Decomposing Students' Design Moves when Programming Agent-Based Models

Adelmo Eloy, Universidade de São Paulo / Columbia University, adelmo.eloy@usp.br
Tamar Fuhrmann, Teachers College, Columbia University, tf2464@tc.columbia.edu
Aditi Wagh, Massachusetts Institute of Technology, awagh@education.mit.edu
Roseli de Deus Lopes, Universidade de São Paulo, roseli.lopes@usp.br
Michelle Wilkerson, University of California, Berkeley, mwilkers@berkeley.edu
Paulo Blikstein, Columbia University, paulob@tc.columbia.edu

Abstract: Constructing computational models is a central practice for K-12 students to refine and validate their understanding of scientific phenomena and develop computational practices. Our work decomposes that practice by investigating the design moves students take while constructing models through programming. Using log data and video recordings from three pairs of students to find patterns in programming actions while interacting with an agent- and block-based modeling environment, we identify six design moves students used. In addition, we situate the occurrence of those moves over time and provide vignettes that illustrate them in context. Finally, we reflect on how our findings support the argument that computational modeling, particularly *programming* models, provides a fertile ground for promoting modeling and computational practices.

Introduction

Constructing computational models is a central practice for concretizing and validating hypotheses, requiring students to express their understanding of scientific phenomena as rules and behaviors (Sengupta et al., 2013; Weintrop et al., 2016; Wilkerson et al., 2015; Xiang & Passmore, 2010). Scholars have examined how constructing computational models supports the learning of computing, with evidence of a varied range of practices and skills, such as debugging and writing code iteratively (e.g., Hutchins et al., 2021; Metcalf et al., 2021; Wagh et al., 2017). Additionally, research has identified challenges students face when constructing models, both computational and science domain-related (e.g., Basu et al., 2016). In this paper, we investigate the *design moves* (sequence of changes in the code informed by the interaction with computational models and real-world data) students make while *programming* agent-based models and describe their relationship to modeling and computational thinking practices. Specifically, we propose the following research question: *what design moves do students make while programming computational agent-based models*?

In this work, we use MoDa ("Modeling + Data", https://moda.education), a web-based computational modeling environment created by the authors and inspired by previous work on domain-specific languages (Wilkerson et al., 2015) and model/real-world data comparison ("Bifocal Modeling," Blikstein, 2014). MoDa lowers the entry point to programming with domain-specific blocks, allowing students to focus on modeling entities' rules and behaviors rather than coding (Fuhrmann et al., 2023; Wagh et al., 2022); and allows for comparison with real-world data. To answer the research question, we analyze log and video data from three pairs of 6th-grade students engaging with MoDa in science class. We identify and describe six prototypical *design moves* students made and reflect on how these moves are related to practices in modeling and computing.

Theoretical background

Our work builds on existing descriptions of constructing computational models as a key modeling practice at the intersection of science and computing. Although both using and developing models have been advocated in K-12 science education (NGSS Lead States, 2013), the benefits of students *creating or developing* rather than *exploring or using pre-built* models have been emphasized by previous work (e.g., Blikstein & Wilensky, 2009; Fuhrmann et al., 2018; Wagh & Wilensky, 2018). In agent-based modeling (ABM), programming models involve encoding entities in the system, their properties, and behaviors in a way that a computer can interpret (Wilensky & Reisman, 2006). Within the ABM paradigm, researchers have adopted block-based and domain-specific programming approaches to lower the threshold for learners to construct computational models (e.g., Wilkerson et al., 2017).

Our work also extends research on identifying computational practices in computational modeling using log data analysis, which is one of the data sources we use in this paper. For instance, Blikstein (2011) employed quantitative techniques to investigate programming strategies from engineering students while writing ABMs of scientific phenomena. Through extensive log data analysis, he identified patterns such as coding in small increments by trial-and-error, building code upon existing examples, and copying and pasting code from external or their own programs. Employing log and video data from two pairs of 7th-grade students, Wagh et al. (2017) defined a construct, namely "anchor code," that represents a stable piece of code for a computational model from which further explorations take place. The authors argued that anchor code also reflects the computational practice of creating modular solutions to a given problem. Likewise, Metcalf et al. (2021) used log-based snapshots from 3rd-grade students to explore the development of computational thinking concepts and practices while designing scientific models in a block-based programming environment. They found evidence of students' increased fluency in concepts such as loops, sequence, and conditionals, along with the enactment of practices such as experimenting, iterating, testing, and debugging computer code. Lastly, by combining log data and discourse analysis from high school students working in groups, Hutchins et al. (2021) identified specific strategies employed by the groups, such as tinkering using multi-visual feedback, and contrasted their use for model-building and debugging.

