created program. For execution, the blocks are converted into more conventional linear, text-based code representations under the hood, which are executed much like a traditional general-purpose programming language. Therefore, the block-based languages provide additional scaffolds. The block descriptions are abstracted with details of the execution code hidden from the students, making it easier for them to focus on the structure and logic of their programs, rather than the syntax and semantics of how the computer specifically executes the code (Selic 2007). Thus, they facilitate learning of domain and CT concepts and practices, and do not overwhelm novice learners (students) with the syntax of traditional programming languages (Grover and Basu 2017, Brennan and Resnick 2012, Werner et al. 2013; Koh et al. 2010). We hypothesize that this approach not only targets difficulties related to programming and CT, but it will lessen the training necessary before students can engage in meaningful model-building activities.

BBPEs have been used for developing STEM curricula by adopting a *learning-by-modeling* approach. Examples include CTSiM, ViMap, and CT-STEM (Basu et al. 2013, Sengupta et al. 2015, Jona et al. 2014). These environments extend NetLogo, a multi-agent programming language for building models that simulate the dynamic behavior of complex natural and social phenomena (Wilensky and Resnick 1999; Tisue and Wilensky 2004). NetLogo provides a simple agent *modeling language* that allows students to create their own models or modify existing models that are available in a large library that comes with the environment. This enables learners to simulate and

“play” with their models, exploring their behavior under various conditions. CTSiM (Basu et al. 2013), ViMap (Sengupta et al. 2015) and CT-STEM (Jona et al. 2014) provide a block-structured visual programming environment as an abstraction layer over NetLogo. This allows students to focus on the domain modeling tasks, without being overwhelmed by the syntax of the NetLogo programming language. Classroom studies conducted with these systems have produced very successful results, demonstrating significant learning gains in the STEM and CT domains (e.g., Basu et al. 2017; Sengupta et al. 2012), promoting collaborative problem solving skills (Hutchins et al. 2018; Snyder et al. 2019), and lowering the burden of programming syntax in order for students to focus on the domain modeling tasks (Weintrop et al. 2016).

To extend the scope of BBPEs and to support learning opportunities in STEM domains, additional scaffolding will be needed within the visual programming environment to create a low entry threshold for learning by modeling in STEM domains (Sengupta et al. 2013). To do so, we adopt a more abstract, domain-specific, visual language to aid students in model building in the integrated STEM+CT *learning-by-modeling* environment by (a) shifting the emphasis from programming to a more foundational introduction to computing concepts and practices, and (b) getting the students to focus on domain-specific concepts (e.g., position, velocity, and acceleration in kinematics). Furthermore, this shift not only facilitates a deeper understanding of the STEM concepts (e.g., Basu et al. 2017; Zhang et al. 2017), but may also support developing foundational computing skills and knowledge to support the future learning of advanced computer science (CS) concepts.

## Establishing an Educational DSML Design Process

In this section, we discuss the design and deployment of DSMLs for curriculum units in three different science domains: (1) Kinematics; (2) Marine Biology; and (3) Earth Sciences. The DSML structures in these domains form a core component of our Collaborative, Computational STEM (C2STEM) learning environment (Hutchins et al. 2020). C2STEM is built on top of the Snap! block-structured programming language. Although we provide examples from three different topics with respect to our C2STEM environment, similarities among block-based programming environments (e.g., Alice (Pausch et al. 1995), App Lab (code.org), App Inventor (Tissenbaum et al. 2019), Scratch (Brennan and Resnick 2012), NetTango (Olson et al. 2011), and Snap!) makes our DSML approach generalizable to many environments. We revisit this issue in the “Discussion and Conclusions” section.

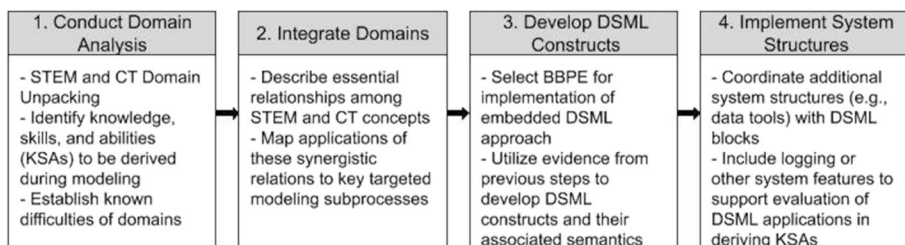


Fig. 2 Educational DSML design process

## Designing Educational DSMLs

The systematic development of educational DSML design leverages our design principles (Section: "[Framing the Design of DSMLs for Learning-by-Modeling Environments](#)") to develop our four-step educational DSML design process that is illustrated in Fig. 2. This design process explicitly answers our first research question on the steps needed to create a proper and transparent mapping from the STEM and CT learning objectives to the DSML constructs developed to support model building in the STEM domain.

To demonstrate an application of the educational DSML design process, we will step through a Physics application. Steps 1 and 2 of the educational DSML design process involve the application of ECD that provides a comprehensive, integrated domain analysis, as outlined in the "[Evidence-Centered Design](#)" section. The current DSML for the kinematics domain in Physics covers Newton's first and second laws of motion, and is directed toward high school Physics curricula based on Tennessee state and NGSS standards. Given the target audience and general curriculum goals, our research team first completed the domain unpacking process, with example Physics, CT, and computational modeling constructs listed in Table 2. This information provided the basis for deeper evaluations of student difficulties with the outlined concepts and practices (e.g., students have difficulties translating mathematical constructs into a computational form). Given the requirements of a high school Physics curriculum, the DSML blocks need to support applications of kinematic equations (instead of a more qualitative representational approach that we have used in middle school classrooms (Basu, et al., 2013, 2017)). The students need to apply their understanding of the kinematic equations to select appropriate DSML constructs to build their model, and

**Table 2** Step 1 - Domain analysis considerations for a Physics implementation

Domain	Concepts and Practices
Science Disciplinary Concepts	<p>(NGSS: PS2.A: Forces and Motion)</p> <p>Relations between velocity, position, and time</p> <p>Velocity-time and position-time graphs</p> <p>Relations between acceleration, velocity, position, and time.n. Revisit velocity-time, and position-time graphs</p>
Computational Thinking Concepts	<p>(K-12 CS Framework: Algorithms &amp; Programming, Data Analysis)</p> <p>Initializing and updating variables</p> <p>Operators and expressions</p> <p>Control structures: Event handlers, conditionals, iterations (as expressed using a simulation step which is an implicit loop in the simulation environment)</p> <p>n. Data collection and visualizations as graphs</p>
Computational Modeling Practices	<p>(NGSS Practices: Develop &amp; use models; K-12 CS Practices: Creating computational artifacts, Developing &amp; using abstractions, Testing &amp; refining computational artifacts)</p> <p>Develop computational models by specifying model elements &amp; representing their relations and interactions</p> <p>n. Evaluate, test, and debug computational models by determining why the model does or does not appropriately explain or predict the phenomenon</p>

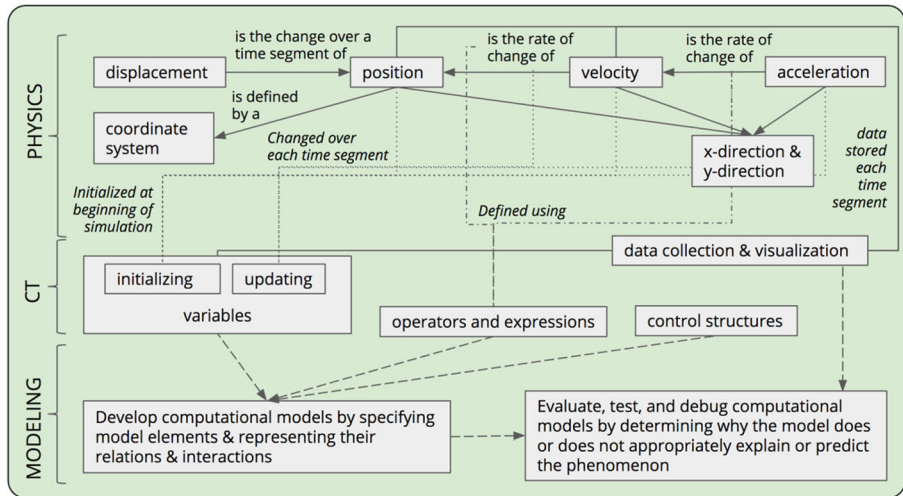


Fig. 3 Step 2 - Integrated Physics and CT domain map

then simulate the model to check, in a *step-by-step* manner, the changes in time to ensure that the modeled objects are generating correct behaviors.

With a comprehensive understanding of the learning boundaries, goals, and content we then developed integrated domain maps to determine the computational modeling blocks that would be needed to match the learning goals while tweaking the structures to take known difficulties into consideration. Fig. 3 illustrates an “Integrated Domain Map” connecting physics constructs (e.g. position, velocity, acceleration, etc.) to relevant CT constructs (e.g. the initialization and updating of variables) and computational modeling practices (e.g., Develop computational models by specifying model elements & representing their relations & interactions). As previously noted, the initial two steps of the DSML design process should result in a structured break down of the STEM and CT concepts and practices targeted, the essential STEM and CT relationships, established levels of abstraction considering the prior knowledge of the target audience, and an evaluation of known domain and CT difficulties that will impact the DSML structures. Addressing these factors better supports the grounding of block semantics, and helps to ensure that although programming applications are simplified, the language still supports application of the key modeling processes defined in Fig. 1.

Following the integration of the domains, Step 3 targets the creation of the DSML blocks based on evidence collected in the first two steps. For example, the integrated domain map in Fig. 3 maps position, velocity, and acceleration to variable initialization and position and velocity to variable updating. As can be seen in the variables in Fig. 4, associated set and change blocks have been created for the Physics constructs. It is important to note that there is currently no block for updating acceleration, but such a block can be easily added to allow for motion models targeting advanced learners.

A properly designed DSML naturally provides a *self-documenting* program in which students can study the simulation results from the developed program to establish its correctness. We hypothesize that this provides an opportunity to improve student understanding of computational constructs (e.g., conditional logic) in context. As an example, a student needs to create a stopping condition for a package dropped from a



Position	Velocity	Acceleration	Simulation Step
		<b>Additional Constructs</b> 	<b>Time</b> 

Fig. 4 Step 3 -Kinematics DSML Implementation

drone at a specific height, so that the package stops moving when it hits the ground (approximation). In order to do so, the student needs to utilize the  $y$ -position variable of the package and ground, compare the values, and stop the motion of the package when the value of the package's  $y$ -position is less than or equal to the  $y$ -position of the ground. By correctly utilizing a DSML and general-purpose blocks created in the BBPE (e.g., an “if” block), a student can model the motion of the object, monitor (observe) its motion, and evaluate if this matches the student's understanding of the motion.

The self-documenting nature of the model and the stopping condition implemented in the model should make this understanding and debugging process easier, thus providing an additional scaffold for novice learners. This design principle incorporates the guidelines and processes of DSML design and development outlined in the “[What Are Domain-Specific Modeling Languages?](#)” section while targeting relevant difficulties presented in the “[Students' Difficulties with Learning-by-Modeling](#)” section.

The DSML designer may include additional blocks to check if students have misconceptions about STEM domain principles. In others words, the flexibility in block creation allows for the instructional decision making warranted in the previous two design process steps, e.g., how constrained should we make the *modeling language* to help students avoid making errors. As system designers, we do not want to over constrain the DSMLs and take away learning opportunities from students.

In C2STEM, students can create blocks to set and change their own custom variables for their modeling tasks. However, we hypothesize that misunderstandings of the STEM domain may interfere with student abilities to create and utilize custom blocks and may negatively impact DSML usage (as noted in Sengupta et al. 2013). For example, unless properly trained on programming language consistency, students may create duplicate programming constructs, especially if they have STEM misunderstandings. Custom blocks and duplication of blocks can lead to confusion in model building (e.g., linking domain principles to blocks) and make the debugging process harder (since the result of executing a block is not clearly specified, or it is based on misunderstandings the student has). We hypothesize that CT difficulties may exacerbate this issue. We demonstrate this in the case studies in a following section.

In the context of creating a DSML using a BBPE, considerations regarding consistency are key. For instance, in Scratch and Snap! users have the ability to create a custom variable and are given blocks to “set” and “change” variable values arbitrarily.

