MILA–S: Generation of Agent-Based Simulations from Conceptual Models of Complex Systems

David A. Joyner
Design & Intelligence Laboratory
School of Interactive Computing
Georgia Institute of Technology
85 5th Street NW
Atlanta, GA 30308
david.joyner@gatech.edu

Ashok K. Goel
Design & Intelligence Laboratory
School of Interactive Computing
Georgia Institute of Technology
85 5th Street NW
Atlanta, GA 30308
goel@cc.gatech.edu

Nicolas M. Papin
Design & Intelligence Laboratory
School of Interactive Computing
Georgia Institute of Technology
85 5th Street NW
Atlanta, GA 30308
npapin3@gatech.edu

ABSTRACT
Scientists use both conceptual models and executable simulations to help them make sense of the world. Models and simulations each have unique affordances and limitations, and it is useful to leverage their affordances to mitigate their respective limitations. One way to do this is by generating the simulations based on the conceptual models, preserving the capacity for rapid revision and knowledge sharing allowed by the conceptual models while extending them to provide the repeated testing and feedback of the simulations. In this paper, we present an interactive system called MILA–S for generating agent-based simulations from conceptual models of ecological systems. Designed with STEM education in mind, this user-centered interface design allows the user to construct a Component-Mechanism-Phenomenon conceptual model of a complex system, and then compile the conceptual model into an executable NetLogo simulation. In this paper, we present the results of a pilot study with this interface with about 50 middle school students in the context of learning about ecosystems.

Author Keywords
STEM education; K6-12 education; complex systems; ecological systems; conceptual models; agent-based simulations; scientific inquiry.

ACM Classification Keywords
1.2.m **[Artificial Intelligence]**: *Miscellaneous*. K.3.1 **[Computer Uses in Education]**: *Computer-assisted instruction.* J.3 **[Life and Medical Science]**: *Biology and genetics*. I.6 **[Simulation and Modeling]**: *Miscellaneous*.

General Terms
Design; Human Factors; Experimentation.

IUI'14, February 24–27, 2014, Haifa, Israel.

ACM 978-1-4503-2184-6/14/02...$15.00.
http://dx.doi.org/10.1145/2557500.2557516

INTRODUCTION
In the course of scientific inquiry, scientists use multiple approaches to specify, share, test, and critique their ideas [9]. Two of the most important approaches scientists use are the construction of conceptual models and the execution of simulations of the system of interest. Conceptual models are abstract, declarative representations of the components, relations, and processes of the system [1, 2, 8, 9, 16]. A conceptual model specifies a scientist's current understanding of a system, allowing externalization, sharing, and critiquing of that understanding; further, it facilitates the use of the model to guide further investigation. Most theories of scientific modeling operationalize the process in terms of multiple phases, including model construction, model usage, model evaluation, and model revision[1, 2, 5, 9, 13].

Simulations can take several forms, including mental, physical, mathematical, and computational [1, 3, 6, 9, 16]. Like models, they specify scientists' current understanding of the natural system. Simulations, however, are executable with specific values for the system's input variables to determine how the values of the system's output variables may evolve over time. If the simulation behaves differently than the actual system under the same parameters, the scientist may modify the simulation to account for the discrepancy. In this way, the process of simulation construction, use, evaluation and revision is parallel to the process of conceptual modeling outlined above. Indeed, many theorists see conceptual models and simulations simply as two different kinds of models [9, 19].

Conceptual models and simulations each have unique affordances and limitations [1, 3, 6, 9]. Conceptual models lend themselves to rapid construction, evaluation, and revision because of their abstract, declarative, often qualitative, nature. However, conceptual models are limited in their precision, detail, and rigor. Executable simulations tend to be more precise, detailed, and rigorous. However, simulations are not typically as suited to rapid construction, use, and revision as conceptual models. Thus, it is useful to leverage the affordances of conceptual models and simulations to mitigate their respective limitations. One way to do this is by generating the simulations based on the

conceptual models, preserving the capacity for rapid revision and knowledge sharing allowed by the conceptual models while extending them to provide the repeated testing and feedback of the simulations. In order to facilitate the use of simulations in the modeling process, it is necessary to devise a technique by which the simulations can respond directly to changes in the conceptual models. Our goal in this paper is to report on an interactive technology that can translate users' conceptual models into computational simulations.

