Introduction to Computational Modeling in Schools

- Computational modeling at school

Computational modeling at school

The importance of computational modeling and simulations in education

In science, understanding the world requires thinking with **models**—simplified representations of reality that help us explain, predict, and explore phenomena. Computational modeling and simulations bring this fundamental scientific practice into education, allowing students not only to observe but to actively engage in the processes of hypothesis formulation, experimentation, data analysis, and argumentation. By integrating these tools into the classroom, educators can foster scientific thinking and inquiry-based learning, equipping students with skills essential for both academia and real-world problem-solving.

One of the greatest challenges in education is helping students grasp abstract concepts. Computational models play a crucial role in this process by offering a structured way to simplify and explore complex ideas. Through modeling, students learn that scientific understanding is not about memorizing isolated facts but about constructing and refining representations of reality.

In one of his short stories, "**Funes the Memorious**", the famous Argentine writer **Jorge Luis Borges** (1899 - 1986) presents a striking analogy for why **abstraction** is essential for thought. Funes, endowed with a perfect memory, is incapable of generalizing or abstracting, as he recalls every detail of every moment with absolute precision. His inability to forget prevents him from forming concepts or making connections, leaving him trapped in a world of pure recollection rather than true thinking.

Similarly, without the ability to simplify and model, scientific thinking becomes impossible. Models allow students to filter out unnecessary details, identify patterns, and develop conceptual frameworks, rather than getting lost in an overwhelming amount of data. Abstraction is at the heart of modeling, as it enables us to focus on what truly matters while ignoring irrelevant complexities. Borges himself captured this idea powerfully:

> *With no effort, he had learned English, French, Portuguese and Latin. I suspect, however, that he was not very capable of thought. To think is to forget differences, generalize, make abstractions.*
>
> (Borges, 1998, p. 154)

By engaging with computational models, students move beyond **passive learning** and start **acting like scientists**. They develop scientific skills by *formulating hypotheses*, *adjusting variables*, and *predicting outcomes* ; *conducting experiments* to test different scenarios; *analyzing data* to observe patterns and relationships; and revising their models based on new evidence, reinforcing the iterative nature of scientific inquiry, among other skills.
Moreover, an essential lesson emerges: all **models** are **approximations**. No model can perfectly capture reality; instead, it highlights certain aspects while omitting others. Understanding this limitation helps students develop a critical mindset, making them more aware of the assumptions behind scientific theories and real-world data interpretations.

The Power of Simulations

While models allow us to think abstractly, simulations allow us to experiment safely. Many real-world scientific experiments are:

- Too dangerous (e.g., studying nuclear reactions).
- Too expensive (e.g., sending spacecraft to different planets).
- Impossible (e.g., modeling the evolution of an ecosystem over thousands of years).

Simulations provide an alternative by creating controlled, **interactive environments** where students can explore "what-if" scenarios that would otherwise be unattainable. A well-designed simulation enables learners to manipulate variables, test hypotheses, and see the effects of their choices in real-time, reinforcing causal reasoning and scientific literacy.

Beyond Science: Computational Modeling Across Disciplines

While commonly associated with the **natural sciences**, computational modeling extends beyond physics, chemistry, and biology. In social sciences, simulations help students explore economic models, population dynamics, and decision-making processes. In humanities, **historical simulations** can illustrate how small changes in events might have led to entirely different outcomes. Interdisciplinary learning becomes more tangible when students can manipulate and interact with models rather than simply reading about theories.

Integrating **computational modeling and simulations** into education transforms students from passive receivers of information into active investigators of knowledge. By **thinking with models**, learners develop essential scientific skills—formulating hypotheses, analyzing data, and refining their understanding—while also recognizing the limitations inherent in any model. **Simulations**, in turn, allow them to experiment beyond the constraints of reality, testing ideas in ways that would otherwise be too risky, expensive, or impossible.

For educators, these tools provide **a bridge between theory and practice**, helping students develop critical thinking, problem-solving abilities, and a deeper appreciation for the scientific process. In an era where data and computational reasoning shape our understanding of the world, **equipping students with these skills** is more essential than ever.