To ensure consistency, variables created for the DSML (e.g., “position in meters” in the kinematics DSML discussed below) must follow a format that is consistent across the different variables introduced. This takes into account the design constraints of the BBPE.

Step 4 involves the implementation of key system structures in the computational modeling environment that utilize the DSMLs to support modeling sub-processes identified in Fig. 1. A key actualization of the “Exploratory Learning of Dynamic Processes” principle lies in the execution semantics of our *Simulation Step* DSML blocks (Fig. 4). Combined with an *Initialization* block that is explicitly used to initialize variable values to the starting state for the simulation (e.g., the object starts with *velocity* = 0 m/s and *position* = 2 m), these blocks incorporate the *step-by-step* generation of simulation behavior, implemented as an infinite loop that updates in a specified time step,  $\Delta t$ . This helps students evaluate and reason about behavior changes in a system as a progression of discrete time-steps as they construct their models. We will discuss this further in the “[Putting it all Together](#)” section.

We reiterate the previously described benefits of the DSML approach, noting that it is important that the visual understanding of the model structure and behavior comes from the self-documenting property of DSMLs. For example, we include data collection support and logging (discussed in Fig. 2) in the form of graphing and table tools that capture the updates in selected DSML variables at each simulation step. The DSML blocks, the visual structure of the code, the animation of the simulation in a *step-by-step* fashion, and the data analysis tools together provide the support that novice learners need to understand the domain and CT concepts and relations. We illustrate this in our case studies that follow.

### Illustrating Applications of the DSML Design Process in Different Domains

As previously noted, we have implemented this design process for three science domains: (1) Kinematics, (2) Marine Biology, and (3) Earth Science. We explore differences in DSML implementations for each domain, by evaluating the choice of abstraction level based on the target audience and curricular objectives in this section. Table 3 presents the core domain analysis considerations for the three domains. Objectives and difficulties included are a few examples found in our design process and used here for the comparisons.

Continuing our Kinematics DSML implementation from the “[Designing Educational DSMLs](#)” section, Fig. 5 provides an example task in the C2STEM computational modeling environment (Section: “[Putting it all Together](#)”). In this task (illustrated in Fig. 5), students simulate the motion of a sloth in the horizontal ( $x$ ) direction given a starting position, velocity and constant acceleration values. Students may initialize the “ $x$ -position” variable using the “set  $x$ -position to [value/expression] m,” the “ $x$ -velocity” variable using the “set  $x$ -velocity to [value/expression] m/s,” and the “ $x$ -acceleration” variable using the “set  $x$ -acceleration to [value/expression] m/s<sup>2</sup>.” To model the motion of the agent, students may use the “change  $x$ -position by [expression/value] m” and the “change  $x$ -velocity by [expression/value] m/s” in a loop that implements the change for each time-step. For the purposes of our kinematics curriculum, students learn to use the “*change  $x$ -position*” block using the expression:  $x\text{-velocity} \times \Delta t$  (a DSML block shown in Fig. 4) implying the update in the  $x$ -position per time step

**Table 3** Domain analysis considerations of each domain

	Physics (Kinematics)	Marine Biology (Coral Reefs)	Earth Science (Water Runoff)
Audience	High School	Middle School	Upper Elementary / Lower Middle
Curricular Objectives	<i>Learning-by-modeling</i> Newton's first and second laws of motion	<i>Learning-by-modeling</i> environmental changes through the examination of increasing ocean temperatures on coral reefs	<i>Learning-by-modeling</i> applied to engineering design by designing a playground considering different surface materials to minimize runoff after a heavy rainfall
Example Difficulties to Consider	<ul style="list-style-type: none"> <li>- lack of prior programming experience</li> <li>- Inability to differentiate between velocity and acceleration</li> </ul>	<ul style="list-style-type: none"> <li>- lack of prior programming experience</li> <li>- Difficulties understanding complex biological processes</li> <li>- Incomplete math knowledge (Algebra 1)</li> </ul>	<ul style="list-style-type: none"> <li>- lack of prior programming experience</li> <li>- Conceptualization of the scenario</li> <li>- Elementary math experience (<i>no algebra experience makes it hard to introduce variables without properly grounding them to explicit domain concepts that students can relate to</i>)</li> <li>- Simulation in time (<i>modeling dynamic processes</i>)</li> </ul>

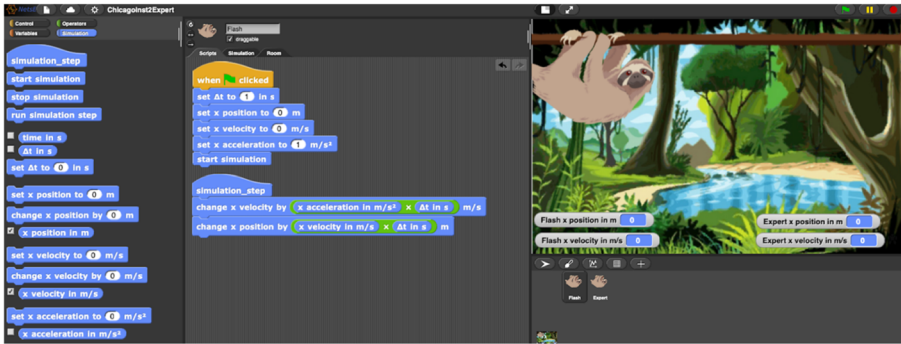


Fig. 5 Step 4 - Physics computational modeling environment

results in:  $x\text{-position}_{\text{new}} = x\text{-position}_{\text{prev}} + x\text{-velocity} \times \Delta t$ . A similar expression applies to “change x-velocity.” The application of this construct using the DSML block in multiple instances helps students understand and link the general principle of how velocity affects position (and, therefore, motion) of an object in a *step-by-step* manner.

The introduction of complex biological processes in Marine Biology represents a transition from Kinematics, where the relations between variables are defined by simple linear relations. In contrast, coral health is impacted by a number of factors that requires complex mathematical relations (nonlinear and they involve multiple interacting factors) to model. Unless these relations are abstracted, middle school students would find it very difficult to understand and model the relations that govern coral bleaching. Therefore, the DSML process needs to systematically create abstractions that simplify the coral’s behavioral processes based on changes in environmental factors but still retain the pertinent science knowledge. Fig. 6 illustrates the integrated domain map we created for the middle school Marine Biology domain. An important consideration was the representation of rates. In our implementation, the teacher we worked with expressed concern about student abilities to model rates, and requested that the

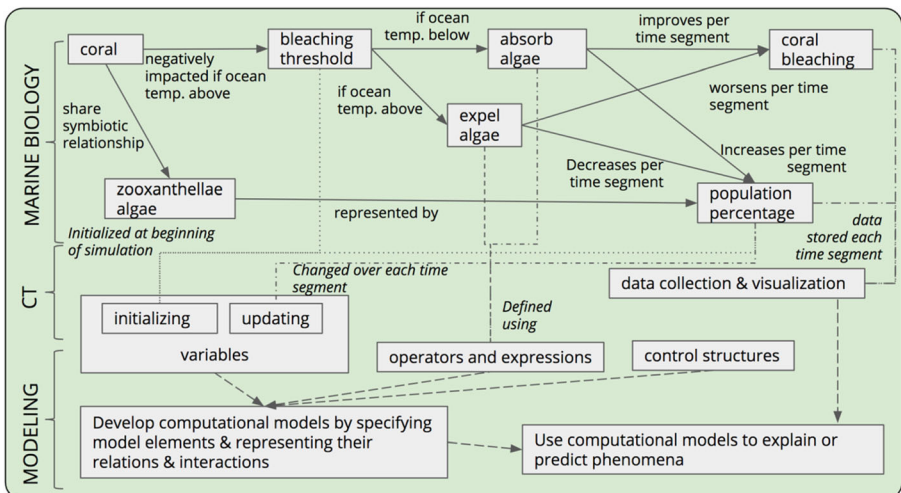


Fig. 6 Marine Biology Integrated Domain Map

curriculum be implemented in multiple steps to help students’ incrementally develop an understanding of rates. Fig. 8 shows how we elected for students to initially express the relations qualitatively (Fig. 8, Marine Biology Part 1) and then progress to a quantitative representation of the bleaching and coral health processes in Part 2 of the curriculum.

The Earth Science DSML is implemented as an NGSS-aligned curriculum unit covering science and engineering for upper elementary or lower middle school students (Chiu et al. 2019; Zhang et al. 2019). The overall unit is designed as a Playground Design Challenge (PDC) with the science focus on the absorption of rainwater and runoff for different surface materials (e.g., concrete, natural grass, poured rubber). Meanwhile, the engineering unit applies the science concepts learned to design a schoolyard with grassy fields, hardtop courts, and play areas that meets cost, runoff, and accessibility constraints. In more detail, students perform the following classroom activities (1) scientific investigations that involve physical experiments on the absorption of different surface materials, (2) build conceptual models to understand the concepts of water absorption and runoff; (3) build computational models to learn and implement the water runoff phenomenon; and (4) solve an engineering design challenge that involves the design of playground models that meet the specified constraints with student-generated computational models. Fig. 7 illustrates the integrated domain map for this Earth Science curriculum.

Since this curriculum unit has been designed for upper elementary and lower middle school students, the computational modeling activity is greatly simplified to match the math proficiencies of average lower middle school students. Therefore, the designed DSML (Fig. 8, under Elementary School Earth Science) abstracts away the concept of rate (e.g., inches/h for the intensity of rainfall), and the notion of *step-by-step* execution to represent the dynamics of system behavior. Instead, students update the total amount of rainfall by a fixed hourly amount × duration of rainfall. Similarly, the absorption of a material is expressed by an absorption limit for the surface material. If the total rainfall is less than or equal to the absorption limit, then the total absorption (in inches) will be

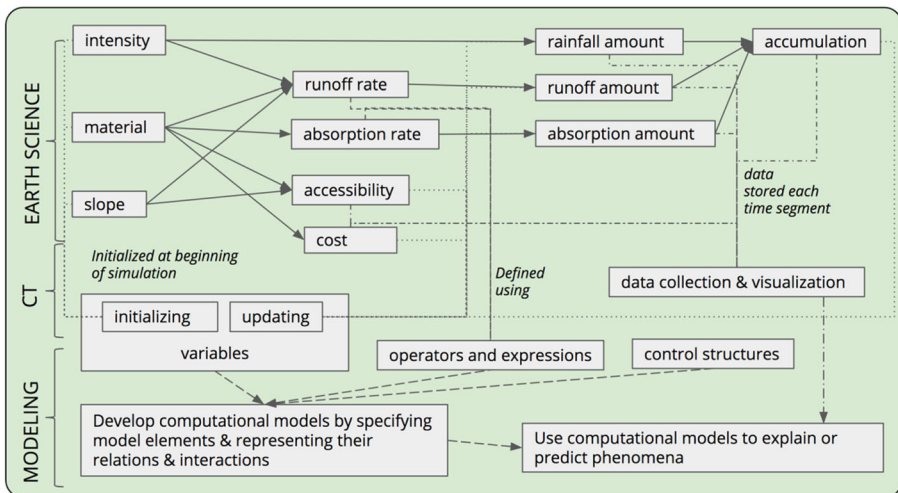


Fig. 7 Earth Science Integrated Domain Map


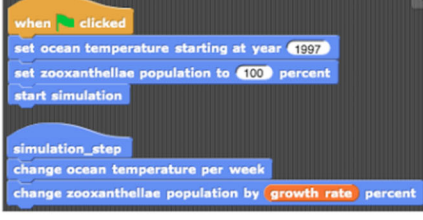
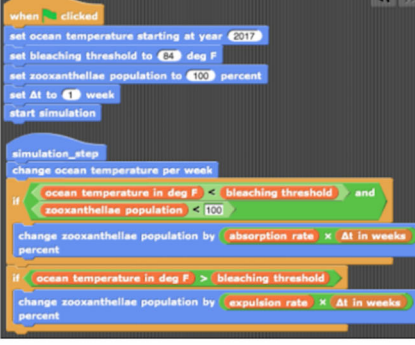
High School Physics	Elementary School Earth Science
 <pre> when clicked   set Δt to 1 in s   set x position to 0 m   set x velocity to 0 m/s   set x acceleration to 1 m/s²   start simulation  simulation_step   change x velocity by x acceleration in m/s² × Δt in s m/s   change x position by x velocity in m/s × Δt in s m </pre>	 <pre> when clicked   set total rainfall (inch) to 0   set elapsed time (hour) to 0   set hourly rainfall (inch) to 0   set rain duration (hour) to 0   set total absorption (inch) to 0   set total runoff (inch) to 0   start raining  repeat until   elapsed time (hour) is equal to rain duration (hour)   change total rainfall (inch) by adding hourly rainfall (inch)   change total absorption (inch) by adding hourly rainfall (inch) × absorption limit   set total runoff (inch) to total rainfall (inch) - total absorption (inch)   update simulation history   stop raining </pre>
Middle School Biology, Part 1	Middle School Biology, Part 2; High School Biology
 <pre> when clicked   set ocean temperature starting at year 1997   set zooxanthellae population to 100 percent   start simulation  simulation_step   change ocean temperature per week   change zooxanthellae population by growth rate percent </pre>	 <pre> when clicked   set ocean temperature starting at year 2017   set bleaching threshold to 84 deg F   set zooxanthellae population to 100 percent   set Δt to 1 week   start simulation  simulation_step   change ocean temperature per week   if ocean temperature in deg F &lt; bleaching threshold and zooxanthellae population &lt; 100     change zooxanthellae population by absorption rate × Δt in weeks percent   if ocean temperature in deg F &gt; bleaching threshold     change zooxanthellae population by expulsion rate × Δt in weeks percent </pre>