In this paper, we focus on ecological systems as the systems of interest, causal models that provide causal explanations for a phenomenon as the conceptual models, agent-based computational simulations as the executable simulations, and STEM students as the users. Ecological systems are comprised of many individual agents, and the observable phenomena typically are the outcomes of localized interactions among the agents. Thus, agent-based simulations tend to be well-suited for simulating ecological systems [18]. Causal models of ecological systems often emphasize a top-down approach to analyzing a system, starting with a phenomenon to be explained, and developing increasingly refined explanations of the phenomenon in terms of mechanisms and components. Vattam, Goel & Rugaber describe a method for integrating Structure-Behavior-Function causal models [4] and agent-based simulations of ecological systems in the context of an interactive learning environment called ACT for supporting middle school science [14, 15]. Their method used the notion of "behavior patterns:" first the user-generated causal model was classified into a pattern of behavior in ecological systems, and then the pattern was translated into the agent-based simulation; the interactive tool had prior knowledge of how to map behavior patterns into input variables for the simulations. In practice, that method was quite complicated and was not much used in actual middle school classrooms.

In this paper, we describe an interactive technology called MILA–S for integrating causal models and agent-based simulations of ecological systems. 'MILA' stands for Modeling & Inquiry Learning Application; the '–S' is an additional simulation layer built on top of MILA. In this paper, we present the design of this system and the process by which MILA–S converts a conceptual model into an agent-based simulation. We also present preliminary results from a pilot study with MILA–S in middle school science classrooms in near Atlanta, USA, and its use by about 50 students for modeling a local aquatic ecosystem.

DESIGN OF THE INTERACTIVE TECHNOLOGY

MILA–S has three main components. (1) A modeling tool for constructing a causal model for explaining an observed phenomenon in an ecological system. (2) An off-the-shelf tool called NetLogo [17, 18] for constructing agent-based simulations of ecological systems. (3) A translational bridge between the causal model and the simulation, whereby MILA–S automatically generates an agent-based simulation based on the contents of the user specified causal model.

Causal Models

MILA–S uses Component-Mechanism-Phenomenon (or CMP) causal models that are variants of Structure-Behavior-Function models used in ACT. Components in CMP modeling can be either biotic (that is, living, such as plants, fish, or bacteria) or abiotic (that is, non-living, such as chemicals, dead matter, or sunlight). Each component has a set of variables associated with it, four for biotic components and one for abiotic components. Biotic components are defined by their population quantity, lifespan, energy level, and likelihood to breed; abiotic components are defined only by their quantity. Figure 1 illustrates a causal model constructed by a team of 7th grade life science students early in their interaction with MILA–S. In this model, there are three components: Sunlight, Oxygen, and "Fishies". The Sunlight and Oxygen are abiotic components, and they have only Amount as a variable that is designated on the node for the component. "Fishies" is a biotic component, and thus has Population, Age, Birth Rate, and Energy as variables; Population is designated on the "Fishies" node itself, while the notations for the other three variables extend downward from the main node.

CMP modeling draws causal relations between the variables associated with the different components. For example, the presence of a chemical like ammonia in the ecosystem that is poisonous to fish may decrease the

Figure 1: A causal model constructed by a team of 7th grade students using MILA–S.

lifespan of the fish, or it may directly decrease the population of the fish (additional information on the difference between the two is provided later in this paper). MILA–S provides the user with a set of prototypes that describe causal relationships among system variables. The choice among the available prototypes is determined by the variables on either end of the relation and the type or direction of the relation. For example, a relation from the Population of a biotic component to the Amount of an abiotic component, such as that from Fish Population to Oxygen Amount, may be 'consumes', 'produces', or 'becomes upon death,' etc. Similarly, a relation from an abiotic Amount to a biotic Population may be 'destroys' or 'feeds'. Similar relationship prototypes are available for links between two biotic and two abiotic components. In the model shown in Figure 1, the prototypes chosen are 'consumes' for the relationship between Fish and Oxygen, and 'produces' for the relationship between Sunlight and Oxygen. The direction of the arrow between the variables of two components indicates the direction of causal influence. For example, the arrow from Fish to Oxygen in Figure 1 indicates that the Population of Fish influences the Amount of Oxygen.

