



### 3 FRED MARTIN

I am an associate dean and professor of computer science at University of Massachusetts Lowell. As a researcher in K-12 CS education, I was one of the creators of the UMC progression. As the panel moderator, I will introduce the panel and the UMC progression recounting some of the discussions that led to its development [5].

### 4 IRENE LEE

As a researcher at MIT's Scheller Teacher Education Program, I study teachers' and students' development as computational thinkers and the integration of CT into science classrooms. I was a co-creator of the UMC progression [5]. Initially, the goal of UMC was to transform learners from users to creators of computational artifacts. More recently, our goal shifted toward science learning through computer modeling experiences. This shift called for an increased emphasis on decoding in pairs to uncover the mechanisms embedded in computer models, and the evaluation of those mechanisms. To the UMC progression, we add an explicit step, "Decode" or analysis of implementations of scientific process, to increase the potential for gains in scientific understanding through modeling.

### 5 NICHOLAS LYTLE

I am a PhD candidate at NC State University. I have adapted UMC to ease novice K-12 teachers and students into coding within block-based environments. In a comparison study [7], middle-grade science students participated in a multi-day activity where they created a Food-Web simulation with multiple actors (Plants, Bunnies, Foxes). One condition programmed all the actor code each day, while the UMC group instead used pre-built Plant code, modified Bunny code, and created Fox code. The results showed UMC students reported no spike in difficulty over the three days compared to a significant difficulty spike in the control group. Teachers preferred working in these UMC-inspired lessons citing it was better scaffolded for students and provided engaging differentiated tasks. We have extended our approach by adding a "Choose" portion [8], letting students extend their simulation by adding an actor of their choice.

### 6 SUE SENTANCE

I developed PRIMM while a senior lecturer in computer science education at King's College London. PRIMM (Predict, Run, Investigate, Modify and Make) is an approach to structuring programming lessons that counters the problem of novices writing programs before they are able to read them, and focuses on students being able to discuss how and why programs work before they edit and write their own programs. In a PRIMM lesson, students look at a short program in pairs then predict the program's output. They then run the program and discuss how similar their predictions were to the output. Next they investigate the way the program works using a variety of code comprehension questions and activities. Armed with an understanding of the program, students modify the program to increase its functionality, gradually taking ownership. Finally, they make their own new programs, developing design and planning skills. As well as UMC [5], PRIMM draws on research around tracing and reading code before writing [6], the Abstraction Transition

Taxonomy [1] and the Block Model [9]. I will share findings from a recent mixed-methods study of the effectiveness of PRIMM [10].

### 7 NATALIE LAO

I am an EECS doctoral student at MIT. I led the design of and co-instructed MIT's 6.S198: Deep Learning Practicum, a semester-long undergraduate course created at MIT in Fall 2018 that applies UMC to teach actionable machine learning (ML) to novices in computer science (CS) and ML [3]. The course helped a broad range of students gain the ability to ideate and implement independent ML projects. First, students used ready-made ML models within fast-response and user-friendly interfaces to develop high-level intuitions about training, testing, and the importance of data. Then, students manipulated models directly to understand how different architectures, hyperparameters, and datasets impact results for different problems. Finally, students scoped a problem suitable for ML and created their own capstone ML application. To the UMC progression we added a "Choose" that engaged students in selecting suitable machine learning models and algorithms for their application. We saw that the progression helped students deepen their understanding of ML concepts and master practical skills that empowered them to create meaningful capstone projects [4].

### REFERENCES

- [1] Quintin Cutts, Sarah Esper, Marlena Fecho, Stephen R. Foster, and Beth Simon. 2012. The Abstraction Transition Taxonomy: Developing Desired Learning Outcomes through the Lens of Situated Cognition. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research (ICER '12)*. ACM Press, New York, NY, 63–70. <https://doi.org/10.1145/2361276.2361290>
- [2] David Kolb. 1984. *Experiential learning: Experience as the source of learning and development*. Prentice-Hall, Englewood Cliffs, NJ.
- [3] Natalie Lao, Irene Lee, and Hal Abelson. 2019. A Deep Learning Practicum: Concepts and Practices for Teaching Actionable Machine Learning at the Tertiary Education Level. In *IATED2019 Proceedings (12th International Conference of Education, Research and Innovation)*. IATED.
- [4] Natalie Lao and MIT 6.S198 students. 2018. MIT 6.S198 students' final presentations. Retrieved August 27, 2019 from <http://people.csail.mit.edu/hal/deep-learning-practicum-fall-2018/>
- [5] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. Computational Thinking for Youth in Practice. *ACM Inroads* 2, 1 (2011), 32–37.
- [6] Raymond Lister, Elizabeth S. Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, and Robert McCartney et al. 2004. *A multi-national study of reading and tracing skills in novice programmers*. ACM SIGCSE Bulletin 4, 119–150 pages.
- [7] Nicholas Lytle, Veronica Catete, Danielle Boulden, Yihuan Dong, Jennifer Houchins, Alexandra Milliken, Amy Isvik, Dolly Bounajim, Eric Wiebe, and Tiffany Barnes. 2019. Use, Modify, Create: Comparing Computational Thinking Lesson Progressions for STEM Classes. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITICSE '19)*. ACM Press, New York, NY, 395–401. <https://doi.org/10.1145/3304221.3319786>
- [8] Nicholas Lytle, Veronica Catete, Amy Isvik, Danielle Boulden, Yihuan Dong, Eric Wiebe, and Tiffany Barnes. 2019. From 'Use' to 'Choose': Scaffolding CT Curricula and Exploring Student Choices While Programming (Practical Report). In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education (WiPSCE'19)*. ACM, New York, NY, USA, Article 18, 6 pages. <https://doi.org/10.1145/3361721.3362110>
- [9] Carsten Schulte. 2008. Block model: An educational model of program comprehension as a tool for a scholarly approach to teaching. In *Proceedings of the Fourth International Workshop on Computing Education Research (ICER '08)*. ACM Press, New York, NY, 149–160.
- [10] Sue Sentance, Jane Waite, and Maria Kallia. 2019. Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education* 29, 2-3 (2019), 136–176.
- [11] Lev Vygotksky. 1978. *Mind in society: The development of higher psychological processes*. Harvard University Press, Cambridge, MA.