Fig. 8 Comparison of DSMLs based on different scaffolding requirements

set to the total rainfall (in inches), and runoff (in inches) will be set to 0. Otherwise, the total absorption will be set to absorption limit (in inches) and runoff (in inches) will be set to total rainfall (in inches) – total absorption (in inches). To implement the runoff model, a student first designates the input variables (intensity and duration of the rain) and chooses the material being studied. In the body of the model, the student computes the total rainfall, the total absorption by the material, and the runoff, if any. The model helps students to determine how the absorption and runoff vary for different materials, and how this also depends on the hourly amount and the duration of the rainfall. The Earth Science DSML offers an example where the DSML blocks and the simulation model itself is highly abstracted to match the grade level math and science proficiency of the students. In future versions, the unit can be easily extended for upper middle school students by introducing the concept of varying rates, for example, the varying intensity of rainfall over a period of time and the changing absorption rate as the surface material becomes saturated.

Fig. 8 presents the final “Delivery” process for each domain. The Kinematics DSML represents the most detail (least amount of abstraction) with the goal of helping students



translate the kinematic equations to a discrete-time computational form, and study the relations between acceleration, velocity, and position for different scenarios. Our Marine Biology unit for middle school students, uses abstraction to remove the functional and temporal complexities of the biological processes linked to coral bleaching and health. Instead, students focus on the symbiotic relationship between the coral and zooxanthellae algae. Part 1 of the Marine Biology implementation (Fig. 8), illustrates the initial, qualitative representation of the biological process and Part 2 illustrates the more advanced approach taking into account rates, discussed above. The complexities of the relations are implemented under the hood, but the results of the outcome variables, algae levels and amount of bleaching are depicted through animations that facilitate an understanding of the processes involved. Finally, the Earth Science application, designed for an upper elementary school curriculum, represents the highest level of domain and computational abstraction for the reasons discussed above. These results demonstrate differences in the DSMLs implemented for the different domains and age levels, with careful consideration of multiple factors for each step of the DSML process. This section addresses our first research question that we stated in the “Introduction” of the paper.

### Putting it all Together: The C2STEM Environment for Computational Modeling

C2STEM targets the building of simulation models, a specific form of computational model for expressing the dynamic behavior of a system. We adopted a conventional approach to scaffold the model-building task as an initialization component (under the “Green Flag”) and a simulation component (indicated as the “Simulation Step”) (see Fig. 9) as a means of targeting our “Exploratory Learning of Dynamic Processes” design principle. When the green flag is clicked (either via the green flag button on the top-right of Fig. 9 or by double-clicking the “When [green flag] clicked” block), the C2STEM environment first executes the set blocks under the “When [green flag] clicked” (Initialization phase) until it reaches the “start simulation” block. Students are required to utilize the “start simulation” block in order to initiate the execution of code under the “Simulation Step.” This code generates the dynamic behavior of the system. We avoid over constraining in C2STEM by making students think about the set of variables they need for building their models. Operators and expressions supporting the use of mathematical constructs to represent the relations between variables, and the expressions associated with conditional control structures, such as the “Green Flag”



Fig. 9 Example model building task in C2STEM

(utilized in most BBPEs) and those listed under “Simulation Step” in Fig. 4 are applicable across domains. In terms of future DSML development, the use of these general-purpose blocks (e.g., “Green Flag,” operators, “Simulation Step”) provides consistency of the format of generic computational constructs in the model building environment, making it easier for students to transition from one domain to the next.

The block code under the “Simulation Step,” runs in an infinite loop until a stopping condition is satisfied. All of the instructions under the “Simulation Step” are executed in one time-step,  $\Delta t$  (if the student does not set a value for  $\Delta t$ , it defaults to a value of 0.03 s). In previous work, we have found this temporal *step-by-step* approach helps students map the fundamental laws of the domain to the observed dynamics in the system behavior (Hutchins et al. 2020).

Along with the animation and variable inspection functionality on the stage, students have access to multiple representations for observing model behaviors. These include graphing and table generation tools that are updated dynamically at each simulation step (Fig. 9). Students can analyze the graphs and tables to understand the relevant physics concepts and laws, e.g., the relationship between velocity and acceleration, and make predictions about object behaviors, e.g., how vehicle position changes as it speeds up, cruises, and slows down. In combination with the self-documenting nature of DSMLs, students can use these features to assess the correctness of their models. From the CT perspective, these tools allow for applications of data analysis and debugging, key CT practices (Grover and Pea 2018; Weintrop et al. 2016). This directly maps to multiple NGSS standards, such as planning and carrying out investigations, using mathematics and computational thinking, analyzing and interpreting data, and constructing explanations and solutions. We believe that the NGSS standards addressed provide additional support for the simultaneous learning of STEM and CT because students have to represent their conceptual STEM knowledge as computational constructs to build their models and to interpret and analyze their model behaviors. Such opportunities are often lacking in traditional science classroom environments.

Finally, we have introduced a “stop simulation” block to provide a mechanism for the students to explicitly control the end of a simulation run. Students typically use a conditional statement to express the terminating conditions for the simulation (Hutchins et al. 2020; Snyder et al. 2019). Overall, this construct also increases the readability and interpretability of the students’ constructed models.

In summary, applications of our DSML design process in Physics, Marine Biology, and Earth Science demonstrate how the design process supports the implementation of language abstractions linked to students’ educational levels and the significance of our design principle ECD for driving the choice of variables and computational constructs associated with the DSML programming blocks. As the Physics example in the “[Designing Educational DSMLs](#)” section illustrates, our design process allows for a systematic approach to define the intersections of the STEM domain and CT to ensure consideration and applicability of each subject throughout the development of the DSML. We extend our evaluation of the design process, including our design principles of ECD, exploratory learning of dynamic processes, and DSML development using block-based programming languages with case study analyses for each science domain.

## Results and Analysis

We address our second research question, “*how do the DSML-created abstractions support learning of STEM and CT?*” by analyzing data from case studies that demonstrate how DSMLs support key modeling processes, including: (1) translating STEM knowledge into computational forms, (2) interpreting model behavior using evidence, and (3) debugging computational models. In addition, we discuss the role of DSMLs in grounding the semantics of *modeling language* constructs to support classroom discussions and STEM learning. To preserve privacy, all participants, whose work is described in this paper, have been anonymized. We evaluate student learning utilizing summative and formative assessments (for representative formative assessments and instructional tasks, please see [expert.c2stem.org](http://expert.c2stem.org)), analyzing modeling solutions and screen capture recordings.

## Methods and Data Sources

We used data from three different studies to perform qualitative and descriptive analyses to answer our research question on student learning. Table 4 illustrates the study parameters for each domain, grade level, number of participants ( $n$ ), duration, training period in the environment, and the different forms of data collected in the environment.

The Physics study used an experimental format with control and experimental groups. Both groups attended lecture classes. The experimental group worked on the C2STEM environment as a replacement for typical lab activities, while the control group performed traditional lab activities. The study ran for a full semester in a high school honors Physics class and covered topics in kinematics and mechanics. Students did not use the system every day (per class schedules), and the study reported here covered three kinematics units (1-D acceleration, 2-D constant velocity, and 2-D acceleration). The study is described in detail in Hutchins et al. (2020). Training time mainly involved an introduction to core Physics and CT concepts and practices to be utilized during the study, with little direct training on the *modeling language*. The DSML blocks and example modeling tasks are illustrated in the “[Establishing an Educational DSML Design Process](#)” section, above. The purpose of this analysis is to demonstrate how DSML blocks facilitate student learning, and also support more detailed understanding of students’ model building approaches.

**Table 4** Overview of study design for different domain implementations

Domain	Grade	$n$	Duration	Training	Data Collected
Physics	10	68	1 semester	5 h	Summative, formative assessments; OBST <sup>TM</sup> screen capture recordings; Classroom video recordings;
Marine Biology	8	34	1 week	1 h*	Summative, formative assessments; Classroom video recordings; Task files and log data
Earth Science	6	99	1 week	1 h	Summative assessments; Task Files and log data

Summative and formative assessments for all studies were generated using the ECD approach (Mislevy and Haertel 2006). Some assessment items were adapted from other studies (Basu et al. 2018; Hestenes et al. 1992; Grover 2019, 2020; McElhaney et al. 2019). Representative CT summative assessment items administered across domains are illustrated in Appendix A. The Physics curriculum was created by our Physics domain expert in conjunction with our high school teacher in whose classes the study was run. For Marine Biology, a subject matter expert helped create the curriculum and the assessments. A Preparation for Future Learning assessment (Schwartz et al. 2005) was developed and administered at the conclusion of the experimental Physics study to evaluate student abilities to transfer problem-solving approaches to new domains. This paper discusses some of the salient results, but a more detailed analysis of student performance can be found in (Hutchins et al. 2020).

Two sessions of student work were video recorded using the OBS™ screen capture system to gain a better understanding of their model building approach and the challenges they faced during model building. For each task, we noted key actions and conversations that highlighted synergistic learning events. The Physics and Marine Biology classroom sessions were also video-recorded to examine the teacher's pedagogical methods and to evaluate how our tool supported instruction and learning.

Students' model building work was captured in log files with timestamps. Each action (e.g., the adding of a DSML block to the "script" stage or a partially developed program, see Fig. 9) was logged using (1) an action name, (2) the DSML block name, (3) the associated block (e.g., inserting a multiplication operator into a "change  $x$ -position by [value/expression] in  $m$ " block would add information about the parent block for the multiplication operator) or change to the value of a block (e.g., changing the starting value of "set  $x$ -position to [value]  $m$ " from 10 to 0), and (4) a timestamp.

In the Marine Biology study, upper middle school students worked on model building activities in two domains. First, they worked on a 1-week Physics curriculum on 1-D acceleration (a modified subset of the high school curriculum) that utilized the Physics DSML outlined in the "Illustrating Applications of the DSML Design Process in Different Domains" section. After a three-month gap, students performed model building activities for one week in the Marine Biology unit.

The Earth Science Unit, developed by a research team from SRI, University of Virginia, and Vanderbilt University, was aligned with a set of NGSS performance expectations (e.g., 5-ESS3-1, 3-5ETS1-1, 305ETS1-2, etc.; NGSS Lead States, 2013). In the curricular unit, the students worked on resolving an engineering design challenge to prevent flooding of a school playground during heavy rains. The disciplinary core idea for the unit focused on the absorption and runoff of water for different surface materials, and this core idea was then applied to generating and revising designs of the playground. A total of 99 6th grade students from a middle school in the Southeastern United States participated in the 4-week study. During the study, the students spent about an hour each day conducting engineering design activities with the Earth Science DSML for 6 days.

