Investigating Student Generated Computational Models of Science

Satabdi Basu, Anton Dukeman, John S. Kinnebrew, Gautam Biswas and Pratim Sengupta
Vanderbilt University, Nashville, TN, USA
{satabdi.basu, anton.dukeman, john.s.kinnebrew, gautam.biswas, pratim.sengupta}@vanderbilt.edu

Abstract: Computational Thinking (CT) is now considered a core competency in problem formulation and problem solving. In spite of the known synergies between CT and science education, integrating CT in K-12 science classrooms is challenging. This paper reports a teacher-led, multi-domain classroom study conducted with 6th graders using CTSiM – a learning environment for CT and middle school science. Pre-post comparisons show that students made significant gains, both in terms of computational thinking and the relevant science concepts. Furthermore, we developed measures for analyzing students' computational models, and our results show that as challenges faced decreased, model accuracy not only increased in general, but also became a good predictor of individual learning gains.

Introduction

Computational Thinking (CT) refers to the concepts and representational practices involved in formulating and solving problems, designing systems, and understanding human behavior by drawing on computer science concepts like problem representation, abstraction, decomposition, recursion, simulation, and verification (Wing, 2010). CT parallels core practices of science education, and computational modeling is an effective vehicle for learning a variety of challenging science and math concepts (Harel, 1990; Guzdial, 1995; Vattam et al., 2011).

In spite of the known synergies between CT and science education, integrating computational programming in science classrooms is challenging. Furthermore, relatively little is known about students' conceptual understanding and developmental processes in curricula that involve learning programming and computational modeling in conjunction with scientific concepts (Grover & Pea, 2013). Our study seeks to make contributions along both these dimensions. Over two years, we have developed CTSiM (Computational Thinking using Simulation and Modeling) (Sengupta et al., 2013), which provides students with an agent-based, visual programming platform where they can conceptualize given science phenomena, program agent-based simulations, and compare their simulations against expert simulations. We present a study in which a 6th-grade science teacher with no computer science background successfully led classroom instruction with CTSiM. The topics covered in the class included the study of kinematics (motion of objects) and fish tank ecosystems.

Specifically, we investigate the following research questions using students' pre-post test scores and activity data from log files:

1. Does CTSiM help students learn both science content and CT skills synergistically? In particular, are the domain and computational measures correlated for (a) learning gains, and (b) correctness of primitives used in the student generated models as compared against the expert model?
2. Does the correctness of the student-generated models vary between different modeling activities and, if so, is this measure related to the number of challenges faced by students in each of the activities?
3. Can students' computational models predict their learning gains from the pre- to post-test?

The CTSiM Learning Environment and Learning Activities

CTSiM adopts a learning-by-design pedagogical approach where students engage in scientific inquiry by participating in various phases of model building, simulation, and verification. Model building is done in two steps: conceptualization and computational modeling. First, students conceptualize the given science phenomenon by creating a concept map using a node-link interface. To construct this map, students select the appropriate entities of the system, their properties, behaviors, and interactions. Students then construct their computational models using a visual programming interface we call the 'Construction world' (see Figure 1). Students can select primitives from a library, which include domain-specific (e.g., "speed-up" in kinematics, "feed" in biology) and domain-general primitives (e.g., conditionals and loops). The set of domain-specific primitives are available to the students only if they have been correctly specified in the conceptualization interface. CTSiM provides scaffolds for algorithm visualization through simultaneous simulation and code step-through. Students can also verify their models by comparing a NetLogo (Wilensky, 1999) simulation of their models with an 'expert' NetLogo simulation for the phenomenon in the 'Envisionment world' (see Figure 2). However, they cannot inspect the expert computational model underlying the expert simulation. Identifying differences between the simulations helps students realize how to refine and correct their conceptual and computational models.