Building on this existing work, we aim to characterize students' *design moves* - a collection of strategies, or "smaller pieces", that decompose the process of programming models. By combining the analysis of MoDa log interactions with qualitative observation from screen and video recordings, we seek to infer how students enact computational practices in programming models.

Methods

Setting and data sources

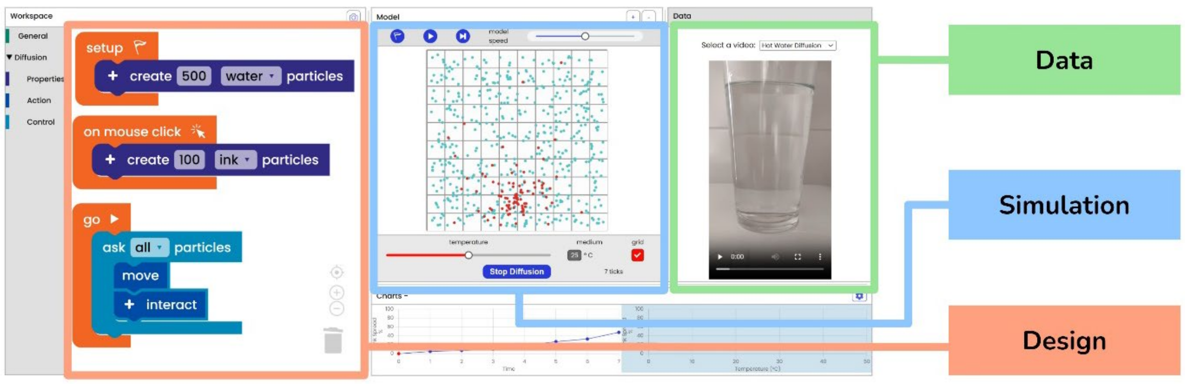
Our focal participants are three pairs of 6th-grade students from three schools in California who engaged with an instructional unit on diffusion. The unit occurred over 6-8 class periods for school but followed the same set of activities. First, students conducted an experiment to compare the rate of ink spread in hot and cold water, and drew "paper models" to explain the difference in the rate of spread across the two conditions. Then, students used MoDa to program models to explain their observations and shared them with the class for feedback and iteratively refined their models over several class periods. Finally, students discussed the validity of their models by comparing them to real-world data and watched a video of the canonical explanation for diffusion.

Data sources included screen and audio recordings from each pair and log records from students' work in MoDa. We selected pairs with the longest screen recording time available in each class. The video length for pairs #1, #2, and #3 were 183, 55, and 59 minutes; while pair #1 interacted with MoDa for 4 class periods, pairs #2 and #3 interacted with it for 3 class periods. Log records captured every action students made while interacting with MoDa through mouse movements, such as clicking the "play" button to start a simulation or dragging a block to delete it. Each action was recorded as an event linked to a specific student account.

MoDa: Modeling environment

MoDa features three main areas (Figure 1) that oriented our analysis. In the *Design* area, built on Google's Blockly, students program ABMs by dragging and dropping a set of domain-specific blocks to create, delete, and refine their models. In the *Simulation* area, built using the NetLogo engine (Wilensky, 1999), students initialize and run their simulations and adjust phenomenon-relevant parameters (e.g., temperature). In the *Data* area, students play and select between real-world videos which, in this unit, were of ink spreading in hot and cold water.