## Impact on Student Learning and Performance

To demonstrate the impact of DSMLs on student learning and performance, our analyses target key modeling subprocesses illustrated in Fig. 1, including the student difficulties in understanding and modeling these processes (Section: "Students'

**Table 5** Overview of the impact of DSMLs on student learning and performance

Domain	Modeling Difficulties Targeted	Hypothesized DSML Benefits
Physics	Translating learnt domain knowledge into computational forms for model building; Debugging the behaviors (results) generated by the abstract representations and interpreting them in terms of scientific principles and theories	Modularized programming constructs with automatic animation of behaviors when running code (Section: “ <a href="#">How DSMLs Support Conceptual Understanding of the STEM Domain through Modeling</a> ”); Contextualized representations of key domain and CT constructs (Sections: “ <a href="#">How DSMLs Support Conceptual Understanding of the STEM Domain through Modeling</a> ”); Ability to link constructs to observed model behaviors (Section: “ <a href="#">How DSMLs Support Debugging</a> ”)
Marine Biology	Translating domain knowledge learned by interpreting data into computational forms for model building; Understanding the mathematical relations between variables and interpreting graphs in relation to generated simulation behaviors	Modularized programming constructs with automatic animation of behavior when running code (Section: “ <a href="#">How DSMLs Support Building Models from Behavior Data</a> ”); Contextualized representations of domain behaviors with CT constructs (Section: “ <a href="#">How DSMLs Support Building Models from Behavior Data</a> ”);
Earth Science	Translating learnt domain knowledge into computational forms for model building	Modularized programming constructs with automatic visual representation of results; Contextualized representations of domain behaviors with CT constructs (Section: “ <a href="#">How DSMLs Can Support Computational Learning and Problem-Solving at Younger Grade Levels</a> ”)

Difficulties with *Learning-by-Modeling*). Table 5 provides an overview of students’ modeling difficulties observed in our analyses, and the hypothesized impact of the DSML approach in helping students overcome these difficulties. Sections “[How DSMLs Support Conceptual Understanding of the STEM Domain through Modeling](#)” and “[How DSMLs Support Building Models from Behavior Data](#)” focus on students’ model building processes. In the “[How DSMLs Support Conceptual Understanding of the STEM Domain through Modeling](#)” section we show how the DSML constructs help students map from domain principles to the appropriate DSML constructs in Physics, whereas in the “[How DSMLs Support Building Models from Behavior Data](#)” section we demonstrate how students interpret their understanding of historical data into relevant DSML constructs to build their models. The section “[How DSMLs Can Support Computational Learning and Problem-Solving at Younger Grade Levels](#)” demonstrates a simplified DSML to help lower middle school students develop runoff models for an earth sciences curriculum. Finally, the “[How DSMLs Support Debugging](#)” section targets debugging processes. We use contrasting cases of successful and unsuccessful students to illustrate the effectiveness of the DSML constructs.

As a baseline evaluation, we analyzed student summative performance in each domain. Overall, the students in each study demonstrated significant learning gains

in STEM and CT. These results further substantiate previous results derived from learning-by-modeling systems for STEM education.

- A *t*-test on the difference in normalized learning gains in both kinematics ( $p = 0.01$ , Cohen's  $d = 1.593$ ) and CT ( $p = 0.008$ , Cohen's  $d = 0.803$ ) was significant between the experimental ( $n = 34$ ) and control ( $n = 34$ ) groups in the high school Physics study. ANCOVA results indicated a significant effect of group on posttest performance controlling for the pre-test scores in kinematics [ $F(1, 65) = 6.748, p = 0.012$ ] and in CT [ $F(1, 65) = 4.801, p = 0.032$ ].
- In the upper middle school Physics + Marine Biology study, pre-post tests conducted during the physics unit showed significant learning gains in Physics ( $p = 0.009$ , Cohen's  $d = 0.654$ ) and CT ( $p = 0.0001$ , Cohen's  $d = 0.977$ ). Similarly, the pre-post test results for the Marine Biology unit produced significant learning gains in Marine Biology ( $p = 0.00002$ , Cohen's  $d = 0.969$ ) and CT ( $p = 0.01$ , Cohen's  $d = 0.616$ ).
- In the lower middle school Earth Science study, students took the pre-post tests that included science, engineering, and CT items. The average pre-test score for the science section was 4.56 ( $SD = 1.03$ ) out of a maximum of 7 points; the average pre-test score for the engineering section was 8.73 ( $SD = 2.62$ ) out of a maximum of 16 points; and the average pre-test score for the CT section was 6.23 ( $SD = 2.60$ ) out of a maximum of 13 points. Their corresponding post-test scores were 5.13 ( $SD = 1.04$ ), 10.50 ( $SD = 2.67$ ), and 8.41 ( $SD = 2.69$ ) points, respectively. The *t*-test results indicated that the learning gains in all three parts were statistically significant ( $p < 0.001$ ,  $p < 0.0001$ , and  $p < 0.0001$ ) with Cohen's  $d$  effect sizes of 0.54, 0.67, and 0.83, respectively.

All three studies clearly indicate that the DSMLs + the C2STEM environment helped students learn both the science domain and CT content. The high school physics study further illustrates that working in the C2STEM environment produced better learning gains than students who learned their physics purely through classroom instruction and homework exercises. This indirectly provides support for RQ2.

### How DSMLs Support Conceptual Understanding of the STEM Domain through Modeling

To understand how the DSML constructs may or may not have helped students in building their models, we take a deeper dive and look at the model construction processes for a few students, and link them to their pre-posttest performance. Two pre-posttest questions targeted students' understanding of the impact of gravity on the motion of an object. One question required students to choose the plot that correctly depicted the motion of a ball in time during free fall under gravity. This question tested the students' understanding of how gravity (constant acceleration) affected the *step-by-step* motion of the ball. The second question was adapted from Hestenes et al. (1992) and tested students' ability to analyze information about speed and range, and get them to compare the duration of flight for two cannon balls. StudentA improved from pre to post on both questions. StudentB did not improve, answering the second question



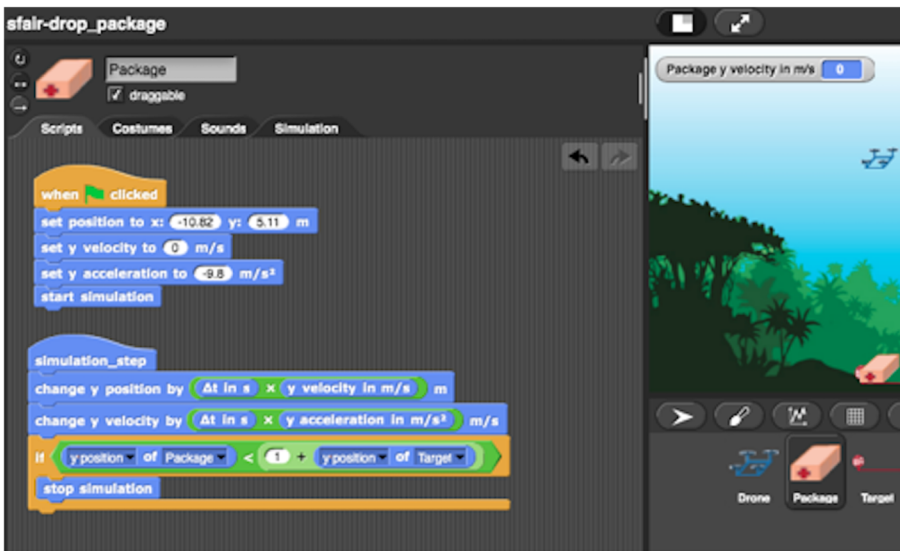


Fig. 10 StudentA's 2D with gravity instructional task solution

incorrectly on the post-test. We hypothesize that the model-building task impacted their understanding of free fall under gravity.

Fig. 10 shows StudentA's code for a model-building task to generate the trajectory for an object dropped in mid-air from a drone. The figure shows that StudentA initialized position ( $x$  and  $y$ ),  $y$ -velocity, and  $y$ -acceleration, with  $y$ -acceleration hardcoded to the value of the gravitational constant (instead of using the provided  $y$ -gravity DSML block) as part of her free-fall model. Under the *simulation step* flag, she used the correct DSML blocks with the correct expressions for updating the  $y$ -position and the  $y$ -velocity of the package. Therefore, she demonstrated the correct use of physics equations of motion in her computational model. On the Physics summative assessment, she improved from 25 to 34 out of 40.

However, StudentA had issues with the use of CT constructs in her solution. She did not set a value for the simulation step,  $\Delta t$ , so the system used the default value of 0.03 s). Second, she modeled the stopping condition, i.e., stop motion of the package when its  $y$ -position is less than 1 + the  $y$ -position of the Target. It is clear that she could have expressed her stopping condition more accurately, rather than choosing an arbitrary expression of 1 +  $y$ -position of the target. Third, her model constructs were written in the order where the  $y$ -position of the object was updated before the  $y$ -velocity. Since the initial  $y$ -velocity was 0, the package's  $y$ -position did not change till time step 2, this caused a small error in her solution. These issues were evident in her post-test. While she did increase from a CT score of 22 to 30.5 out of 37 on the two summative assessments, she struggled with questions on debugging of a loop and correctly updating a variable based on a sequence of prior actions.

StudentA also selected the value of the Package's " $y$  velocity in m/s" to be shown on the stage (see Fig. 10). She could view the change in  $y$ -velocity over time using the data tools. Overall, this provides circumstantial evidence (pre-post-test gain plus the ability to build the almost-correct physics model) that the Physics DSML structures helped



Fig. 11 StudentB's 2D with gravity instructional task solution

StudentA learn the domain concepts of *position*, *velocity*, and *acceleration*, and the visual structure helped her organize the constructs to generate an almost correct model. Her viewing of the *y*-velocity of the package during the simulation also helped her understand how that velocity changed as time advanced.

Fig. 11 provides the final code for StudentB's model for the package drop problem. He elected to use custom blocks for "*update velocity in m/s*" and "*update position in m*" in addition to using the DSML blocks "*change y position in m*" or "*change y velocity in m/s*" as done by StudentA (Fig. 10). Custom blocks are treated as a function, and other blocks can be inserted and connected together to represent the functionality of the custom block. When StudentB executed his model, assuming it would work correctly, he was surprised by the results (saying aloud "*I don't understand this*"). He opened the graphing tool, ran the simulation numerous times, and realized that the position was decreasing with a negative linear slope (Fig. 11). StudentB was aware what the correct position-time plot should look like, and even said "*we need to get it to where it curves though*," to a student sitting beside him. At this point, StudentB also asked to plot the "*y velocity in m/s*," and saw the velocity-time plot was horizontal. He then opened the "*update velocity in m/s*" custom block and made a correction to the incorrect expression for the update *y* velocity, but he set his corrected updated equation to  $\Delta t$  times *y*-velocity (and not *y*-acceleration).

Therefore, his model was still incorrect, and he was unable to make further corrections though he understood what the final behavior should be. In this case, we hypothesize that, if he had used the DSML blocks directly instead of the custom blocks, the

self-documenting nature of the DSML blocks would have made it easier for him to overcome his difficulties in understanding his domain errors, and correcting the computational expression for *update velocity* variable. As hypothesized above, the lack of simplicity in the custom block implementation made it difficult to identify the location of the error. Moreover, the incorrect use of the DSML blocks may have exacerbated the common Physics misunderstanding between velocity and acceleration, described in the “[Evidence-Centered Design](#)” section. The DSML blocks would have provided a direct mapping between DSML constructs “*change y velocity by [expression] m/s*” and “*change y position by [expression] m*” (in each case, the expression term would correspond to a relevant equation of motion) and the *step-by-step* change in *y*-velocity and *y*-position as observed in the graph. StudentB eventually dragged the “*change y velocity by [value] m/s*” block into his model code, but did not replace his custom code with the update velocity expression using the block. The inability to use the DSML blocks affected the students’ ability to correctly model the physics and CT constructs and generate the correct model even though he realized where he was making errors.

The difficulties StudentB experienced in the modeling task are also reflected in his pre-test and post-test performance. In Physics, StudentB went from a score of 26 to 36 out of 40 (above experimental group average on both), demonstrating gains in Physics; in this case as he knew how the object should be moving. However, on the summative assessment questions described above, his scores decreased. We hypothesize this was due to his inability to interpret and use the appropriate DSML blocks, thus resulting in confusion about the correct form of the equations of motion. In CT, StudentB had a lower than average score on the pre- and post-tests (22.5 and 25 out of 37, respectively).