A Mechanism in CMP modeling is a causal chain of component variables connected by causal relations. For example, Figure 1 illustrates a mechanism hypothesized by a team of students according to which the Amount of Sunlight (an abiotic component) influences the Amount of Oxygen (another abiotic component) and the Population of Fish (a biotic component) also influences the Amount of Oxygen.A Phenomenon in CMP is an observation about the system of interest. For example, the phenomenon for the mechanism illustrated in Figure 1 is a change in the Amount of Oxygen in an aquatic ecosystem. A user starts the process of CMP causal modeling using MILA–S with the goal of constructing a causal explanation of a given phenomenon. She then specifies a mechanism as the explanation for the phenomenon, incrementally composing the mechanism from the components of the system, their variables, and the relations between the variables. As Figure 1 illustrates, a CMP model in MILA–S is an external visual representation with textual annotations.

Agent-Based Simulations

MILA–S generates simulations using the popular NetLogo agent-based simulation platform [18]. NetLogo is well suited for simulating ecosystems because it models all behaviors of the ecosystem in terms of interaction among locally interacting agents in the system. Here we briefly summarize the NetLogo simulation platform and refer the reader to Wilensky & Resnick [17, 18] for details.

NetLogo can model both biotic and abiotic components. Biotic agents are organisms in an ecosystem that have intentional behaviors, such as eating, reproducing, and dying. Abiotic components have no intentionality and instead move around inside the simulation waiting for a biotic agent operate on them. Figure 2 illustrates the NetLogo simulation generated from the model illustrated in Figure 1. Note that all three components of the causal model are represented in the simulation: the Fish are in red, Sunlight hits the water at the brown dots, and the Oxygen produced by that interaction appears as blue dots.

Each biotic agent in a simulation has a set of methods or functions that describe its behaviors under different circumstances. Each agent also has variables or properties that can change as a result of these methods; in this