Figure 1
MoDa Design, Simulation, and Data Areas Used to Categorize and Analyze Log Data

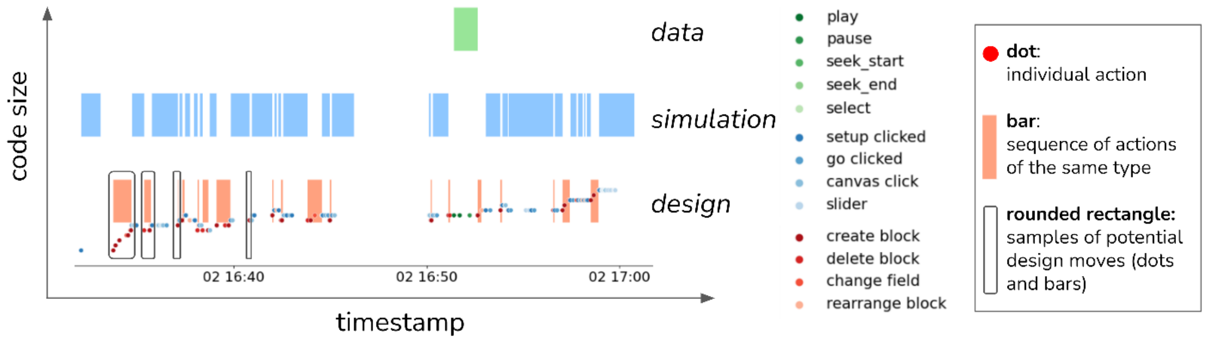


```
setup
  + create 500 water particles
on mouse click
  + create 100 ink particles
go
  ask all particles
    move
    + interact
```

Data analysis

We based our analysis on existing work that uses learning analytics techniques to support qualitative analysis (Sherin et al., 2018) by using log and video data from students. The first author developed a Python script to generate visualizations using students' log data. This script represented students' specific actions over time, categorized according to the three areas in Figure 1. The sample graph in Figure 2 illustrates the visualization: each student's action is presented by a dot, with its x- and y-positions indicating the timestamp and the code size (i.e., number of active blocks) at that moment, respectively. To enhance clarity, a color scheme was implemented to differentiate between categories and actions. In addition, vertical bars were included to help identify sequences of actions from the same category. The definition and criteria for when a bar should "end" varied depending on their respective area and were based on watching a sample of students' screen recording video; design bars ended immediately after a student's last design action, while simulation and data bars were extended until an action of a different area was recorded.

Figure 2
Sample Visualization of Sstudents' Log Data over Time



The first two authors analyzed the graphs from each pair to identify patterns in students' actions around constructing computational models. This analysis was based on the length of each sequence of actions as well as on the actions that immediately preceded and succeeded it. The authors independently identified these patterns; the rounded rectangles in Figure 2 represent four examples of patterns identified in the graph. The first two authors compared and refined their analyses to arrive at a mutual understanding about which patterns to retain for the subsequent stages. Next, they examined video segments corresponding to each of the visual patterns. For each visual pattern, at least one sample was chosen from the three pairs, with pair #1 being the primary source during this phase. The authors collaboratively reviewed these video segments, engaging in in-depth discussion to establish a shared interpretation. For example, the authors noticed that a specific visual pattern in the log data (individual or short design actions followed by interaction with the simulation area) could either resemble students fixing a bug in their model ("debugging") or making small improvements, such as calibrating the number of particles ("fine-tuning"). Although those behaviors could not be distinguished based on the log data only, they were classified as different moves. The action segments derived from this analysis are referred to as "*design moves*". The authors defined each *design move* based on both students' work in the video and how the move was reflected in the log data, associating them with practices identified in prior work. An outcome of this analysis was that each *design move* had a corresponding video fragment, as elaborated in the subsequent section. Finally, the first author expanded the analysis to identify *design moves* within the log data graphs of each pair, utilizing video data when available to validate the categorization. This process resulted in the creation of a visual representation depicting the occurrence of each *design move*.