### How DSMLs Support Building Models from Behavior Data

This case study illustrates how DSMLs support the advancing of students’ skills in data analysis and reasoning. In order to analyze students understanding of DSML structures and the ability to link the DSML constructs to evidence in the data tools, we evaluated:

1. Pre-post results along with students’ performance on a Marine Biology modeling task. Modeling task performance includes logged action data indicating if data tools were utilized during the modeling task (see Section: “[Methods and Data Sources](#)”).
2. Student answers to a word problem targeting their understanding of conditional logic structures in the summative assessment. The summative assessment question on growth rate asked students: “*Marine biologists are collecting data on brain coral in the Caribbean, a region, where we have been having abnormally high ocean temperatures in the summer. The table below shows the year and the average width of the coral for that year. Coral width is a measure of coral health. We have data from 2008 to 2017. In which years did the coral most likely experience temperatures that were too high for a healthy growth rate? Explain.*” The table listed three years in which the coral’s growth was 1 cm, whereas the remaining years showed a growth of 2 cm per year. This task was scored out of 3 points.
3. Student responses to an embedded assessment that focused on their knowledge of the growth rate of corals under different environmental conditions (primarily ocean temperatures). The embedded assessment question: “*A marine biologist*

*hypothesizes that a brain coral has been impacted by high temperatures. From 2010–2015, the coral grew by 2 cm a year and now had a width of 152 cm. In 2018, she measured the coral’s width and it was 156 cm. Do you agree or disagree with the marine biologist’s measured width? Why or why not?”* asked students to answer a question and provide an explanation.

For the modeling task, the students were tasked to study the cumulative effect on coral health in a region where the ocean temperatures (provided in a data set) went above the bleaching threshold for periods of time. Students constructed a computational model of coral health, using the ocean temperature at each time step of the simulation to determine if the coral would expel or absorb zooxanthellae algae (these algae give the corals their colors). They programmed the coral survival based on the zooxanthellae population percentage present in the coral as a function of time. The ocean temperatures came from data collected by a research station along the Great Barrier Reef during a major bleaching event that happened in 2016–17.

Results indicate that students who received high scores on the post-test, both overall and the individual growth rate question, (1) were able to generate a correct model for the modeling task, (2) utilized the data tools to support model building and problem solving, and (3) correctly answered the embedded assessment question, referencing provided data (with correct variable naming) to do so. For instance, StudentD scored a 0 out of 3 on the pretest growth rate question and improved to a 3 out of 3. StudentD’s modeling task solution (shown in Fig. 12a) correctly initialized the needed variables, and she set the coral’s growth rate after checking the ocean temperature at each simulation step by comparing against the coral’s bleaching threshold. The simulation changed the width of the coral using the newly set growth value. StudentD’s action data also indicated data tool use and she answered the task question correctly. For the embedded assessment following Task B, StudentD answered the Marine Biologist word problem with *“I agree with the marine biologist that a piece of brain coral has been impacted by high temperatures because between the years of 2010–2015 the piece of brain coral grew at a constant rate of 2cm per year, but between the years of 2015–2018 the brain of the coral only grew at 1.3cm per year.”* This detailed response demonstrates an understanding of rates, correct identification of the time intervals in

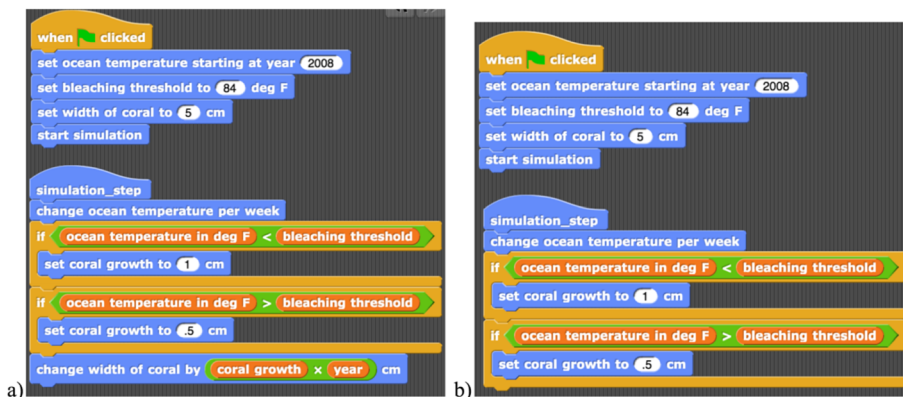


Fig. 12 StudentD (a) and StudentE (b) solutions for the Marine Biology Tasks

which the growth of the coral showed abnormal values, and the ability to utilize data to generate answers to questions and provide explanations. StudentD's model building actions in the Marine Biology unit indicates a positive benefit of DSMLs that helped her focus on the relevant relations between temperature and bleaching, thus creating a bridge for her to link the data provided to the visualization of the coral bleaching. This understanding also translated to her being able to explain the processes involved in a clear and concise manner.

Students who did not perform well on the post-test were either unable to generate the correct model code (e.g., incorrectly updating the coral growth and width variables or creating a separate custom block that was improperly initialized) or, if they were able to do so, did not utilize the data tools to help visualize and interpret the results. For example, StudentE's solution for the modeling task is provided in Fig. 12b. StudentE initialized the required variables correctly, using the appropriate DSML blocks. In addition, he included the change block to correctly update the ocean temperature values in each simulation step, as specified in the data set. He correctly set the coral growth rate by checking whether the current ocean temperature data was above or below the coral's bleaching threshold. Each of these steps indicated an understanding of STEM domain concepts. However, StudentE did not include the block to change the width of the coral, and answered "year 8" for the task prompt, indicating a random or uninformed guess as neither the visualization nor the data needed to make the decisions updated correctly. It is important to note that in looking at his action data, StudentE did not understand the modeling task, and did not utilize the data tools for either task. He did not seem to comprehend the *step-by-step* update approach for the simulation. Therefore, his inability to understand the DSML blocks and the simulation step constructs affected his model building abilities. Finally, in the embedded assessment following task B, which asked whether the student agreed with the marine biologist's hypothesis about the effect of rising ocean temperatures on corals, StudentE's answer "*No because it is a consistent rate and the pattern doesn't follow it*" again showed his lack of understanding. This response was hard to interpret (the rate was not consistent), and he did not refer to the data to support his explanation. His inability to use the correct DSML constructs to build his model and observe the model results, also indicates that he did not learn from the intervention, which contributed to his lack of understanding. On the posttest, StudentE was unable to answer the question about coral reef data correctly. StudentE did improve on the use of constructs, such as the impact of increasing ocean temperatures on the relationship between corals and the zooxanthellae algae (a topic covered in Task 1 when a growth rate variable was given to the students). We hypothesize that additional support in the use of the DSML blocks, the simulation step construct, and the data tools may have helped StudentE learn in similar ways that StudentD was able to learn her Marine Biology and CT constructs.

Finally, of the students that improved from a 0 to 3 out of 3 on the growth rate summative assessment question, one student did indicate difficulties on the modeling task, primarily with conditional logic. This student utilized the appropriate set and change blocks, but appeared to have difficulties in implementing the conditional statements. The student wrote the condition as "*if (ocean temperature in deg F = bleaching threshold) → set coral growth*". Since the equality condition may not have ever been satisfied (students needed to use a > or < operator), the model did not produce the correct behaviors. This student seemed to struggle with debugging this block using

the visual feedback provided by the C2STEM environment. She had difficulties with conditional logic on the CT post-test as well. In such cases, where students have difficulties with their CT constructs, as opposed to their domain concepts, we will have to determine appropriate CT feedback to support these students' model building processes. For instance, a suggestion could be given to evaluate data or advice on methods of data evaluation if a student attempts to run the simulation more than 1 time without opening a data tool. In addition, multiple attempts to run a simulation without the use of the "change coral width" block could be an indication that feedback is needed on possible reasons why the coral width is not changing. This type of feedback supports Step 4 of the DSML design process, including the development of system structures based on DSML usage to support model building processes.

### How DSMLs Can Support Computational Learning and Problem-Solving at Younger Grade Levels

As discussed in the "Illustrating Applications of the DSML Design Process in Different Domains" section, our DSML approach instantiated in the Earth Science unit supported the implementation of a curricular unit designed for upper elementary and lower middle school students (Chiu et al. 2019; Zhang et al. 2019, Zhang et al., 2020). The computational modeling activity is simplified to match the math proficiencies of average lower middle school students (absorption and runoff related to different surface materials). In a recent classroom implementation, 59% of the 99 sixth-grade students created a correct computational model of the runoff system before the answer was shown to them (Zhang et al., 2020). In addition, students' modeling performance using the DSML significantly contributed to their performance in the succeeding learning activity where they designed playground models that meet specified engineering criteria of minimizing runoff while maintaining the cost and the accessibility requirements. Our recent findings showed that those who build better computational models ended up creating more satisfying playground designs that caused less runoff to the neighboring area of the school (Zhang et al. 2020). More specifically, we found that the students' performances in computational model building and engineering design had a moderate but statistically significant correlation (Pearson's  $r = 0.29$ ).

### How DSMLs Support Debugging

Debugging represents an important overlapping practice between science modeling and CT, especially because students are required to translate their understanding of physics into computational constructs to build their models. As an analysis question: *how do the DSMLs support computational modeling tasks in STEM?* we hypothesized that the DSML constructs would support the debugging process. Again, we present a contrasting example, where StudentC was successful, and StudentB (the same student from the last set of Physics examples) was not successful in combining the use of DSML constructs and data analysis tools to correct errors in their models. In this case, the students



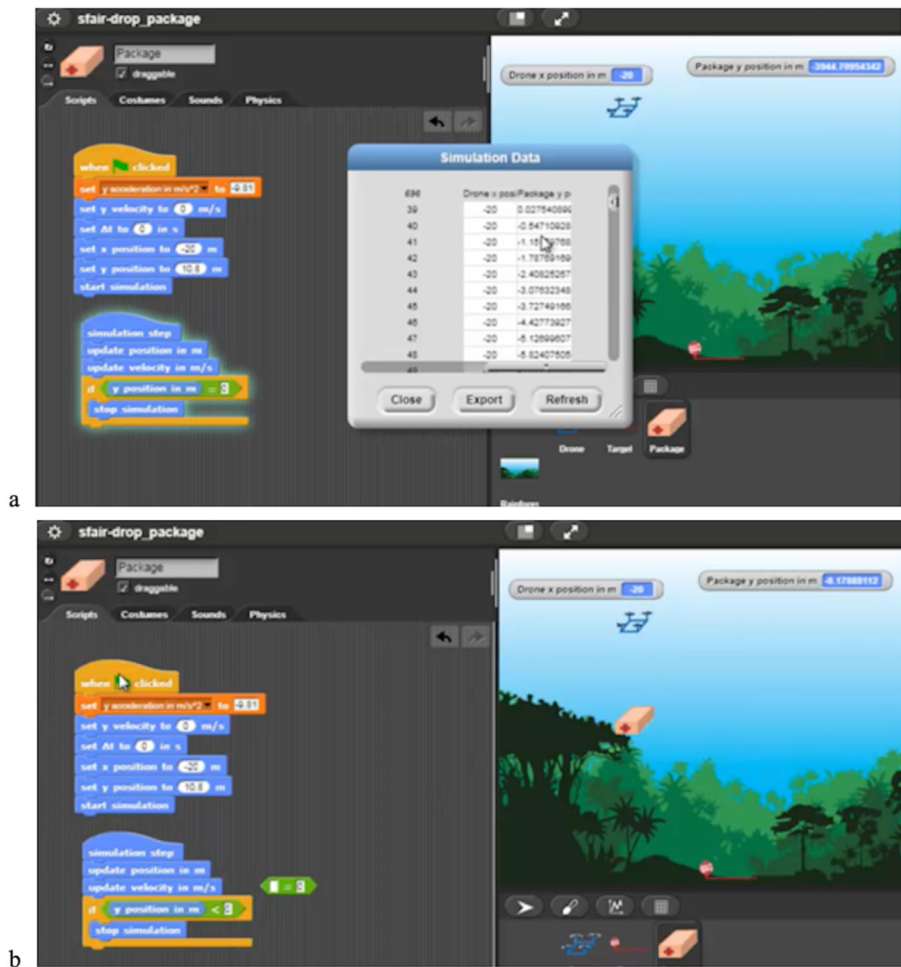


Fig. 13 StudentC's correct use of table tool



Fig. 14 Incorrect use of graph tool

were working on modeling a package drop from a moving drone, and the challenge was to land the package at a specific location on the ground.

