systems

MDPI

Editorial

The Complex Evolution of Technologies and Pedagogies for Learning about Complex Systems

Eric Klopfer

MIT Education Arcade, Massachusetts Institute of Technology, Cambridge, MA 02139, USA; klopfer@mit.edu

It was the mid 1990s. I was doing my doctorate in ecology. When I say that to most people, they picture me out in the field observing lions through binoculars and taking notes in my field notebook. Instead, I spent most of my time staring at a computer screen running simulations to explore various ecological and evolutionary phenomena. It was the early days of complex systems research, and my work relied heavily upon my own low-level computer code to do my work and achieve suitable performance. Eventually, I connected with researchers in diverse fields from entomology to computer science who were exploring similar modeling concepts. An annual gathering at the Santa Fe Institute, one of the premiere institutions for complex systems research, connected me with the leading minds in this exciting new domain. One of those researchers had his own parallel processing machine that could run these kinds of models at what at the time seemed like breakneck speed. However, I had to program my models in the obscure programming language of Forth. If only there had been a simple programming language to accomplish those tasks, I could have graduated at breakneck speed.

A couple of years later, I had finished my PhD work, and I returned to the Santa Fe Institute to work with Mitchel Resnick and his StarLogo team from MIT. He had developed that simple programming language that I had longed for that would make complex systems programming accessible to people who had never programmed before, and now we were going to try to work with teachers to figure out an effective way to integrate this into school practices. Thus began my journey of complex systems modeling tools in schools. I have been pursuing that goal ever since then, having the good fortune to work with many great educators and researchers who share that vision over the years. This issue features some of those wonderful colleagues. I also took over the development of the StarLogo lineage from Mitchel Resnick after I came to MIT and have been leading that project ever since, seeking ways to make it more accessible and more useful to teachers and students. In so doing, I've had the chance to collaborate with and learn about the work of many other colleagues doing great work in complex systems modeling in schools with diverse participants.

This issue covers many important angles in the extension and integration of complex systems and computational modeling in schools. The articles cover systemic issues around the integration of computing tools, preparation of teachers, extension of tools to new domains, and ways of providing students with the insights that they need to make sense of a complex world. These insights point to a promising future and more breakthroughs ahead. However, to understand the significance of these more recent accomplishments, it helps to contextualize them in the history of where we've been, the history that I've followed from complex systems researcher to complex systems educator. Here, I provide some of the history that got us to this point both technically and conceptually, noting that while the technology itself is key to some of this success, it is only part of the puzzle for making a real impact. The majority of the work here that exists alongside the technical development is what leads to meaningful change.

In recent years, the role of modeling has jumped from the domain of scientists to that of everyday experience. Over the last year, we have seen models guiding our everyday lives. We see models of how different masks might disrupt transmission of COVID-19, or

check for updates

Citation: Klopfer, E. The Complex Evolution of Technologies and Pedagogies for Learning about Complex Systems. *Systems* **2021**, *9*, 31. https://doi.org/10.3390/systems9020031

Received: 9 April 2021
Accepted: 25 April 2021
Published: 29 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

how resistance to the vaccine within certain communities could affect "herd immunity". We have seen models used to predict elections and explain past results. However, to many, these models are just something that someone "made up", and are particularly perplexing and counterintuitive when they are models of decentralized complex systems. Understanding and interrogating these models requires experience that most people do not have, and so belief in these models, and the practices and policies that they dictate, has become an act of faith, or lack thereof, to many.

Fortunately, there is a great community working on how to make computational modeling of complex systems accessible and understandable to students and teachers with little or no background in the area, and this community is, in turn, fortunate to be able to build on the great work that has been going on for decades in computing and modeling in schools.

In order to understand where we are now in the computational modeling of complex systems in schools, we need to take a look at the past. That past can be traced back to the work of Seymour Papert and others around the Logo programming language. A recent paper [1] documents that history and the many people that contributed to this breakthrough in both computing and learning at MIT's AI lab in the 1960s. Ultimately, Logo was meant to be a language accessible to young people, through which they could express themselves and develop a deep understanding of computing concepts. Importantly, Logo made the abstract world of bits tangible through what was at first a physical turtle and later came to be a metaphorical digital turtle. Kids could tell the turtle to algorithmically move around on a two dimensional surface and draw a path of where it had been. Subsequent generations of Logo maintained that metaphor as Logo leapt from the lab to the personal computers that started to take off in schools in the 1980s.

As the History of Logo shows, many versions of Logo were spawned from this original ancestor, each with a different purpose. In the early 1990s, Mitchel Resnick, a student of Papert's, saw the emerging scientific domain of complex systems as the next frontier of Logo programming [2,3]. Scientists were using the newfound capabilities of computers to model systems in ways that provided powerful new insights. Modeling moved from aggregates to individuals and from the top down to the bottom up. This provided new insights into the way systems worked, including physical, biological, and social systems. Resnick connected this new way of thinking and modeling to the accessible metaphors of the language of Logo through StarLogo. Resnick brought StarLogo to the masses as he migrated it from the laboratory where it was built on a specialized parallel processing machine (*Logo) to personal computers where it could find its way into schools.