Currently, CTSiM comprises four modeling activities across two domains (Kinematics and Ecology):

Activity 1 – Shape drawing: Students generate algorithms to draw simple shapes to explore the relations between speed, acceleration and distance. They start by modeling shapes like squares and triangles where each segment of the shape is of the same length, implying constant speed. This also familiarizes them with programming primitives such as "forward", "right turn", "left turn", and "repeat". Then, students modify their algorithms to generate spirals of the same shapes, where each line segment is longer (or shorter) than the previous one, implying constant acceleration. Students also re-represent a speed-time graph by drawing shapes such that the length of segments in the shapes is proportional to the speed in the given graph.

Activity 2 – Modeling a roller coaster: Students progress from modeling simple shapes to modeling a real-world phenomenon – a roller coaster's behavior as it moves along segments of a track: (1) up (pulled by a motor) at constant speed, (2) down (free fall), (3) flat (cruising), and (4) up again (without a motor). An expert simulation is provided and students try to build models to match the expert roller coaster behavior. This activity also involves use of more complex computational constructs like conditionals and nested conditionals.

Activity 3 – Modeling a fish tank (macro-level): Students progress from modeling a single agent with a single behavior in the previous activities to modeling multiple agents, each with multiple behaviors. They model part of a closed fish tank system - a macro-level semi-stable model with two agents: fish and duckweed. This involves constructing a model of the food chain, the respiration and reproductive processes of the fish and duckweed, and the macro-level elements of the waste cycle. The non-sustainability of the macro-model (the fish and the duckweed gradually die off), encourages students to reflect on what might be missing from the model, prompting the transition to Activity 4, after they identify the build-up of toxic fish waste as the culprit.

Activity 4 – Modeling a fish tank (micro-level): Students recognize the need for bacteria, and model two types of bacteria in the fish tank, which, through stages, help convert the fish waste to nutrients (nitrates) for the duckweed. They model the waste cycle where the Nitrosomonas bacteria convert the toxic ammonia in the fish waste to nitrites, and then the Nitrobacter bacteria convert the nitrites to nitrates, which help sustain the duckweed. The plots generated in the simulation provide students with an aggregate level understanding of the interdependence and balance among the different agents in the system.

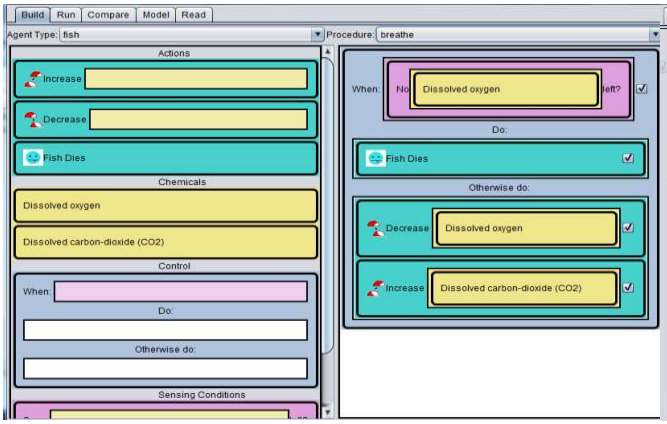

Figure 1. The Construction world with a 'breathe' procedure for 'fish' agents in a fish-tank unit

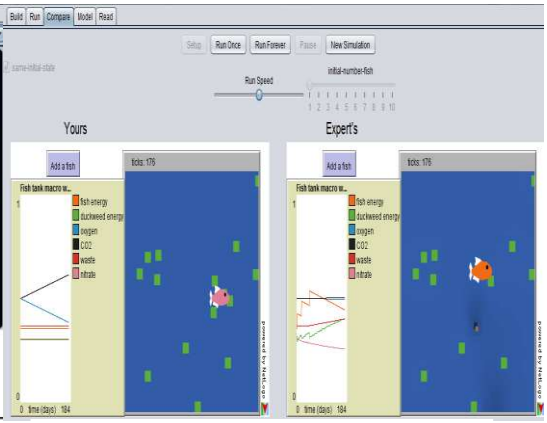

Figure 2. The Envisionment world for a fish-tank unit

Method

We conducted an in-class study using CTSiM with 25 6th-grade, middle Tennessee students (13 female, 12 male, average age=11.5). Each student worked individually on each of the four modeling activities described in Section 2. The study was run daily during the science period and the science teacher conducted all the instruction and led the classroom discussions. The teacher had no significant prior experience with programming and was introduced to CTSiM during two 90-minute training sessions before the study. He alternated between instructing (including facilitating class discussions) using CTSiM and having students work individually on the modeling tasks. One person from our research group was always present in the classroom to assist him with answering students' questions, primarily those related to technical issues.