In the first scenario, StudentC programmed the stopping condition as stop simulation when  $y$ -position of the package is equal to 0 (i.e., the package is at the target). The student executed her model multiple times, and each time, the package did not stop when it hit the ground ( $y = 0$ ), but continued to move downward. The student then opened the table and ran the program (see Part A of Fig. 13). As can be seen in the figure, the student's cursor is located around the  $y$ -position data. The table shows that the  $y$ -position of the package is never equal to 0. The student then proceeded to change the condition so that the simulation stopped as soon as the  $y$ -position of the target was less than 0 (Part B of Fig. 13). StudentC recognized that her conditional logic was not working as expected, and was able to debug the issue utilizing Physics data generated by her simulation to correct her code and make the model work correctly. The 1–1 correspondence between the DSML blocks and *step-by-step* execution made it easier for her to detect and correct her problem.

On the other hand, StudentB's example (see the “How DSMLs Support Conceptual Understanding of the STEM Domain through Modeling” section for the description for the first time point shown in Fig. 11; and a later time point is shown in Fig. 14) represents a case where the use of custom blocks in addition to DSML constructs affected his ability to correct his model, even though he realized his error, and its possible cause.

### DSMLs Support *Learning-by-Modeling* in STEM Classrooms

An important contribution of DSMLs to STEM classrooms is how the modeling representations, the behavior generation, and the visualization of behaviors support instruction and classroom learning. We target the previously described difficulty of developing a shared understanding of the *modeling language* used to support communication of model behavior or bugs through a case study on how the classroom teacher used the DSMLs to communicate the underlying STEM domain principles to his class. In this case study, the classroom instructor specifically utilized the DSMLs along with the system tools to link the *step-by-step* modeling approach and the kinematic equations in a 1-D acceleration module. The students modeled a truck speeding up from rest, then cruising at the speed limit, and finally coming to rest at a stop sign. Students were having difficulties in linking the traditional kinematics equations to the DSML model constructs for *step-by-step* modeling.

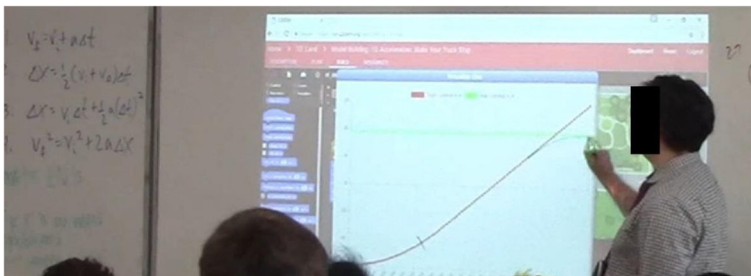


Fig. 15 Teacher promoting inquiry through class discussion

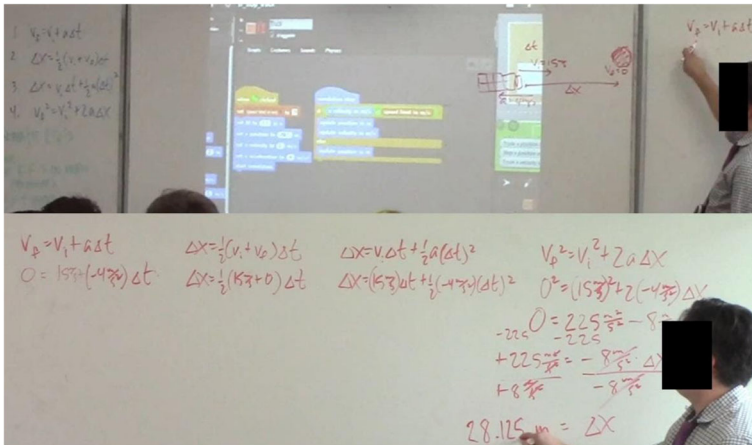


Fig. 16 Class discussion utilizing the DSMLs and simulation to problem-solve with the kinematic equations

Fig. 15 shows that the teacher reviewed the overall motion of the truck by invoking parts that the students have already modeled (the speed up and cruise phases of the motion). Referring back to discussions from the previous class, he noted while also drawing perpendicular green color lines on the position-time plot to delineate the different phases of motion:

*“Notice somewhere right around here it goes from that curving up (the plot), where it is speeding up, to a linear plot. We want to see that you are actually able to do that. Now clearly this isn’t achieving the goal of actually making it stop and so if you actually have the [plot] do this (drawing the curve seen at the time point of Fig. 15), then you’re in better shape than I am.”*

At this point, the teacher had linked the DSML blocks that the students used to create the speeding up and cruise behaviors with a plot of the distance-time graph. The

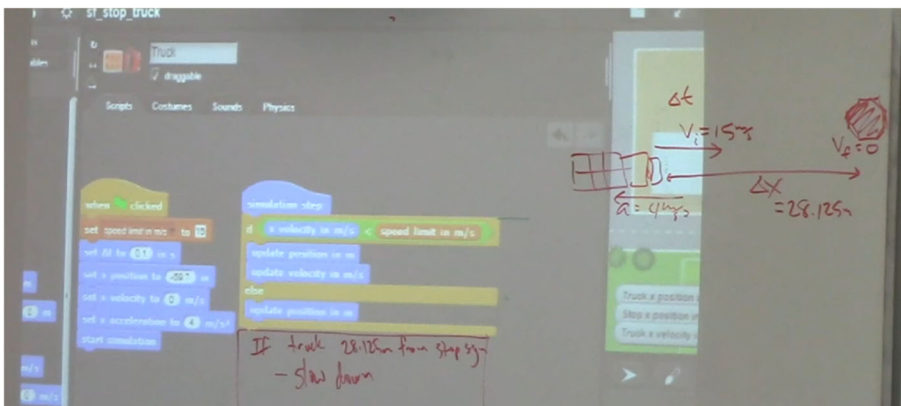


Fig. 17 Pseudocode given by teacher for using the lookahead distance

primary purpose was to help students review and understand the behaviors generated by the *step-by-step* execution of their DSML constructs.

Using the graphs from the previous phases, the teacher also clarified that the current model would not generate the stopping behavior of the truck. So, he went back to the kinematics equations he had written on the whiteboard earlier (see Fig. 16), and began a discussion on how students could calculate the time at which to start decelerating, so the truck's  $x$ -velocity = 0 at the position of the STOP sign. The teacher went through the solution steps and showed students the process for deriving the *lookahead distance* (see the bottom image of Fig. 16).

As a next step, the teacher guided students to pick the appropriate DSML block for modeling deceleration, and the conditional constructs needed to initiate the deceleration. Fig. 17 demonstrates the conditional logic developed through classroom discussion (in text form), but then the teacher left it to the students to add the slowdown and stopping phase to the model. However, the teacher also provided additional hints, “*on the speed up, update velocity, if it is not at the speed limit. If it is at the speed limit, then I just keep moving (with constant velocity). This is going to become a problem, because as soon as it starts to slow down [given the newly calculated lookahead distance and written conditional logic], it will actually try to speed up here (pointing to the DSML blocks in the first conditional statement).*” The idea was to make students rethink their logic of the initial statements, and to take into account the period for which the truck needed to slow down in a *step-by-step* manner until it came to a stop. This example demonstrates how the DSMLs, when coordinated with system tools, supports the teacher in classroom discussion.

## Discussion and Conclusions

We summarize the primary contributions of our work below.

### **A Framework for the Integrated Design and Development of DSMLs Supporting Learning-by-Modeling in STEM Domains**

The integration of computational modeling concepts and practices into STEM classrooms can provide opportunities for the simultaneous learning and application of CT and STEM concepts and practices. While research has demonstrated the significant benefits of this integration, difficulties in implementation may impact the introduction of *learning-by-modeling* paradigms into K-12 STEM classrooms. As a means of addressing some of these limitations, this paper presents a novel approach to designing age-appropriate K-12 modeling environments, inspired by applications in software engineering and integrated knowledge representation schemes (i.e., representation and accompanying reasoning mechanisms) in AI. To establish this, we analyzed the following research question: *What are the design steps necessary for the developing abstractions and mapping the DSML constructs created to ensure curriculum learning objectives are met?* We adopted a theoretical framing for DSML design that included the necessary science concepts and relations expressed in computational form at a level of abstraction that made computational modeling feasible for different levels of K-12 science classrooms. The DSML design framework was supported by an ECD process

that was used to generate and combine science and CT concepts and practices to define the curricular objectives. The result is that our proposed framework:

1. Adopts a set of design principles, that includes evidence-centered design, exploratory learning of dynamic processes, and the utilization of block-based programming languages, to scaffold a learning by modeling approach framework that supports students and their teachers.
2. Provides a systematic approach to developing a visual block-structured programming language that maps designated curriculum standards, including curriculum-specific concepts and practices in multiple science domains to computational modeling languages.

In addition to outlining the theory and design framework, we also provide example applications of the DSML design framework for three separate science domains: two of them focused on learning domain concepts and principles by building computational models, and the third focused on using science models to support engineering design. We compared DSML implementations across these domains, and demonstrated how our design principles supported systematic development of the corresponding modeling languages across these domains. Using case studies, we showed that correct interpretation and use of the DSML structures improved students' model building and learning performance in the science domain and CT.

Although we describe the application of our DSML design approach using one block-based programming environment (an extension of Snap!) our approach is generalizable in that it can be applied to multiple STEM domains, and multiple modeling environments. For example, we have implemented DSMLs in CTSiM (Basu et al. 2016a and b; Zhang et al. 2017), which incorporates the NetLogo modeling environment.

### **Interpretive Summary of Our Case Study Findings**

To demonstrate the impact of DSMLs in each domain, i.e., to provide answers to research question 2, we performed case study analyses demonstrating how the choice of modeling constructs and the level of abstraction support the synergistic learning of science and CT. Our analyses primarily focused on how DSML use related to student performance. In addition, we provided examples of how a teacher exploited the DSML constructs to help his students understand the nuances of science concepts and how they could be combined with computational constructs to build and test computational models. Overall, we used pre-posttest learning gains as a measure of science learning by students in the three different domains. In addition, we demonstrated a unique and successful approach to integrating science and engineering learning in lower middle school classrooms.

In previous work (e.g., Hutchins, et al. (2018, 2020) and Snyder, et al. (2019)), we have analyzed students discourse and model building work in much greater detail to demonstrate how they combine their domain and CT knowledge of concepts and practices in a synergistic manner to build their science models. Discourse analysis further illustrates how students go back and forth between the domain and CT constructs integrated into DSML structures to analyze errors in their models and correct

them. To keep this paper to a reasonable length, and to keep the focus on DSML design and development, we have not brought in results from past work into this paper, but we believe that they provide additional justification in supporting the effectiveness of our DSML structures.

Our case studies further demonstrated the advantages of these benefits, especially among students who used the DSML structures effectively. It has been shown that students using general purpose, block-based programming languages (e.g., Scratch, Snap!) face difficulties in applications of key CT constructs such as loops, variables, and expressions (Grover and Basu 2017). Similar issues have been reported in translating mathematical equations into simplified systems dynamics models (Wetzel et al. 2017) and not understanding how to generate qualitative constraints when building constraint-based (causal) models (e.g., Bredeweg et al. 2013). Through our case studies, we demonstrate that the DSML structures provide the context (by combining domain and CT concepts) to help students overcome these difficulties. In our work, the *step-by-step* execution approach, as well as the linked animations and plotting functions provided students with additional help to overcome these grounding issues. We demonstrated this through our contrasting case studies in the “[Impact on Student Learning and Performance](#)” section.

In recent work on the integration of computational modeling with STEM disciplines, additional difficulties that students experienced were outlined in the “[Students' Difficulties with Learning-by-Modeling](#)” section. In this paper, we have demonstrated that DSMLs provide a representational structure that explicitly targets STEM domain concepts, which, in turn support learning by computational modeling (Fig. 1). In this manner, they seem to directly target difficulties students have experienced in previous work. Through our case studies we have shown where DSMLs helped students overcome difficulties in understanding the nature of variables and their relations. For instance, the debugging of a stopping condition for the package drop task in Physics allowed the student to use the data table generated by the motion of the package to come up with the correct logical expression for stopping the package when it hit the ground. In all examples where students correctly implemented the dynamic, *step-by-step* change in the model's behavior (StudentA, StudentC, and StudentD), the students also demonstrate proficiency in translating a mathematical equation into computational form to update the position of the object (Physics) or the width of a coral using a growth rate (StudentD, Marine Biology). Students demonstrated abilities to qualitatively reason about models and data (e.g., describing agreement with a marine biologist given a set of data). However, issues regarding proper application of the simulation step and the use of custom blocks do highlight the need for additional scaffolding on these concepts to help students benefit even more in their *learning-by-modeling* processes.