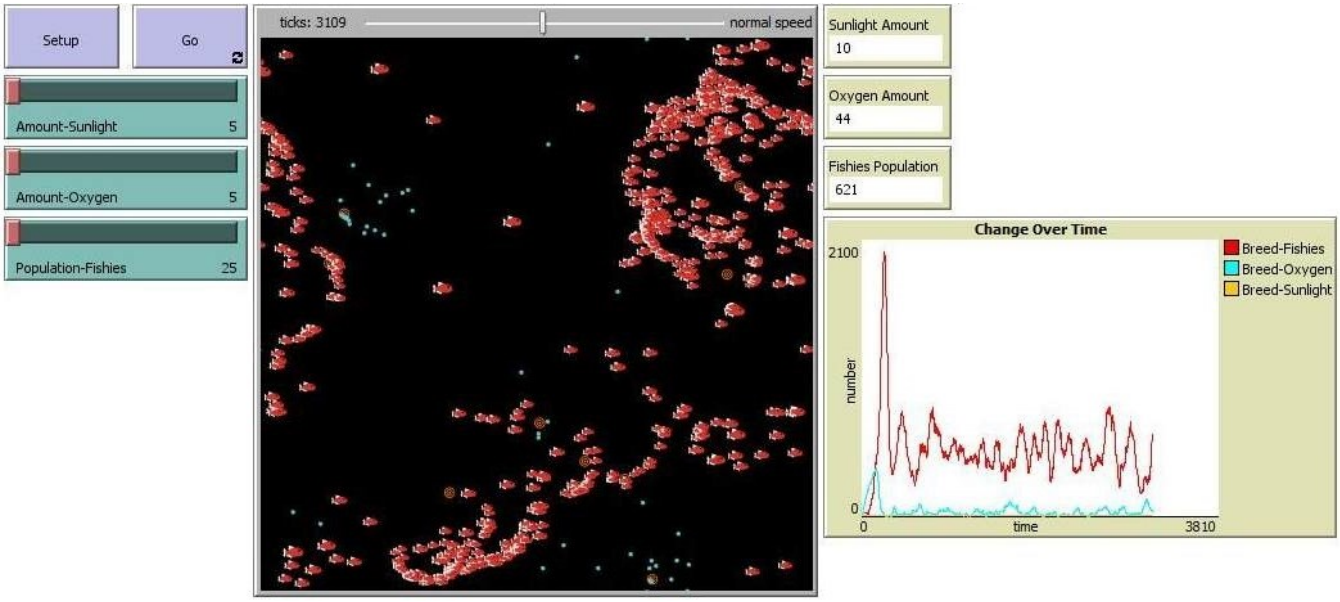

Figure 2: The simulation intelligently generated by MILA-S from the model in Figure 1.

example, an individual fish's energy level may rise when it executes its 'eat' method and fall when it executes its 'move' method. A biotic agent may also have methods dictating when it dies or when it reproduces; for example, the fish may have a method that dictates they die when their energy level drops below a certain threshold, and that they reproduce when their age and energy level are both above certain thresholds.

Each agent in a NetLogo simulation also has a special 'run' method that dictates the agent's behavior at every time step of the simulation. Every time the simulation clock ticks, the agent executes this method to determine its next behavior. Running the simulation amounts to executing this 'run' method for every agent present in the simulation. The engine iterates through every agent providing it with an opportunity to act. Upon completing the time step, the simulation advances to the next step, giving every agent the opportunity to act again. This proceeds until the user stops the simulation or the simulation reaches some built-in end condition, such as all agents dying.

As Figure 2 illustrates, NetLogo depicts the agents in a window showing their actions and behaviors. Also as Figure 2 illustrates, NetLogo provides graphs and counters for illustrating the temporal evolution of various variables of the simulation. Before running a simulation, the user sets the simulation's start condition. The input variables are set through the sliders and toggles on the left side of the simulation window illustrated in Figure 2. The user then clicks the Setup button to apply those changes to a new simulation. The user next clicks the Go button to start the time steps of the simulation.

Model-to–Simulation Bridge

As we mentioned in the introduction, one of the benefits of simulations in scientific inquiry is that they allow the user to test the predictions of their models. If a user finds that the model does not match observations of the real world, she may revise the model to more closely approximate the real world. However, this process operates most effectively when the cost of executing and revising simulations is not very high.

NetLogo simulations are typically designed with the its own dedicated programming language, which allows for enormous flexibility, However, this flexibility of designing simulations makes rapid evaluation and revision of models difficult. First, it requires at least a rudimentary background in programming. Secondly, even if the simulation designer is relatively experienced in NetLogo, it can still take significant time to make non-trivial changes to the way in which the simulation operates: these changes can involve writing all-new methods, creating new variables, or defining new agents. Clearly, it would be useful if the cost of generating NetLogo simulations could be controlled.

MILA–S provides one technique for controlling the cost of generating NetLogo simulations: it automatically generates the simulations from the user's casual model. Note also that the generation of the CMP causal model illustrated in Figure 1 does not require any knowledge of programming. Instead, MILA–S provides a visual syntax for CMP modeling.

Managing Initial Settings

All NetLogo simulations, both those generated automatically in MILA–S and those written manually, have certain boilerplate settings and elements that must be taken care of initially. MILA–S starts the simulation generation process by generating these boilerplate settings. This includes the majority of the visible elements of the NetLogo simulation illustrated in Figure 2. The Setup and Go buttons are initialized and added to the canvas, the main simulation visualization area is created and sized in the center, and the graph area on the right is created the contain the ongoing developments on the different graphed variables. These different elements are later filled in with specific content from for simulation that is being generated; the Go button, for example, is linked to the time step functions for each agent in the simulation, the visualization is provided the locations of the different agents, and the variables to be tallied are added to the graph at the right.