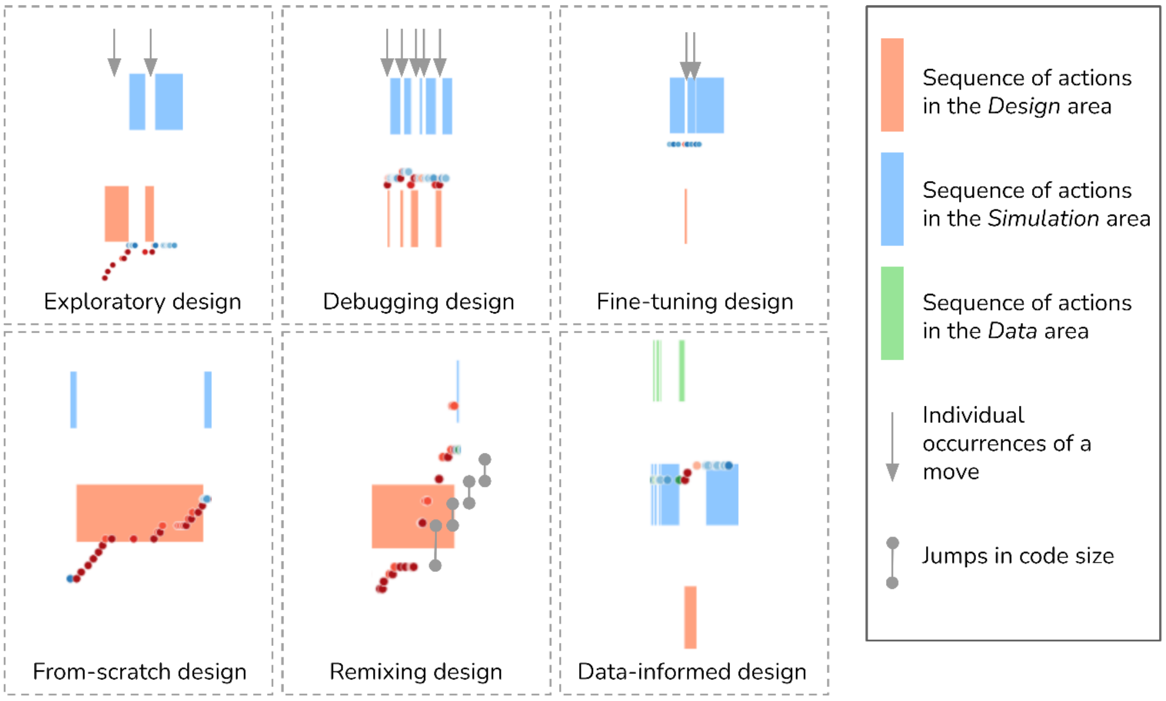
Findings

Based on our analysis, we identified six distinct *design moves* enacted by the three student pairs. Below, we define each move, ordered chronologically based on its occurrence in our data. We then present vignettes to demonstrate the moves in action. In addition, we offer representations of *design moves* over time to indicate when and how frequently these moves took place throughout the unit.

Design moves: Definition, appearance, and context

Figure 3 presents each *design move* and a sample of the visual pattern in the log data that represents them; arrows point to individual occurrences of a move in the same figure, while vertical lines highlight the jumps in code size. We then describe each move and the context in which it was identified.

Figure 3
Sample Visualization of Students' Log data over time



Exploratory design: students make *changes in their code and test them for the first time*. The log data includes 1-4 design actions, followed by a small number of actions in the simulation area. *Exploratory design moves* were often prompted by the unit's introductory challenges and by student-driven experimentation on the role of specific MoDa blocks.

Debugging design: students make *small changes in their code and test them after facing an error or a malfunction in the simulation*. The log data typically includes one design action preceded and succeeded by numerous actions in the simulation area (e.g., setting and resetting the model before running it). Students often debugged their code after getting error messages when the simulation did not work as expected.

Fine-tuning design: students make *small changes in their code and test them out after successfully running their simulation*. The log data includes 1-2 design actions preceded and followed by successfully running the simulation and adjusting its parameters (e.g., temperature). *Fine-tuning* and *debugging design moves* are very similar to each other, differing in what drives small changes in the code. While *debugging design* was driven by unexpected malfunctions, students often fine-tuned their code to improve the model's accuracy.