StarLogo NOVA is an **accessible** and **powerful** tool that enables teachers and students, even those with no prior programming experience, to *use, modify, and create* (Lee, 2018) computational simulation models. Designed specifically for educational settings, StarLogo NOVA **scaffolds** the processes of **modeling and simulation**, offering a 3D, fun, game-oriented environment that fosters engagement and creativity. Rooted in the principle of providing a "*low threshold and high ceiling*" (Papert, 1980), it ensures that beginners can easily get started while still allowing for advanced exploration and deeper learning.

The History and Evolution of Computational Modeling

The origins of **computational modeling** in education are closely tied to the development of **educational programming languages**, particularly **LOGO**, created by Seymour Papert and his colleagues at MIT in the late 1960s (Papert, 1980). LOGO was designed as a **tool for learning through exploration**, embodying Papert's vision of **"constructionism"**—the idea that students learn best when they actively construct knowledge rather than passively receive it. Through the use of **turtle graphics**, LOGO allowed students to visualize geometric patterns by issuing simple commands, fostering **computational thinking** long before the term became widely recognized. Papert argued that **"computers are instruments whose music is ideas"**, emphasizing their potential to revolutionize learning (Papert, 1980, *Mindstorms: Children, Computers, and Powerful Ideas*).

As computational modeling evolved, the development of **domain-specific programming languages** for education gained traction. **StarLogo**, an extension of LOGO developed by Mitchel Resnick in the 1990s, shifted from individual **turtle-based commands** to **agent-based modeling**, enabling students to simulate **complex systems** such as flocking behaviors or predator-prey interactions (Resnick, 1994, *Turtles, Termites, and Traffic Jams*). This marked a shift in educational programming, where the focus moved from controlling single agents to **modeling emergent behaviors in decentralized systems**. By manipulating simple local rules, students could observe how global patterns emerged—an essential concept in both science and computational thinking.

The rise of **block-based programming environments** in the early 2000s, such as **Scratch** (Resnick et al., 2009) and later **StarLogo NOVA**, further democratized access to computational modeling by making programming more **intuitive and visual**. These platforms eliminated **syntax barriers**, allowing learners to focus on the **logic of computation** rather than on the specifics of text-based coding. This shift reflects a broader educational trend: rather than training students solely in coding, **modern computational modeling environments aim to develop broader problem-solving skills** applicable across disciplines (Grover & Pea, 2013, *Computational Thinking in K-12: A Review of the State of the Field*).

Today, computational modeling is not only a tool for learning programming but also a **gateway to scientific inquiry**, enabling students to construct and test hypotheses in virtual environments that mirror real-world phenomena. The increasing integration of **machine learning and data science into educational platforms** continues to push the boundaries of how students engage with models, reinforcing the idea that computational modeling is not just about **learning to program**—it is about **learning to think**. As Wing (2006) famously stated, **"computational thinking is a fundamental skill for everyone, not just for computer scientists"**, a perspective that has only grown more relevant in modern education (*Computational Thinking*, Communications of the ACM).

StarLogo NOVA as a Computational Modeling Tool

"**The whole is more than the sum of its parts**" is a well-known phrase, popularized in the early 20th century by the Gestalt school of psychology to highlight that our perception is greater than the raw sensory information we receive.

Marvin Minsky (1986) references this idea in **The Society of Mind** through a dialogue between a holist and an ordinary citizen about **how a wooden box can contain a mouse**. The holist argues that since none of the six individual boards that make up the box can hold the mouse on its own, the box itself should not have that property either. In response, the citizen points out that the mouse still cannot escape, prompting the holist to suggest that a good box simply creates the illusion that it has this capacity. As a result, the mouse—deceived—believes it cannot escape and remains inside.

At the end of the dialogue, Minsky (1986, p. 27) concludes that the box's ability to contain the mouse does not reside in any single part but rather in the interaction between them. Each side of the box contributes to the overall function: the left side prevents the mouse from escaping in that direction, the right side does the same, and so on. The capacity of the box emerges from the way its components work together as a whole—a principle that Minsky extends to the functioning of the mind. Although, he notes, in 1986 it was still unclear how "*mental agents*" interact to achieve what we perceive as unified thought and cognition.