Students were administered pre- and post-tests in which the questions assessed Kinematics and Ecology knowledge, as well as Computational thinking skills. The Kinematics questions tested the concepts and relationships among speed, acceleration, and distance, including the generation and interpretation of speed-time graphs. The Ecology questions focused on the roles of species in the ecosystem, interdependence among the species, the waste cycle, and the respiration cycle. CT skills were assessed by asking students to use a provided set of primitives to model various scenarios that were not part of the CTSiM modeling activities.

Students worked on the Kinematics and Ecology units in 50-minute daily sessions for five days each. On Day 1, students took pre-tests for both the Kinematics and Ecology units. They worked on Modeling Activi-

ty 1 for days 2-4, and Activity 2 on days 5 and 6 before taking the Kinematics post-test on day 7. Students then moved on to the Ecology unit and worked on Activity 3 during days 8-10 and on Activity 4 on days 11 and 12. All students took the Ecology post-test on day 13. One student was absent for the Ecology post-test and is not included in the results presented for the Ecology unit. All student actions on the CTSiM system were logged for post-hoc analysis.

To evaluate the accuracy of the student-generated models in relation to an expert model, we calculated a vector-distance metric. Accuracy is measured as the distance between a student's model and the expert model, where a model distance of 0 implies that the model contained all the primitives in the expert model, and additionally contained no extraneous and incorrect primitives. Since our metric is based on a comparison with primitives in the expert model, and shapes in Activity 1 can be correctly modeled in a number of ways using different sets of primitives, vector distances were calculated only for models built in Activities 2-4.

Our distance measure for comparing models was derived from the Bag of Words metric (Piech et al., 2012). For each modeling activity, each procedure associated with each of the agents was represented as a bag of words, which was the collection of visual primitives used by a student in that procedure. Based on whether the primitives were computational or related to the domain being modeled, the bag of words was further labeled as computational or domain collections. To calculate a correctness measure, we normalized the size of the intersection of the student and expert models by the size of the expert model (summation of number of copies of each primitive). As a result, correctness scores are bounded between 0 and 1 inclusive. A score of 1 means the student had all the primitives necessary to build the model correctly. However, the bag of words does not analyze the structure further. For example, a student may have used all the right primitives, but the metric was not sensitive to incorrect assembly of the primitives, or if extraneous primitives were used in the procedure. Though a metric which neglects ordering of primitives might seem problematic, the specifics of our expert models and how our primitives were defined helped minimize the probability of such errors. For each modeling activity, we combined the correctness scores from its procedures using a weighted average based on the size of each procedure's expert model as shown in equation 1. Similarly, the computational and domain splits were used to get computational-specific and domain-specific correctness scores.

Since the correctness measure does not account for extraneous primitives, we also calculated an incorrectness measure for each procedure by counting the number of extra primitives in the procedure and normalizing it by the size of the expert model. A score of 0 meant no extra blocks were used in that procedure, while a score of 2 would mean that twice as many extraneous primitives as the total number of primitives required for a correct model had been used. We also combined the incorrectness scores from all procedures of a modeling activity using a weighted average as shown in equation 2. In order to assign an overall score to each model, we created a two-dimensional vector with the correctness and incorrectness measures and calculated its distance to the vector (1,0), as shown in equation 3.