As Resnick states [3], StarLogo provided three important additions to the Logo language. First, it provided for multiple turtles working in parallel. At the time, that number was typically dozens or hundreds of turtles. Second, it provided more sensory functions for those turtles so that they could interact with each other and their "environment". Finally, it provided a structured environment in which those turtles lived on a gridded world of "patches", each of which could act on its own. Through these tools, kids could explore the same complex phenomena that scientists were exploring, and in so doing, start to overcome the "centralized mindset".

For most people, the intuitive explanation for many observed phenomena is that there is a single central source of control. If a flock of birds is moving in unison, then a single bird must be the leader and tell them what to do. If a traffic jam is backing up, then a single car accident must be causing it. Instead, however, these phenomena are often actually explained by a decentralized mechanism. A flock of birds can coordinate through many peer-to-peer interactions. A traffic jam can be caused by the differential acceleration and deceleration of cars on the road. However, these mechanisms are hard to understand. StarLogo allowed kids to easily create models that showed how this phenomena could result from many individuals following a simple set of rules in parallel.

StarLogo itself then spawned a number of offspring, which combined new features with new platforms, from StarLogoT [4] to StarLogo Java [5] and later NetLogo [6]. These

platforms had a lot of success in reaching new audiences. However, for many, these platforms only achieved part of their goals. They made models of complex systems accessible to many people. Teachers *used* models in their classes, but many teachers stopped short of engaging students in *creating* models themselves. This meant students did not develop as deep an understanding of these phenomena as many would have liked, nor did they develop the agency to create and understand new phenomena. At the same time, the independent goal of developing students as computational thinkers started to emerge, and students needed to engage in computing to do this.

In the mid 2000s, seeking ways to make the creation of computational artifacts more accessible, the StarLogo team revisited an idea that had emerged from Resnick's lab in a thesis years earlier [7]: the idea of using graphical programming blocks to represent code instead of text. The effort to do this in StarLogo paralleled Resnick's own pursuit of programming blocks for a new platform, which ultimately became the popular Scratch programming language. For StarLogo, this became StarLogo TNG. StarLogo TNG was designed based on the principle that the blocks programming language would be a first class language, meaning that there was no "text behind the scenes". It was the only way to program StarLogo TNG. It also integrated a 3D world, which had already been incorporated into NetLogo. The 3D perspective allowed students to take the perspective of a single individual within the model, which helped to bridge their personal experiences with the mechanisms of the underlying model.

The blocks programming language made the creation of models much easier for teachers, who did not need to worry about spotting missing commas or spelling errors in their student's code. It also made the modification of models more accessible, an intermediate step between using models and creating models. Lee et al. [8] codified a lifecycle of learning about computational thinking through the lens of "use–modify–create". Students could quickly dive into using models, then start looking under the hood to modify them, and ultimately create their own. The use–modify–create cycle fit perfectly with the blocks' interface, which almost made the process a continuous process, where even "using" a model might involve some simple manipulation of parameters directly in the blocks' code.

Like any long living platform, StarLogo has also had to adapt to the availability of different computing platforms. However, its reliance on performance, being able to simulate hundreds or thousands of turtles, has required some careful engineering as well as patience in waiting for some of those platforms to mature. The initial versions were on a single platform, relying on low level code to optimize speed and eke out every bit of performance on early hardware. The next wave of Java StarLogo and Netlogo rode the wave of "write once and run anywhere" Java code. However, as schools have moved to the web as primary platform, and Chromebooks have become the defacto standard, StarLogo has in turn moved to the web. StarLogo Nova [9] brings the next evolutionary version of the blocks language and 3d world to the native web. In addition to providing for ubiquitous deployment, the web interface also allows for community functionality, including sharing of models and classroom management. It also allows for easy updates as new features are added.

As much as these changes in software have been essential to the evolution of the StarLogo platform and its corresponding accessibility to students and teachers, the history of computing in schools clearly shows that technology alone rarely makes an impact. Impact only comes with the presence of a curriculum that clearly connects to what teachers want to accomplish, professional development that meets teachers where they are, and collaborative communities of designers, practitioners, and researchers who put these things into practice.

Early efforts to build teacher communities around StarLogo can be traced back to the Adventures in Modeling [10] project, which was a collaboration between the StarLogo team at MIT and the Santa Fe Institute, a leading research institution in complex systems. This collaboration engaged a community of teachers from New Mexico in a years-long

process to make modeling of complex systems a pillar of science education at the middle- and high-school level. Adventures in Modeling combined professional development and support with Design Based Research [11]. While the goal imposed by the researchers was to make modeling of complex systems a cornerstone of science learning, the models, tools, and specific goals were co-developed with a community of practitioners, many of whom served as design partners.

The Adventures in Modeling work planted firm roots in New Mexico, and the community has remained at the forefront of this work for more than two decades. After Adventures in Modeling ended, Project GUTS (Growing Up Thinking Scientifically) (Lee as cited in [12]) and Teachers with GUTS continued to work with the community and position computational modeling of complex systems as a key approach for developing computational thinking in and around schools, while giving students great agency in their work. The Project GUTS curriculum has become the standard for connecting computing to science in schools through its positioning in the Code.org curriculum.