From-scratch design: students make *large changes in their code, usually starting over a blank screen*. The log data includes a long sequence of uninterrupted design actions, with no clear influence of previous actions. Students usually performed *from-scratch design* when they had issues opening the previous model, or decided to create models for alternative theories.

Remixing design: students make *small changes in their code by copying, pasting, and adapting a chunk of existing code in their model*. The log data includes a jump in the code size during a sequence of design actions, with no clear influence of previous actions. Students often remixed their code when they realized they could design a specific feature or mechanism in their model by adapting a piece of code already available.

Data-informed design: students make *small changes in their code after interacting with video data*. The log data includes 1-3 design actions preceded by actions with both the video and simulation areas. As the description above suggests, students often performed *data-informed design* after comparing their models with real-world video data. In other words, while *debugging* and *fine-tuning* moves were informed by interactions with the simulation, *data-informed design* was informed by the alternation between simulation and video data.

Design moves in action: Vignettes

Below, we provide illustrative vignettes for two *design moves*, based on the data of a single pair of students.

Vignette #1: Exploratory design moves. Student 1(S1) and Student 2 (S2) were exploring MoDa for the first time. At first, they used the design challenges provided by their teacher (T).

1. T:	*So, your goal is to get water particles moving around. (S1 drags the block "ask water particles" and a "move" block to the "Setup" block setting. T then intervenes).*
2. T:	*What do you want to do in the setup?*
3. S1:	*Oops, go and mouse click, go. (S1 drags the blocks to the "Go" setting and drags the block "create water particles" to the "Setup" setting. Then S1 clicks on "setup" and "go" buttons under the Simulation area, to "play" the simulation, which leads to small blue circles moving around. After solving the remaining challenges, S2 goes through the library of blocks and adds "if" and "touching ink particles" under "ask water particles", which is under the "go" setting block).*
4. S2:	*I don't need help. I have an idea.*
5. S1:	*I have an idea, too; I just want to show you.*
6. S2:	*You can explain your idea to me, and I will do it for you. But it is my turn to touch the computer.*
7. S1:	*If touching ink particles, attach particles (S2 drags an "attach particles" block to an "if touching ink particles" condition).*
8. S2:	*No, we don't want them to attach. We want them to spread, not to attach (S2 clicks on "setup" and "go" buttons under the simulation area).*
9. S1:	*Yes, and that is what they're doing.*
10. S2:	*No, it's not, you're guessing.*

This vignette illustrates two instances of an *exploratory design move*. In the first one (lines 1-3), S1 assembled a piece of code with three blocks and tests it in response to a challenge to make water particles move around. In the second instance (lines 4-10), S2 added a conditional behavior into the code, and S1 interjected by asking S2 to test the attached mechanism for particles' spread (represented by the "attach" block).

Those two examples represent situations in which students explored the functionality of the blocks, usually for the first time, either following a task given by the teacher or testing an idea for the first time. In those moments, the students needed to interact with the simulation area immediately after designing to test their code before making new changes, usually another set of exploratory *design moves*. Although the sequence of designing and compiling the code is present in all the *design moves*, the *exploratory design* strongly illustrates the incremental and iterative nature of computational thinking (Brennan & Resnick, 2012).

Vignette #2: Data-informed design moves. T asked S1 and S2 to demonstrate their MoDa model. They played the video for cold water diffusion and initiated the simulation at 0ºC. The teacher asks about a specific part of the code and then comments on their model.