From the classroom perspective, DSMLs added two key benefits. The first is in the grounding of *modeling language* definitions to help the teacher link domain principles to model constructs and then to model behavior and results. The systematic design and implementation of the blocks supported a uniform approach to classroom *learning-by-modeling* that was directly connected to the classroom teacher's curriculum. The second is the reduction in necessary training time for all students as compared to previous literature based on integrating text-based STEM modeling approaches (sometimes requiring multiple weeks of training (Hashem and Mioduser 2011) or major variations in training required based on prior knowledge (VanLehn et al. 2016)). It is



important to note that our training focused on key STEM and CT constructs implemented (e.g., what is a conditional statement) and not specifically on the *modeling language*. While the semester-long high school study did require 5 class hours of initial training, this time included the introductions to key Kinematics concepts (the start of C2STEM was the start of the kinematics unit in their class) and CT concepts. A brief review of studies conducted with general purpose, block-based languages appeared to indicate similar requirements to our approach.

A consistent theme apparent in each of our examples is the use of multiple linked representations (Basu et al. 2017) and a seamless way to transition between representations. Visual, numerical, and tabular data representations of models and modeling behavior, and as the teacher demonstrated in his Physics classroom, it also brings out the advantages of the self-documenting nature of DSMLs. Furthermore, students can compare the narrative of code they generate as their computational models to a *step-by-step* execution of the simulations, and they can also observe the results in the data tables and plots generated. When blocks are used appropriately, this process can be seen to support key learning opportunities.

We also have some evidence of the impact of the DSML-centered *learning-by-modeling* approach on the transfer of problem-solving skills to new domains. We implemented a Preparation for Future Learning (PFL) assessment (Schwartz et al. 2005) that was designed to utilize evidence from students' learning trajectories and current understanding to see how they may apply this knowledge to solve new problems (Hutchins et al. 2020). Results from the PFL assessment conducted at the end of the Physics unit showed that students who used the DSML-based modeling environment were more likely to apply general problem-solving strategies, especially the *step-by-step* simulation approach in a new physics context than students in a traditional physics classroom. A detailed analysis of our Preparation for Future Learning results is outlined in Hutchins et al. (2020). Given the differences in the transfer of problem-solving approaches between students who utilized our *learning-by-modeling* approach versus those that did not, further research on the effect of DSMLs in near and far transfer situations need to be conducted.

## Future Directions

Our DSML framework implemented in the C2STEM environment provides a systematic approach for designing domain-specific languages using visual programming constructs that focus science learning in multiple domains across multiple age levels. As we run studies with our existing DSML constructs, we will continue to analyze student difficulties in using our DSML constructs, and how they may be re-designed to facilitate better learning of STEM and CT concepts and practices.

Our case study analysis demonstrated the potential benefits of providing feedback based on DSML use (for instance, providing feedback to StudentB that they were updating position and velocity using the same equation). There have been efforts to analyze students' learning behaviors from their logged activity data, including the use of machine learning techniques to deepen insights about self-regulated learning (Gasevic et al., 2017; Biswas et al. 2018) and collaborative learning (Järvelä et al. 2020) strategies by analyzing sequences of learning activities. Additional exploratory data-driven approaches have targeted the design partial solution feedback (Piech et al.

2015), the identification of program states and the likelihood of reaching a solution state or facing a “sink” state in which a student was likely to get stuck (Blikstein et al. 2014), and applications of analytics measures to determine tinkering versus planning strategies in program and model development (Berland et al. 2013). Basu et al. (2016a), described students’ modeling progressions by calculating the distance to expert model at each model revision. DSMLs provide a unique opportunity in analyzing students’ activities. The domain-specific name of each block provides a context in which to better understand students’ reasoning processes associated with their model building actions. Tracking their use of DSML blocks over time helps us develop a fine-grained analysis of students’ CT and STEM learning and difficulties using approaches such as differential sequence mining (Kinnebrew et al. 2013; Dong et al. 2016) and clustering (Segedy et al. 2015; Zhang et al. 2017). A deeper understanding of students’ reasoning processes also provides opportunities for adaptive scaffolding that can help them get past their difficulties and advance their domain and CT learning.

**Acknowledgements** We thank Shuchi Grover, Brian Broll, Satabdi Basu, Kevin McElhane, Justin Montenegro, Beth Sanzenbacher, Naveed Mohammed, Kristen Pilner Blair, Doris Chin, Rachel Wolf and all of our C2STEM and SPICE project contributors for their assistance on this project.

**Availability of Data and Material** Data available on request from the authors.

**Code Availability** Available at <https://github.com/c2stem>

**Funding** This project was supported under National Science Foundation Award DRL-1640199 and National Science Foundation Award DRL-1742195.

## Compliance with Ethical Standards

**Conflicts of Interest/Competing Interests** No conflicts of interest to declare.

## References

- Araujo, I., Veit, E., & Moreira, M. (2008). Physics students' performance using computational modelling activities to improve kinematics graphs interpretation. *Computers and Education*, *50*(4), 1128–1140.
- Basu, S., Biswas, G., & Kinnebrew, J. S. (2016a). Using multiple representations to simultaneously learn computational thinking and middle school science. In *Proceedings of the thirtieth AAAI conference on artificial intelligence* (pp. 3705–3711). Arizona, USA: Phoenix.
- Basu, S., Biswas, G., & Kinnebrew, J. S. (2017). Learner modeling for adaptive scaffolding in a computational thinking-based science learning environment. *User Modeling and User-Adapted Interaction*, *27*(1), 5–53.
- Basu, S., Biswas, G., Sengupta, P., Dickes, A., Kinnebrew, J. S., & Clark, D. (2016b). Identifying middle school students' challenges in computational thinking-based science learning. *Research and Practice in Technology Enhanced Learning*, *11*(1), 1–35.
- Basu, S., Dickes, A., Kinnebrew, J. S., Sengupta, P., & Biswas, G. (2013). CTSiM: A computational thinking environment for learning science through simulation and modeling. In *Proceedings of the 5th international conference on computer supported education* (pp. 369–378). Germany: Aachen.
- Basu, S., McElhane, K., Grover, S., Harris, C., & Biswas, G. (2018). A principled approach to designing assessments that integrate science and computational thinking. In *Proceedings of the 13th international conference of the learning sciences* (pp. 384–391). London, England.
- Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the ACM*, *60*(6), 72–80.
- Berland, M., Martin, T., Benton, T., Smith, C. P., & Davis, D. (2013). Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences.*, *22*(4), 564–599.

- Biswas, G., Baker, R. S., & Paquette, L. (2018). Data mining methods for assessing self-regulated learning. In D. H. Schunk & J. A. Greene (Eds.), *Educational psychology handbook series. Handbook of self-regulation of learning and performance* (p. 388–403). Routledge/Taylor & Francis Group.
- Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences*, 23(4), 561–599.
- Bredeweg, B., Liem, J., Beek, W., Linnebank, F., Gracia, J., Lozano, E., Wißner, M., Bühling, R., Salles, P., Noble, R., Zitek, A., Borisova, P., & Mioduser, D. (2013). DynaLearn – An intelligent learning environment for learning conceptual knowledge. *AI Magazine*, 34(4), 46–65. <https://doi.org/10.1609/aimag.v34i4.2489>.
- Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking. Paper presented at annual American Educational Research Association meeting*. BC, Canada: Vancouver.
- Brodie, M. L., Mylopoulos, J., & Schmidt J. W. (Eds.). (2012). On conceptual modelling: Perspectives from artificial intelligence, databases, and programming languages. Springer Science & Business Media.
- Brown, N. C., Mönig, J., Bau, A., & Weintrop, D. (2016, February). Panel: Future directions of block-based programming. In *Proceedings of the 47th ACM technical symposium on computing science education* (pp. 315–316).
- CCSSO. (2011). The common core state standards for mathematics. Retrieved February 1, 2020, from [http://www.corestandards.org/wp-content/uploads/Math\\_Standards1.pdf](http://www.corestandards.org/wp-content/uploads/Math_Standards1.pdf)
- Chi, M. T. H. (2005). Common sense conceptions of emergent processes: Why some misconceptions are robust. *Journal of the Learning Sciences*, 14, 161–199.
- Chiu, J., McElhaney, K. W., Zhang, N., Biswas, G., Fried, R., Basu, S., & Alozie, N. (2019). A principled approach to NGSS-aligned curriculum development integrating science, engineering, and computation: A pilot study. In *Paper presented at the 2019 NARST annual international conference*.
- Clark, D., Nelson, B., Sengupta, P., & D'Angelo, C. (2009). Rethinking science learning through digital games and simulations: Genres, examples, and evidence. In *Learning science: Computer games, simulations, and education workshop sponsored by the National Academy of Sciences*. Washington DC.
- Dede, C. (2010). Technological supports for acquiring 21st century skills. In P. Peterson, E. Baker, & B. McGaw (Eds.), *International encyclopedia of education* (pp. 158–166). Oxford, England: Elsevier.
- van Deursen, A. (1997). Domain-specific languages versus object-oriented frameworks: A financial engineering case study. In *Smalltalk and Java in industry and academia, STJA'97* (pp. 35–39). Ilmenau Technical University.
- van Deursen, A., Klint, P., & Visser, J. (2000). Domain-specific languages: An annotated bibliography. *SIGPLAN Notices*, 35, 26–36.
- DiSessa, A. A. (2001). *Changing minds: Computers, learning, and literacy*. MIT Press.
- Dong, Y., Kinnebrew, J., & Biswas, G. (2016). Comparison of selection criteria for multi-feature hierarchical activity Mining in Open-Ended Learning Environments. In *Proceedings of the 9th international conference on educational data mining* (pp. 591–592). North Carolina: Raleigh.
- Gasevic, D., Jovanovic, J., Pardo, A., & Dawson, S. (2017). Detecting learning strategies with analytics: Links with self-reported measures and academic performance. *Journal of Learning Analytics*, 4(2), 113–128. <https://doi.org/10.18608/jla.2017.42.10>.
- Grover, S. (2019). An Assessment for Introductory Programming Concepts in Middle School Computer Science. Presented at the 2019. In *Annual meeting of the National Council on measurement in education (NCME)*. Toronto: CA.
- Grover, S. (2020). Designing an Assessment for Introductory Programming Concepts in Middle School Computer Science. In *Proceedings of the 51st ACM Technical Symposium on Computing Science Education (SIGCSE'20)*, Portland, OR
- Grover, S., & Basu, S. (2017). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and Boolean logic. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education* (pp. 267–272). ACM.
- Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. In S. Sentance, E. Barendsen, & S. Carsten (Eds.), *Computer Science Education: Perspectives on teaching and learning*. Bloomsbury.
- Harvey, B., Garcia, D. D., Barnes, T., Titterton, N., Armendariz, D., Segars, L., Lemon, E., Morris, S., & Paley, J. (2013). SNAP! (build your own blocks). In *Proceedings of the 44th ACM technical symposium on computer science education, SIGCSE '13* (p. 759).