As much as the specific tools have persisted, the community of practitioners approach of combining professional development and Design-Based Research has been equally as important a legacy of this work, which is clearly shown by the papers in this issue. As computational modeling advances and enters new arenas, this collaborative and practical approach is essential to having an impact. Another aspect of that original approach that has been essential is collaboration with other researchers with diverse expertise. While the original Adventures in Modeling work engaged researchers in complex systems, as the work has continued, it has engaged other science disciplines and researchers in other educational subdisciplines, from language acquisition to professional learning. Again, these efforts are reflected here by the deep and diverse perspectives of many education thought leaders.

The papers in this issue reflect the important findings of the past, engaging in collaborative Design-Based (Implementation) Research that positions computational modeling in new ways; at the same time, it points the way forward, examining the systemic issues required to integrate computational modeling into schools. I see the papers in this issue making important contributions to pointing the way forward in both research and practice supporting computational modeling in schools.

Both Pierson and Brady and Haas et al. focused their work on English language learners. They used computational modeling as a form of expression for systems thinking in elementary-school-aged students, many of whom were learning English. Their studies show some promising successes in this regard, in some ways decoupling developing language skills from the ability to express complex scientific ideas. Language can often be a barrier for expression in other domains, and being able to utilize technology to decrease this barrier could help address the needs of many learners. Additional research and development in this area can help identify best practices for how to represent phenomena with more limited language, including which words to use, how to express concepts visually, and how to connect to other forms of modeling.

Pierson and Brady additionally focus on the role of embodied modeling in sensemaking. In agent-based modeling, connecting the individual perspective to the systems-level outcomes is a key component to sensemaking, but this is often difficult to do. Using on- and off-screen activities helps students take different perspectives and understand the inputs and outputs of these models. From a technology perspective, we are coming to a time when on- and off-screen boundaries are blurring, with mobile technologies, mixed reality, and more immersive computing. These are exciting interfaces to explore to help further connect multimodal perspectives.

Cottone et al. look at the usability of innovations and how that helps or hinders their integration into schools. That goal of ease of implementation has been an important driver of innovation on the technical side. Many educational technologies, StarLogo Nova included, have migrated from installed software to web-deployed environments. Along with that shift in platform has come an emphasis on increasing the value and decreasing the

management challenges for teachers. As these technologies advance, we should seek ways to increase capability, connect better with school culture, and ease policy/management.

Finally, Marei et al. examine teacher learning in an online environment where they are learning about computational modeling. The role of feedback and community development within these online learning environments is key to teacher learning. It also suggests that we can and should have better integration of the computational modeling environments themselves with community and feedback. Both students and teachers could benefit from this integration.

With the great need to make computational modeling accessible to all, these papers and these findings help provide a roadmap toward making that possible.

References

1. Solomon, C.; Harvey, B.; Kahn, K.; Lieberman, H.; Miller, M.L.; Minsky, M.; Papert, A.; Silverman, B. History of Logo. *Proc. ACM Program. Lang.* **2020**, *4*, 1–66. [CrossRef]
2. Resnick, M. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*; MIT Press: Cambridge, MA, USA, 1994.
3. Resnick, M. Beyond the Centralized Mindset. *J. Learn. Sci.* **1996**, *1*, 1–22. [CrossRef]
4. Wilensky, U. StarLogoT; [Computer Software]. 1997. Available online: https://education.mit.edu/project/starlogo-tng/ (accessed on 28 April 2021).
5. Klopfer, E.; Begel, A. StarLogo under the hood and in the classroom. *Kybernetes* **2003**, *1/2*, 15–37. [CrossRef]
6. Tisue, S.; Wilensky, U. Netlogo: A simple environment for modeling complexity. *Int. Conf. Complex Syst.* **2004**, *21*, 16–21.
7. Begel, A. LogoBlocks: A Graphical Programming Language for Interacting with the World. Unpublished Advanced Undergraduate Project Report, MIT Media Lab. 1996.
8. Lee, I.; Martin, F.; Denner, J.; Coulter, B.; Allan, W.; Erickson, J.; Malyn-Smith, J.; Werner, L. Computational thinking for youth in practice. *ACM Inroads* **2011**, *1*, 32–37. [CrossRef]
9. MIT Scheller Teacher Education Program. StarLogo Nova [Computer Software]. 2019. Available online: https://education.mit.edu/project/starlogo-nova/ (accessed on 1 March 2021).
10. Colella, V.S.; Klopfer, E.; Resnick, M. *Adventures in Modeling: Exploring Complex, Dynamic Systems with StarLogo*; Teachers College Press: New York, NY, USA, 2001.
11. Brown, A.L. Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *J. Learn. Sci.* **1992**, *2*, 141–178. [CrossRef]
12. Ridgway, R. Project GUTS. *Sci. Scope* **2018**, *3*, 28–33.