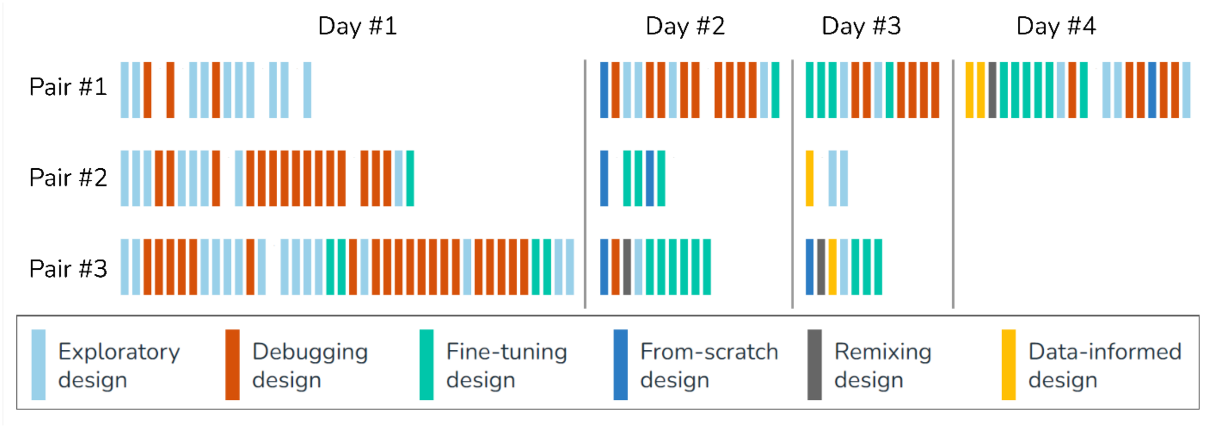
1. T:	*I think you need to do a little tinkering with it because that doesn't look exact.*
2. S1:	*It is not exactly but compare it to the hot water diffusion (S1 changes the video to hot water, resets the model, changes the temperature to 50ºC, and runs the model).*
3. T:	*But your ink is staying in, like, one spot.*
4. S1:	*That is true. I should make it move; it should move.*
5. T:	*Yeah, so think about it. Because you want your ink [to move] (T leaves; S1 adds "ask water particles" and "move" under "setup". S1 sets and runs the model again at different times without playing the video).*

This vignette illustrates *data-informed design*. Prompted by the teacher to compare students' model simulation with the video, students noticed a discrepancy between model simulation and video data and came up with an idea for further modifications. This example illustrates students making changes in their code because of a previous interaction with the video data and the simulation, more specifically, the discrepancies between the real and the model (Fuhrmann et al., 2023).

Design moves over time

Figure 4 presents the occurrence of each *design move* over time for each pair of students analyzed. Note that each rectangle represents a single move, which may encompass various design actions (e.g., *from-scratch design* usually encompassed over ten design actions, while *fine-tuning design* was mostly represented by single design actions in the log data). The blank spaces between *design moves* on each day represent design actions we could not classify into any of the *design moves* we defined (8% of the moves).

Figure 4
Design Moves of Three Pairs of Students over Time



Observing across pairs reveals potential patterns associated with each of the moves. *Exploratory design* was evident on all days, with a notable concentration on the first day of engaging with MoDa; in general, students engaged with multiple rounds of *exploratory design moves* until they got to an error or started polishing their code. This trend aligns with the curriculum's structure, as the initial prompts encouraged students to explore the functionality of various blocks. It also suggests an ongoing experimentation with different ideas throughout the subsequent days, showcasing the diversity of achievable combinations even with a limited set of domain-specific blocks.

On the other hand, *debugging design* moves were not consistent across pairs. While pairs #2 and #3 dealt with a significant number of errors on the first day, pair #1 made a consistent amount of *debugging design moves* throughout the unit. Those preliminary observations might inform different programming profiles, from paying attention to the rules and following instructions to risk-taking and exploring while coding, which can likely benefit from in-depth analyses of the types or errors and solutions. Instances of *fine-tuning design moves* emerged on day #1, but increased on day #2 forward, typically after achieving a stable version of their code, either following successful exploration or after debugging an error.

From-scratch design moves appeared in all pairs, particularly on day #2. Although driven by different motivations, such as an issue of loading a previous model or students deciding to start a new project, *from-scratch design* indicates some levels of mastery with MoDa, usually on students' second day of using the tool. It was usually aligned with the teacher asking students to compare their model against the real-world video data. *Remixing design* was identified in pairs #1 and #3 in similar contexts, that is, when students copied and pasted if-conditions for temperature and speed and adapted parameters for different temperature levels. Finally, *data-informed design* was also identified in all pairs on their last day of MoDa work.