Equation 1:
$$weightedAverageCorrectness = \frac{\sum_{each\ procedure} |user \cap expert|}{\sum_{each\ procedure} |expert|}$$

Equation 2:
$$weightedAverageIncorrectness = \frac{\sum_{each\ procedure} (|user| - |user \cap expert|)}{\sum_{each\ procedure} |expert|}$$

Equation 3:
$$distance = \sqrt{incorrectness^2 + (correctness - 1)^2}$$

Results

The intervention resulted in statistically significant learning gains for each of the two units, as seen in Table 1. Both units produced significant overall learning gains, as well as significant gains for domain content and CT skills, measured separately.

Table 1: Paired t-tests showing learning gains for the Kinematics and Ecology pre-post tests

Domain		Pre-test Mean(S.D.)	Post-test Mean(S.D.)	Max value	2-tailed *p*-value	Effect size
Kinematics (n=25)	Total	19.1(5.9)	25.9(7.1)	53	<0.0001	0.44
	Domain	15.3(4.3)	19.1(4.4)	36	<0.0001	0.34
	CT	3.8(2.5)	6.7(3.8)	17	<0.001	0.42
Ecology (n=24)	Total	15.3(5.9)	31.6(12.3)	62	<0.0001	0.65
	Domain	9.1(3.1)	19.7(7)	34.5	<0.0001	0.70
	CT	6.2(3.4)	12(6.3)	27.5	<0.0001	0.50

Correlations between Domain and Computational Measures

In order to investigate the potential synergy between learning CT and science in CTSiM, we first calculated the correlation between the domain and CT learning gains for both the science units. We used normalized learning gains to avoid potential ceiling effects for students with relatively high prior knowledge. The normalized gains were calculated by dividing the actual gains by the maximum amount that could have been gained based on pre-test scores. The correlations ($p < 0.01$) presented in Table 2 support our belief that CT and science concepts can be learned synergistically through computational modeling and simulation.

Table 2: Correlations between domain and CT learning gains

Normalized domain and CT gains	Pearson correlation coefficient (r)	2-tailed p-value
Kinematics (n=25)	0.52	<.01
Ecology (n=24)	0.56	<.01

To determine whether model accuracy scores for domain-related primitives and computational primitives were related in these activities, we calculated correlations between the vector distance scores for the domain-based and computational primitives for activities 2-4. Table 3 shows the correlations were high and significant (p <.0001). Since the domain and computational modeling scores are so strongly correlated, we only use the overall model accuracy score while reporting results in Sections 4.2 and 4.3.

Table 3: Correlations between domain and computational modeling scores (vector-distances)

Domain and computational vector distances	Pearson correlation coefficient (r)	2-tailed p-value
Roller coaster model (n=25)	0.72	<.0001
Fish macro model (n=24)	0.95	<.0001
Fish micro model (n=24)	0.93	<.0001

Variation of Student Model Correctness between Different Learning Activities

In order to answer Research question 2, we first compared the vector distance scores of students' roller coaster (RC) models (Activity 2) with that of students' fish tank macro models (Activity 3). A paired t-test comparing the vector distance scores for the RC models (mean vector distance = 0.23) versus the fish tank macro models (mean vector distance = 0.53) showed a significant decrease in accuracy ($t = 4.8$, p <0.0001). However, a comparison of the vector distance scores between the fish tank macro models (mean = 0.53) and micro models (mean = 0.34) showed a significant increase in model accuracy (t =2.6, p <0.05). We hypothesized that the change in the model accuracy across units could be linked to the number of challenges students faced while developing the models for each of the units. For example, the RC modeling activity, though more challenging than Activity 1 (shape drawing), required modeling only one agent and a single procedure for that agent. On the other hand, when students switched to the fish-macro modeling activity, they had to deal with new domain content, multiple agents and multiple procedures for each agent. In a previous, one-on-one interview-based study, we coded student videos for the number of challenges faced by students in the different learning activities. This analysis confirmed that challenges increased from activities 2 to 3 and then decreased from activities 3 to 4 (Basu, et al., 2013). While no quantitative conclusions can be drawn from this pattern across two different studies, the basic modeling activities were the same in each study, so we believe that the change in the correctness metric observed between the units in this study can be linked to the corresponding change in the number of challenges students faced across units.