- Hashem, K., & Mioduser, D. (2011). The contribution of learning by modeling (LbM) to students' understanding of complexity concepts. *International Journal of e-Education, e-Business, e-Management and e-Learning*, 1(2), 151–157.
- Hestenes, D., Wells, M., & Swackhamer, G. (1992). Force concept inventory. *The Physics Teacher*, 30, 141–166.
- Hilton, M. (2010). *Exploring the intersection of science education and 21st century skills: A workshop summary*. National Academies Press.
- Hudak, P. (1996). Building domain-specific embedded languages. *ACM Computing Surveys (CSUR)*, 28(4), 196–1es.
- Hutchins, N., Biswas, G., Conlin, L., Emara, M., Grover, S., Basu, S., & McElhaney, K. (2018). Studying synergistic learning of physics and computational thinking in a learning by modeling environment. In J. C. Yang et al. (Eds.), *In proceedings of the 26th international conference on computers in education* (pp. 153–162). Philippines: Manila.
- Hutchins, N., Biswas, G., Maróti, M., Lédeczi, A., Grover, S., Wolf, R., Blair, K. P., Chin, D. B., Conlin, L., Basu, S., & McElhaney, K. (2020). C2STEM: A system for synergistic learning of physics and computational thinking. *Journal of Science Education and Technology (JOST)*, 29, 83–100. <https://doi.org/10.1007/s10956-019-09804-9>.
- Järvelä, S., Gašević, D., Seppänen, T., Pechenizkiy, M., & Kirschner, P. A. (2020). Bridging learning sciences, machine learning and affective computing for understanding cognition and affect in collaborative learning. *British Journal of Educational Technology*. <https://doi.org/10.1111/bjet.12917>.
- Jona, K., Wilensky, U., Trouille, L., Horn, M. S., Orton, K., Weintrop, D., & Beheshti, E. (2014). *Embedding computational thinking in science, technology, engineering, and math (CT-STEM)*. In *future directions in computer science education summit meeting*. FL: Orlando.
- Jonassen, D., Strobel, J., & Gottdenker, J. (2005). Model building for conceptual change. *Interactive Learning Environments*, 13(1–2), 15–37.
- van Joolingen, W. R., De Jong, T., Lazonder, A., Savelsbergh, E. R., & Manlove, S. (2005). Co-lab: Research and development of an online learning environment for collaborative scientific discovery learning. *Computers in Human Behavior*, 21, 671–688.
- Karsai, G., Krahn, H., Pinkemell, C., Rumpe, B., Schindler, M., & Völkel, S. (2014). Design guidelines for domain specific languages. *ArXiv*, abs/1409.2378.
- Keating, T., Barnett, M., Barab, S. A., & Hay, K. E. (2002). The virtual solar system project: Developing conceptual understanding of astronomical concepts through building three-dimensional computational models. *Journal of Science Education and Technology*, 11(3), 261–275.
- Kelly, S. & Tolvanen, J. (2008). Domain-specific modeling : Enabling full code generation. Retrieved from <https://ebookcentral.proquest.com>
- Kinnebrew, J. S., Loretz, K. M., & Biswas, G. (2013). A contextualized, differential sequence mining method to derive students' learning behavior patterns. *Journal of Educational Data Mining*, 5(1), 190–219.
- Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010). Towards the automatic recognition of computational thinking for adaptive visual language learning. In *Proceedings of the 2010 IEEE symposium on visual languages and human-centric computing* (pp. 59–66). Leganes.
- Lédeczi, A., Bakay, A., Maroti, M., Volgyesi, P., Nordstrom, G., Sprinkle, J., & Karsai, G. (2001). Composing domain-specific design environments. *Computer*, 34(11), 44–51.
- Leclawong, K., & Biswas, G. (2008). Designing learning by teaching agents: The Betty's brain system. *International Journal of Artificial Intelligence in Education*, 18(3), 181–208.
- Lehrer, R., & Schauble, L. (2015). The development of scientific thinking. In R. M. Lerner, L.S. Liben, & U. Mueller (Eds.), *Handbook of child psychology and developmental science*, 2(7), 671–714.
- Levesque, H. J. (1986). Knowledge representation and reasoning. *Annual review of computer science*, 1(1), 255–287.
- McElhaney, K. W., Basu, S., Wetzel, T., & Boyce, J. (2019). Three-dimensional assessment of NGSS upper elementary engineering design performance expectations. In *NARST Annual International Conference*.
- Metcalfe, S. J., Krajcik, J., & Soloway, E. (2000). Model-it: A design retrospective. In M. J. Jacobson & R. B. Kozma (Eds.), *Innovations in science and mathematics education: Advanced designs for technologies of learning* (pp. 77–115). Mahwah, NJ: Lawrence Erlbaum Associates.
- Mislevy, R. J., & Haertel, G. D. (2006). Implications of evidence-centered design for educational testing. *Educational Measurement: Issues and Practice*, 25(4), 6–20.
- Mislevy, R. J., & Riconscente, M. (2005). *Evidence-centered assessment design: Layers, structures, and terminology (PADI technical report 9)*. Menlo Park, CA: SRI International.
- N. G. S. S. Lead States (2013). *Next generation science standards: For States, by States*. Washington, DC: The National Academies Press.

- Nikolai, C., & Madey, G. (2009). Tools of the trade: A survey of various agent based modeling platforms. *Journal of Artificial Societies and Social Simulation*, 12(2), 2.
- Niwa, K., Sasaki, K., & Ihara, H. (1984). An experimental comparison of knowledge representation schemes. *AI Magazine*, 5(2), 29–29.
- Olson, I. C., Hom, M., & Wilensky, U. (2011). Modeling on the table: Agent-based modeling in elementary school with NetTango. In *Proceedings of 10th international conference on interaction design and children*. Ann Arbor: ML.
- Paige, R. F., Ostroff, J. S., & Brooke, P. J. (2000). Principles for modeling language design. *Information and Software Technology*, 42, 665–675.
- Pausch, R., Burnette, T., Capeheart, A. C., Conway, M., Cosgrove, D., DeLine, R., Durbin, J., Gossweiler, R., Koga, S., & White, J. (1995). Alice: Rapid prototyping system for virtual reality. *IEEE Computer Graphics and Applications*, 15(3), 8–11.
- Piech, C., Huang, J., Nguyen, A., Phulsuksombati, M., Sahami, M., & Guibas, L. (2015). Learning program embeddings to propagate feedback on student code. In *Proceedings of the 32nd international conference on machine learning* (pp. 1093–1102). France: Lille.
- Redish, E. F., & Wilson, J. M. (1993). Student programming in the introductory physics course: M.U.P.P.E.T. *American Journal of Physics*, 61, 222–232.
- Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proceedings of the 41st ACM technical symposium on computer science education (SIGCSE)*. Milwaukee: ACM Press.
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60–67.
- Schwartz, D. L., Bransford, J. D., & Sears, D. (2005). Efficiency and innovation in transfer. In J. Mestre (Ed.), *Transfer of learning: Research and perspectives* (pp. 1–52). Greenwich, CT: Information Age Publishing.
- Schwarz, C. V., & White, B. Y. (2005). Metamodelling knowledge: Developing students' understanding of scientific modelling. *Cognition and Instruction*, 23(2), 165–205.
- Segedy, J. R., Kinnebrew, J. S., & Biswas, G. (2015). Using coherence analysis to characterize self-regulated learning Behaviours in open-ended learning environments. *Journal of Learning Analytics*, 2(1), 13–48.
- Selic, B. (2007). A systematic approach to domain-specific language design using UML. In *Proceedings of the 10th IEEE international symposium on object and component-oriented real-time distributed computing* (pp. 2–9). Santorini Island.
- Sengupta, P., Dickes, A., & Farris, A. (2018). Toward a phenomenology of computational thinking in STEM education. In M. Khine (Ed.), *Computational thinking in the STEM disciplines*. Cham: Springer.
- Sengupta, P., Dickes, A., Farris, A. V., Karan, A., Martin, D., & Wright, M. (2015). Programming in K-12 science classrooms. *Communications of the ACM*, 58(11), 33–35.
- 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 & Children* (pp. 78–87).
- Sengupta, P., Farris, A. V., & Wright, M. (2012). From agents to continuous change via aesthetics: Learning mechanics with visual agent-based computational modeling. *Technology, Knowledge and Learning*, 17(1–2), 23–42.
- 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.
- Shen, J., Lei, J., Chang, H. Y., & Namdar, B. (2014). Technology-enhanced, modeling-based instruction (TMBI) in science education. In *Handbook of Research on Educational Communications and Technology* (Fourth ed., pp. 529–540). New York: Springer. [https://doi.org/10.1007/978-1-4614-3185-5\\_41](https://doi.org/10.1007/978-1-4614-3185-5_41).
- Sherin, B. L. (2001a). A comparison of programming languages and algebraic notation as expressive languages for physics. *International Journal of Computers for Mathematical Learning*, 6(1), 1–61.
- Sherin, B. L. (2001b). How students understand physics equations. *Cognition and Instruction*, 19(4), 479–541.
- Sherin, B., diSessa, A. A., & Hammer, D. M. (1993). Dynaturtle revisited: Learning physics through collaborative design of a computer model. *Interactive Learning Environments*, 3(2), 91–118.
- Snyder, C., Hutchins, N., Biswas, G., Emará, M., Grover, S., & Conlin, L. (2019). Analyzing students' synergistic learning processes in physics and CT by collaborative discourse analysis. In *Proceedings of the international conference on computer supported collaborative learning* (pp. 360–367). Lyon, France.



- Sun, D., & Looi, C.K. (2013). Designing a web-based science learning environment for model-based collaborative inquiry. *Journal of Science Education and Technology*, 22(1), 73–89.
- Tissenbaum, M., Sheldon, J., & Abelson, H. (2019). From computational thinking to computational action. *Communications of the ACM*, 62(3), 34–36.
- Tisue, S., & Wilensky, U. (2004). *NetLogo: Design and Implementation of a Multi-Agent Modeling Environment. Paper presented at the Agent2004 Conference*. Chicago, IL.
- Trowbridge, D. E., & McDermott, L. C. (1981). Investigation of student understanding of the concept of acceleration in one dimension. *American Journal of Physics*, 49(3), 242–253.
- VanLehn, K. (2013). Model construction as a learning activity: A design space and review. *Interactive Learning Environments*, 21(4), 371–413. <https://doi.org/10.1080/10494820.2013.803125>.
- VanLehn, K., Chung, G., Grover, S., Madni, A., & Wetzel, J. (2016). Learning science by constructing models: Can dragoon increase learning without increasing the time required? *International Journal of Artificial Intelligence in Education*, 26(4), 1033–1068. <https://doi.org/10.1007/s40593-015-0093-5>.
- VanLehn, K., Wetzel, J., Grover, S., & van de Sande, B. (2015). Learning how to construct models of dynamic systems: An initial evaluation of the dragoon intelligent tutoring system. *IEEE Transactions on Educational Technology*, 10(2), 154–167. <https://doi.org/10.1109/TLT.2016.2514422>.
- Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127–147.
- Werner, L., McDowell, C., & Denner, J. (2013). A first step in learning analytics: Pre-processing low-level Alice logging data of middle school students. *Journal of Educational Data Mining*, 5(2), 11–37.
- Wetzel, J., VanLehn, K., Chaudhari, P., Desai, A., Feng, J., Grover, S., Joiner, R., Kong-Silver, M., Patade, V., Samala, R., Tiwari, M., & van de Sande, B. (2017). The design and development of the dragoon intelligent tutoring system for model construction: Lessons learned. *Interactive Learning Environments*, 25(3), 361–381. <https://doi.org/10.1080/10494820.2015.1131167>.
- Wieman, C. E., Adams, W. K., & Perkins, K. K. (2008). PhET research: Simulations that enhance learning. *Science*, 322, 682–683.
- Wilensky, U., Brady, C. E., & Horn, M. S. (2014). Fostering computational literacy in science classrooms. *Communications of the ACM*, 57(8), 24–28.
- Wilensky, U., & Reisman, K. (2006). Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories—An embodied modeling approach. *Cognition and Instruction*, 24(2), 171–209.
- Wilensky, U., & Resnick, M. (1999). Thinking in levels: A dynamic systems perspective to making sense of the world. *Journal of Science Education and Technology*, 8(1), 3–19.
- Wing, J. (2011). Research notebook: Computational thinking—What and why. *The Link Magazine*, 20–23.
- Zhang, N., Biswas, G., Chiu, J. L., & McElhane, K. W. (2019). Analyzing students' design solutions in an NGSS-aligned earth sciences curriculum. In *Proceedings of the 20th international conference on artificial intelligence in education* (pp. 532–543). Chicago.
- Zhang, N., Biswas, G., & Dong, Y. (2017). Characterizing students' learning behaviors using unsupervised learning methods. In E. André, R. Baker, X. Hu, M. Rodrigo, & B. du Boulay (Eds.), *Artificial intelligence in education* (pp. 430–441). Wuhan, China: Lecture notes in computer science (Vol. 10331). Cham: Springer.
- Zhang, N., Biswas, G., McElhane, K. W., Basu, S., McBride, E., & Chiu, J. L. (2020). Studying the Interactions Between Science, Engineering, and Computational Thinking in a Learning-by-Modeling Environment. In I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin, & E. Millán (Eds.), *Artificial Intelligence in Education. AIED 2020. Lecture notes in computer science* (Vol. 12163). Cham: Springer.