Discussion and implications

Despite the small sample size, our findings identify and describe six distinct *design moves* 6th-grade students made when constructing computational models. These moves differ from each other in the log data's structure and the student's objectives, marking various phases of the computational modeling process. We argue that *design moves* can be interpreted as a collection of strategies that decompose the process of constructing computational models. By unpacking the construction process of computational models, we learned about students' learning in relation to modeling and computational practices. It's worth noting that, in each move, students not only constructed pieces of their models but also tested, evaluated, and revised them, all of which are defined as key elements of modeling (Schwarz et al., 2009; Weintrop et al., 2016). For instance, while *debugging* and *fine-tuning*, students were required to identify potential errors and points for improvement, implement changes in their code and test them out. Likewise, while making *data-informed design*, students evaluated the differences between a model simulation and real-world data to inform potential changes in their code (Fuhrmann et al., 2023). In other words, each of the six *design moves* is intertwined with scientific modeling practices.

On the other hand, the *design moves* also illustrate specific computational practices the context of constructing computational models. For instance, *debugging design* illustrates a practice commonly defined in the computational thinking literature as detecting and identifying errors and then fixing the errors when a solution does not work as it should (Brennan & Resnick, 2012; Hutchins et al., 2021; Shute et al., 2017). Similarly, *fine-tuning design* being performed over a stable version of the model resembles the notion of an *anchor code* for students to build on, as defined by Wagh et al. (2017). *Exploratory design moves* illustrate the iterative experimentation with block-based code in the process of designing computational artifacts (Brennan & Resnick, 2012; Metcalf et al., 2021). Also, comparing *exploratory design* and *from-scratch design moves* resemble different programming patterns students enact while learning to code, such as tinkering and planning (Blikstein et al., 2014). Finally, *remixing design* suggests the ability to adapt existing parts of a model and use it in slightly different contexts by replicating a general code structure and changing specific elements on it (e.g., modifying a parameter or replacing a behavior), which illustrates students' ability to modularize and abstract computer code (Blikstein, 2011; Shute et al., 2017). Although further investigation is required to associate each *design move* with computational practices, the documented *design moves* can serve as preliminary evidence of students performing practices commonly described in the field of computer science. Most importantly, our findings support the argument that computational modeling, specifically constructing computational models, provides a fertile ground for promoting both modeling and computational practices.

To summarize, this paper identifies *design moves* students make for programming scientific models and underscores their relationship with broader practices in modeling and computing. Going forward, we plan to extend our analysis of the *design moves* to larger data sets using quantitative analysis and machine learning techniques like clustering. This could include data from additional instructional units as well as data from more students to enhance generalizability. Extending our analysis across student groups could also deepen our understanding of how factors such as programming expertise might impact students' *design moves*. Nevertheless, this study contributes to the computational modeling literature by unraveling the complexities of students' modeling processes, allowing us to link these steps with the design of resources for scaffolding learning. This equips teachers and designers with more specific strategies to support students in the central practice of constructing computational models.

References

Basu, S., Biswas, G., Sengupta, P., Dickes, A., Kinnebrew, J. S., & Clark, D. (2016). Identifying middle school students' challenges in computational thinking-based science learning. *Research and practice in technology enhanced learning, 11*(1), 1-35.

Blikstein, P. (2011). Using learning analytics to assess students' behavior in open-ended programming tasks. In *Proceedings of the 1st International Conference on Learning Analytics and Knowledge* (pp. 110-116).

Blikstein, P. (2014). Bifocal modeling: Promoting authentic scientific inquiry through exploring and comparing real and ideal systems linked in real-time. *Playful User Interfaces: Interfaces that Invite Social and Physical Interaction*, 317-352.

Blikstein, P., & Wilensky, U. (2009). An atom is known by the company it keeps: A constructionist learning environment for materials science using agent-based modeling. *International Journal of Computers for Mathematical Learning, 14*, 81-119.

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.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association*, Vancouver, Canada (Vol. 1, p. 25).

Fuhrmann, T., Wagh, A., Rosenbaum, L. F., Eloy, A., Wilkerson, M., & Blikstein, P. (2023). How can computational modeling help students shift their ideas towards scientifically accurate explanations?. In Blikstein, P., Van Aalst, J., Kizito, R., & Brennan, K. (Eds.), *Proceedings of the 17th International Conference of the Learning Sciences - ICLS 2023* (pp. 441-448). International Society of the Learning Sciences.