Student Models as Predictors of Learning Gains

In order to answer Research question 3, we calculated the correlations between students' pre-post learning gains and the correctness of their models as measured by the vector-distance scores. We used normalized learning gains and calculated correlations with overall gains, as well as gains for domain content and CT separately. Table 4 reports the correlations between the RC model and Kinematics learning gains, as well as correlations between Ecology learning gains and the fish macro and micro models, respectively. The 2-tailed p-values are listed only for the correlations that are statistically significant.

The low correlations and lack of significance for the RC and fish-macro models indicate that students' performance on these models is not predictive of the respective pre-post learning gains. However, students' fish-micro model's accuracy is strongly negatively correlated to their Ecology learning gains (note that a smaller distance value implies a better model). We believe that the lack of predictability of the RC and fish-macro models is linked to the initial complexity of those modeling activities, as reflected in the challenges faced by the stu-

dents. Learning during these activities involved learning about new computational abstractions and scientific concepts with students facing difficulties in both. However, we posit that the resolution of some of these challenges, specifically those related to domain learning, may not be reflected in the computational models. Though most students got better at using both domain and computational primitives as these activities progressed, their individual model accuracy is not predictive of learning gains. On the other hand, when students attempted the more familiar, and consequently less challenging, task in the fish-micro activity, their models were predictive of their learning gains with high confidence.

Table 4: Correlation between correctness of student generated models and pre-post learning gains

Vector distance	Kinematics normalized gains (n=25)	Kinematics Domain normalized gains (n=25)	Kinematics CT normalized gains (n=25)	Ecology normalized gains (n=24)	Ecology Domain normalized gains (n=24)	Ecology CT normalized gains (n=24)
RC model (n=25)	r = -0.13	r = -0.01	r = -0.19	-	-	-
Fish macro model (n=24)	-	-	-	r = -0.06	r = -0.24	r = 0.17
Fish micro model (n=24)	-	-	-	r =-0.59 (p<.005)	r = -0.6 (p<.005)	r = -0.44 (p<.05)

Discussion and Conclusions

As Grover & Pea (2013) point out, there are currently no accepted or standard assessments for measuring students' CT skills, and only a few studies have been conducted where students' progressive use of different programming artifacts has been studied over time. Our study makes a contribution along both these dimensions. Our CT assessment is based on computational modeling and students' abilities to generate a model or algorithm for a given scenario using computational and domain-specific constructs. Our vector-distance metric, as an extension to the well-known 'bag of words' measure, assesses the accuracy of students' models with respect to an expert model. Using this metric, we show that students' abilities to use domain-based and computational primitives are highly correlated, reflecting the synergy between CT and modeling science phenomena. Our results also suggest that when students find the challenges associated with a modeling activity manageable, their computational models generally improve and become better predictors of their pre-post learning gains.

References

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). Aachen, Germany.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher, 42(1), 38-43.*

Guzdial M. (1995) Software-realized scaffolding to facilitate programming for science learning. *Interactive Learning Environments, 4(1). 1-44.*

Harel, I. (1990). Children as software designers: a constructionist approach for learning mathematics. *The Journal of Mathematical Behavior, 9(1), 3-93.*

National Research Council. (2011). A framework for K-12 Science Education: Practices, Crosscutting Concepts, and Core Ideas. Washington, DC: The National Academies Press.

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

Sengupta, P., Kinnebrew, J.S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating Computational Thinking with K-12 Science Education Using Agent-based Computation: A Theoretical Framework. *Education and Information Technologies, 18*(2), 351-380.

Vattam, S. S., Goel, A. K., Rugaber, S., Hmelo-Silver, C. E., Jordan, R., Gray, S., & Sinha, S. (2011). Understanding Complex Natural Systems by Articulating Structure-Behavior-Function Models. *Educational Technology & Society*, *14* (1), 66–81.

Wilensky, U. 1999. NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

Wing, J. M. (2010). Computational Thinking: What and Why? *Link Magazine*

Acknowledgments

This work was supported by the NSF (NSF Cyber-learning grant #1237350).