Complex Systems: A New Perspective

Examples of complex systems—where the whole is greater than, more intricate, or even more powerful than the sum of its parts—can be found everywhere: Termite nests; the climate system; stock market fluctuations; the spread of fires and epidemics; ecosystems and bird flocks; the human mind.

As Mitchell Waldrop (1992, p.10) puts it:

> What is a mind? How can it be that a three-pound lump of ordinary matter, the brain, gives rise to such ineffable qualities as feeling, thought, purpose, and consciousness?

While the study of **complex systems** dates back to the 1960s, the ability to analyze them computationally is more recent. Traditional analytical methods struggle to capture the intricate **interactions between system components**. This challenge has driven the rise of two computational approaches for studying complex systems:

1. System Dynamics (SD)

2. Agent-Based Modeling (ABM)

The approach differs significantly between them:

• In **System Dynamics**, aggregate variables represent an entire population. For example, in a grazing ecosystem with sheep and grass, parameters such as the reproduction rate of the herd are defined at the global level.

• In **Agent-Based Modeling**, individual agents (such as sheep) follow local rules, and global patterns emerge from their interactions (Resnick, 1994, p.64).

These approaches are not mutually exclusive—they are used interchangeably, depending on the characteristics of the system and the study objectives.

Education and Complex Systems Thinking

Many of today's global challenges—such as epidemics, climate change, ethnic segregation, and biodiversity loss—can be explored through the lens of complex systems. These issues are too intricate to be understood through simple cause-and-effect relationships, making computational models essential for analysis and decision-making.

At the school level, both teachers and students can benefit from these tools to analyze complex problems by using, modifying, and creating agent-based simulations.

Among the most accessible tools for schools are StarLogo NOVA and NetLogo with NetTango, both of which were designed specifically for educational use. These platforms empower students to visualize, experiment with, and understand complex systems, fostering a deeper appreciation for emergent behavior and scientific inquiry.

Aquí tienes una versión refinada y más clara de tu texto, con mejoras en la fluidez, la estructura y el tono académico:

Computational Models

Up to this point, we have discussed **complex systems** and **agent-based modeling**, the latter being a powerful tool for exploring these systems. However, we have yet to fully define these concepts.

What Is a Model?

The first term to clarify is *model*, one of the most **polysemous words** in the English and Spanish languages. In everyday speech, *model* can refer to a **fashion model; a person or organization** that serves as an example to follow; a **scale model of an airplane; Bohr's atomic model; a solar system representation** made of Styrofoam balls and wire, to name just a few common uses. In this book, we adopt a **simple, yet broad and powerful** definition of *model*, coined by **Marvin Minsky**:

>

> *For an observer B, an object A' is a model of an object A to the extent that B can use A' to answer questions he is interested in about A." (Minsky, 1965, p. 1)*

To be even more precise, we will focus on **relational models**, distinguishing them from **pictorial models**.

• **Pictorial models** aim to **visually represent** carefully chosen attributes of objects and depict **observable interactions** between them.

• **Relational models**, on the other hand, emphasize **relationships that are not easily observable** in the real world (Snir et al., 1993).

Based on this distinction, **computational modeling** can be understood as **the process of constructing (or reconstructing) a relational model**—in Minsky and Snir's terms—**within a computer environment**.

Defining the Agent in Agent-Based Modeling

Now that we have discussed **complex systems**, defined what we mean by **model**, and clarified the concept of **computational modeling**, the next step is to define what we mean by **agent**.

An **agent** is an **autonomous entity**—a computational object—that has specific **properties** and **actions**. **Agent-based modeling (ABM)** is a computational modeling approach in which a phenomenon is simulated in terms of **agents and their interactions** (Wilensky & Rand, 2015).

A key clarification is that **these interactions occur both between agents and between agents and their environment**. Furthermore, in agent-based modeling, **agents can belong to different types or categories**, commonly referred to as **breeds**.

Thus, **creating a relational computational simulation model** using an **agent-based modeling tool** is, fundamentally, **programming the relationships that govern these interactions**.