Hutchins, N. M., Snyder, C., Emara, M., Grover, S., & Biswas, G. (2021). Analyzing Debugging Processes during Collaborative, Computational Modeling in Science. In Hmelo-Silver, C. E., De Wever, B., & Oshima, J. (Eds.), *Proceedings of the 14th International Conference on Computer-Supported Collaborative Learning - CSCL 2021* (pp. 221-224). Bochum, Germany: International Society of the Learning Sciences.

Metcalf, S. J., Reilly, J. M., Jeon, S., Wang, A., Pyers, A., Brennan, K., & Dede, C. (2021). Assessing computational thinking through the lenses of functionality and computational fluency. *Computer Science Education, 31*(2), 199-223.

NGSS Lead States. (2013). *Next Generation Science Standards: For states, by states*. Washington, DC: The National Academies Press.

Schwarz, C. V., Reiser, B. J., Davis, E. A., Kenyon, L., Achér, A., Fortus, D., Shwartz, Y., Hug, B., & Krajcik, J. (2009). Developing a learning progression for scientific modeling: Making scientific modeling accessible and meaningful for learners. *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching, 46*(6), 632-654.

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*, 351-380.

Sherin, B., Kersting, N., & Berland, M. (2018). Learning Analytics in Support of Qualitative Analysis. In Kay, J. and Luckin, R. (Eds.) *Rethinking Learning in the Digital Age: Making the Learning Sciences Count, 13th International Conference of the Learning Sciences (ICLS) 2018*, Volume 1. London, UK: International Society of the Learning Sciences.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational research review, 22*, 142-158.

Wagh, A., Fuhrmann, T., Eloy, A. A. da S., Wolf, J., Bumbacher, E., Blikstein, P., & Wilkerson, M. (2022). MoDa: Designing a Tool to Interweave Computational Modeling with Real-world Data Analysis for Science Learning in Middle School. In *Proceedings of Interaction Design and Children*, 206–211.

Wagh, A., Levy, S., Horn, M., Guo, Y., Brady, C., & Wilensky, U. (2017). Anchor Code: Modularity as Evidence for Conceptual Learning and Computational Practices of Students Using a Code-First Environment. In Smith, B. K., Borge, M., Mercier, E., and Lim, K. Y. (Eds.). (2017*). Making a Difference: Prioritizing Equity and Access in CSCL, 12th International Conference on Computer Supported Collaborative Learning (CSCL) 2017*, Volume 2. Philadelphia, PA: International Society of the Learning Sciences.

Wagh, A., & Wilensky, U. (2018). EvoBuild: A quickstart toolkit for programming agent-based models of evolutionary processes. *Journal of Science Education and Technology, 27*(2), 131-146.

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*, 127-147.

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

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.

Wilkerson, M., Shareff, B., Gravel, B., Shaban, Y., & Laina, V. (2017). Exploring Computational Modeling Environments as Tools to Structure Classroom-Level Knowledge Building In Smith, B. K., Borge, M., Mercier, E., and Lim, K. Y. (Eds.). (2017). *Making a Difference: Prioritizing Equity and Access in CSCL, 12th International Conference on Computer Supported Collaborative Learning* (CSCL) 2017, Volume 1. Philadelphia, PA: International Society of the Learning Sciences.

Wilkerson, M., Wagh, A., & Wilensky, U. (2015). Balancing curricular and pedagogical needs in computational construction kits: Lessons from the DeltaTick project. *Science Education, 99*(3), 465–499.

Xiang, L., & Passmore, C. (2010). The use of an agent-based programmable modeling tool in 8th grade students' model-based inquiry. *Journal of the Research Center for Educational Technology, 6*(2), 130-147.

Acknowledgments

This study is supported by funding from the National Science Foundation (#2010413.) We thank our partner science teachers for using MoDa in their classrooms, Jacob Wolf for data collection efforts and for developing the MoDa block library, João Eloy for supporting log data processing, and Leah Rosenbaum for thorough reviewing the manuscript.