Managing Agents

After initializing the boilerplate for the simulation, the simulation generation engine moves on to first gathering together the various components that are present in the model and writing the necessary methods to instantiate them as agents in the simulation. First, the generation engine compiles a list of all the different components present in the simulation, both biotic and abiotic. This also involves checking for components duplicated across multiple nodes. Then, the engine iterates over all present components and calls on each to automatically write the code necessary to define themselves as agents into the simulation file.

A noteworthy element of the automatic generation of code for agents in the NetLogo simulation is that in many ways, generating the code itself is agent-based. Within the causal model that users construct, the components themselves have certain variables in their class structures. As Figure 3 illustrates, some of these parameters are toggleable: the user can toggle whether or not certain variables are monitored by a counter on the right, graphed by a line in the chart, or controlled by a slider on the left. Other controls presented here manipulate the visualization of the specific component or the range of potential values for user input when executing the simulations. In addition to the settings presented here, there are certain predefined features within each component as well, such as the variables associated with biotic or abiotic components. These are arguably the most important among these settings as proper values for the initial state of the simulation play a significant role in determining the usefulness of the outcome. Each individual

component in the model holds these parameters internally. When the generation engine is ready to write the code for these components, it calls these components one by one, passing them the file to which to write the simulation, and each one writes the code necessary to define itself as an agent. In this way, one can visualize this process as the agents writing their own code for the simulation!

When creating these agents, the generation engine applies certain assumptions to the structures and behaviors of these agents. These assumptions pertain to behaviors and other requirements that typically must be manually coded into the agents in NetLogo simulations. Thus, a major part of the work performed by the generation engine is in intelligently determining what can be assumed about these agents. While the user has control over the initial values of all of the variables present in the simulation, the assumptions herein presuppose initial conditions that would be plausible in an actual ecological system.

Abiotic agents come with a relatively simple set of assumptions. As passive, non-volitional agents, the movement of abiotic agents is assumed to be random. Upon creation, each abiotic agent is given a direction, and this direction changes pseudo-randomly throughout the duration of the agent's presence in the simulation. Abiotic agents do not reproduce to create new instances of themselves, and they do not disappear from the simulation unless removed by the activity of some agent. Abiotic agents can produce other biotic or abiotic agents as dictated by the model, but they do not have any automatic behaviors toward self-replication or destruction.

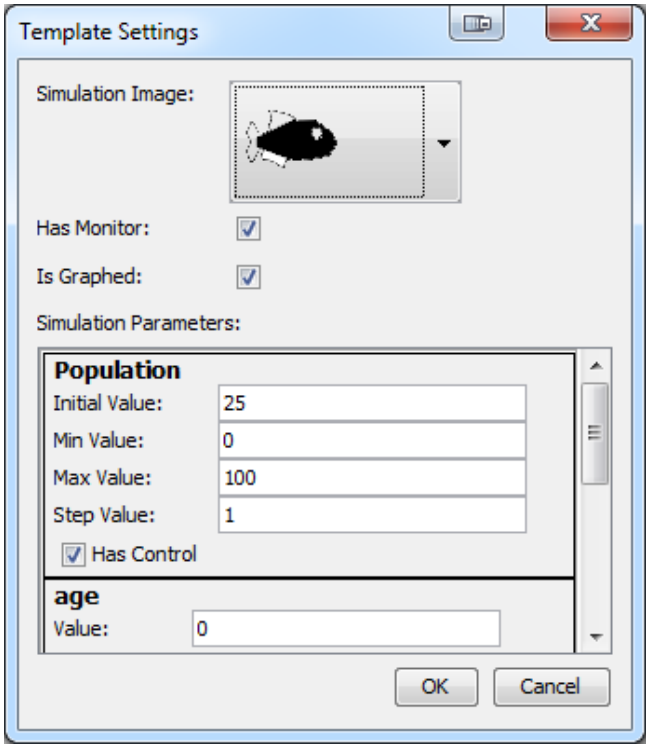

Figure 3: In this box, users annotate their model components and relationships with information for the simulation.