StarLogo: A Tool for Agent-Based Modeling

StarLogo was developed specifically for this purpose. It follows the tradition of its predecessor, **LOGO**, but while the original LOGO focused primarily on **turtle geometry**, **StarLogo is designed for computational simulations** , particularly for the creation of **agent-based models**.

In the words of **Mitchel Resnick**, in his book *Turtles, Termites, and Traffic Jams*:

> *“ I did not want users to merely manipulate parameters in a standardized application program; I wanted them to build and modify their own programs, exploring situations of interest to them.*
>
> Resnick, 1994, p. 60

This quote encapsulates one of the **main objectives** of this book: to introduce StarLogo NOVA as a **programmable environment** that enables both **teachers and students** to **create their own relational simulation models on a computer, using an agent-based programming approach**.

Computational Thinking in the Teaching of Natural Sciences and Mathematics

The **corollary** of the previous section leads us to the **central focus of this book**: exploring tools and **an approach** based on the **philosophy of the LOGO language** (as part of the **Papertian framework** mentioned earlier) to promote the **integration of computational thinking and programming in the teaching of natural sciences**.

Computational Thinking in STEM Education

In **2016**, **David Weintrop** and a research team from **Northwestern University**—including **Uri Wilensky**, the creator of **NetLogo**—published an article in the *Journal of Science Education and Technology*. In it, they presented a **taxonomy of computational thinking practices** for STEM education, emphasizing **"the decision to include computational thinking as a central scientific practice"** (Weintrop et al., 2016).

Their taxonomy categorizes **computational thinking practices** into four broad areas:

1. **Practices with Data (PCD)**

2. **Modeling and Simulation Practices (PMS)**

3. **Computational Problem-Solving Practices (PRP)**

4. **Systems Thinking Practices (PPS)**

To establish these categories, they conducted an extensive **literature review** on computational thinking, analyzed **teaching materials developed by educators**, and consulted **experts in science and technology disciplines**.
The relevance of Weintrop's work for this book lies in how it **clarifies the different ways computational thinking can be applied in STEM education**.

Computational Thinking Practices Relevant to This Book

For our purposes, we will focus on the following computational thinking practices (with their corresponding taxonomy category in parentheses):

- **Using computational models to understand a concept (PMS)**

- **Computational model design (PMS)**

- **Construction of computational models (PMS)**

- **Investigating a complex system as a whole (PPS)**

- **Creating computational abstractions (PRP)**

- **Programming (PRP)**

- **Data visualization (PCD)**

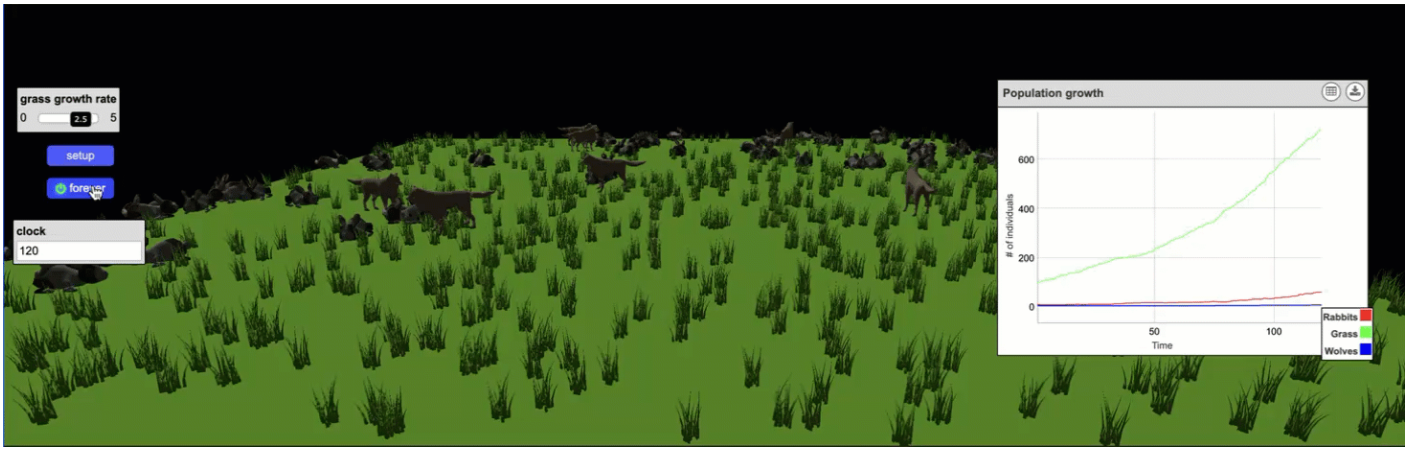
Implementation Through Agent-Based Modeling Tools

StarLogo NOVA provide **intuitive, interactive environments** where students and educators can engage in **modeling, programming, and simulation**, fostering a deeper understanding of **scientific and mathematical concepts**.

StarLogo NOVA is a **programmable agent-based modeling environment** designed primarily for the **creation of computational simulation models** in educational settings.

One of its distinguishing features is its **three-dimensional interface**, which allows for more dynamic and immersive simulations. **StarLogo NOVA is entirely web-based**, meaning that users can access it directly from a browser without needing to install additional software. This makes it **highly accessible**, particularly in educational environments where installing software can be a barrier. And also StarLogo NOVA is **free to use**.

Below is an animation of **StarLogo NOVA** running an **ecosystem simulation model consisting of rabbits, grass, and wolves** (2021):



In the chapter **"Simulations and Stimulations,"** we will use **StarLogo NOVA** as a modeling tool to explore various natural phenomena through **agent-based simulation models**, using **block-based programming** as ***" objects to think with"*** (Turkle, 1986). Each challenge will serve as an opportunity to view the world around us through a new lens, utilizing a powerful tool that challenges us, fosters **critical thinking**, encourages **experimentation**, promotes **hypothesis development**, and, ultimately, helps us **engage in scientific exploration**.

Each challenge is accompanied by **explanations, screenshots, animations, and videos** to support your journey in understanding nature through the **use, modification, and creation of computational simulation models**.

Designing New Modeling Challenges

When designing **new modeling challenges**, it is essential to choose **natural phenomena that involve surprising interactions** and can serve as an entry point for teaching **important scientific concepts**. A well-designed challenge should engage students in **exploring emergent behaviors**, where individual agent interactions give rise to unexpected patterns at a larger scale. For example, ecological systems, predator-prey dynamics, disease spread, or social behaviors in animal groups provide **rich opportunities for computational modeling**. The key is to **identify interactions that defy intuition**, prompting students to investigate why certain patterns emerge and how small changes in individual behavior can lead to significant global effects. These challenges should also **balance structure and openness**, guiding students toward meaningful discoveries while allowing space for **creativity and experimentation**.

Assessing and Providing Feedback on Students' Models

For a modeling challenge to be **educationally effective**, it must be anchored in **key curriculum concepts** that students are expected to learn. Assessment should go beyond simply checking whether a model "works"—it

should evaluate **students' understanding of the underlying scientific principles**. Providing feedback involves helping students reflect on their models, asking questions like: *Does the model accurately represent the phenomenon? What assumptions were made? What patterns emerge, and how do they compare to real-world data?* Encouraging students to **iterate and refine their models** fosters deeper learning and strengthens their computational thinking skills. Assessment strategies can include **rubrics that evaluate conceptual accuracy, computational implementation, and explanatory depth**, as well as peer review processes where students critique and improve each other's models.

Encouraging Collaboration and Teamwork

Computational modeling is not just an individual endeavor—it thrives on **collaboration, discussion, and diverse perspectives**. While students can work independently, **pair programming** and **group-based problem-solving** have been shown to enhance both learning outcomes and engagement (Werner et al., 2004). Working in **pairs or small teams** allows students to verbalize their thought processes, debug more effectively, and develop stronger problem-solving skills. Beyond direct collaboration, sharing models with peers for **feedback and discussion** is equally valuable. By presenting their models, students learn to **articulate their design choices**, compare different approaches, and refine their thinking. It is also crucial to emphasize that **there are no single correct answers in agent-based modeling**—students may take different paths to reach similar outcomes, and **even unintended explorations can lead to creative breakthroughs**. Encouraging an environment where **divergent thinking and "going off track" are seen as valuable learning experiences** helps foster **innovation and resilience** in problem-solving.