Biotic agents, on the other hand, come with more assumptions. First, MILA–S's simulation generation engine assumes that biotic agents have two inherent behaviors of life: reproducing and dying. All biotic agents are assumed to have a reproduction process. The user can modify the variables associated with this process, specifically the odds of reproducing (reducing it to 0 if no reproduction is desired), but the initial assumption is that any living agent in the simulation must be able to reproduce. Second, the generation engine assumes that all biotic agents have conditions under which they die. Initially, this is assumed to be a certain age threshold after which the likelihood of death increases as the agent's age increases. Agents can also die from lack of energy (the effect of a lack of food, oxygen, or carbon dioxide), although this only takes place in the presence of an external energy source, which is optional.

Managing Interactions Among Agents

With the simulation boilerplate and agent definitions in place, MILA–S' simulation generation engine proceeds to the most complex part of the simulation process: generating the interactions between agents in the simulation. Like the code for the individual agents, the code for interactions is similarly contained within the components. As we described earlier, each connection between two nodes in the CMP causal model has an origin and a destination; the component associated with the origin contains a reference to the connection itself. After iterating over all the components and writing them as agents in the simulation, the engine reiterates over all the components. In this new iteration, the generation engine iterates over all the connections of each component and writes them as interactions in the simulation. The simulation agent for each component in the causal model is called to write its interactions into the simulation code.

First, the engine defines the conditions under which the interaction will occur based on the interaction's prototype For example, a 'consumption' prototype occurs whenever a biotic agent intersects with the abiotic component it consumes while its energy level is below a certain threshold. Similarly, a 'production' prototype occurs at a regular rate as defined within the connection's properties screen (similar to that shown in Figure 3). A 'becomes upon death' prototype occurs whenever an agent's 'death' method is invoked. Thus, the engine determines when each specific interaction is triggered.

Second, the engine calls on the agents to define the actual bodies of their interactions. Each interaction prototype has an abstract definition; this abstract definition contains slots for the variables and operands of the specific components on which the interaction operates. In writing their

relationships, the agents input their specific variables into the appropriate slots in the interaction prototype, then output the completed interaction definition into the simulation code itself. Then, for each such interaction definition, the agent writes the additional code necessary to realize the full spectrum of the interaction. For example, in the case of the 'consumes' interaction, the agent being consumed must call its 'death' method (if biotic) upon consumption. In this way, while the interaction is initiated by the component on the origin side of the connection, the component on the destination side must also behave in certain ways to realize the interaction in the simulation.

Like the definitions of the agents in the simulation, MILA–S's simulation generation engine also makes certain assumptions about the interactions among the agents. The most significant assumption it makes is that any time an agent is defined as consuming another agent, it is assumed that this consumption is necessary to the agent's survival. For example, the model in Figure 1 shows that fish consume oxygen. Based solely on this relationship, the simulation generation engine infers that fish rely on oxygen to survive. Oxygen, then, is connected in the simulation to the fish's energy level. This energy level decreases automatically over time and the fish relies on oxygen to replenish it. If, however, the biotic agent does not have a consumption relationship with anything else in the model, the engine infers that the agent does not rely on any external consumption to survive, and as a result the biotic agent does not die from a drop in energy.

Several similar assumptions are made based on the different types of relationships present in the models. The simulation generation engine similarly infers that if a biotic agent consumes multiple other components, the agent requires all of them to survive. When a biotic agent consumes another agent, it is inferred that that consumption provides energy to the biotic agent, and that the consumed agent ought to be removed from the simulation. When an abiotic agent, such as ammonia, is said to destroy the population of a biotic agent, such as a fish, it is inferred that contact with that agent kills the biotic agent immediately; if, on the other hand, the ammonia is drawn as destroying the energy of the fish, then contact with ammonia lowers the fish's energy level, potentially eventually leading to death. All of these assumptions and inferences are principles or relationships that would typically have to be coded into the simulation manually, and broadly represent much of the work and overhead required in such a process. By intelligently extrapolating from the model necessary details like these, MILA–S helps make it significantly easier to use and modify NetLogo simulations and thus enables the rapid construction and revision of CMP causal models.

Managing Visualization & Interactions

After creating all the agents and interactions present in the simulation, MILA–S's simulation generation engine iterates over the agents in the simulation and their interactions, and creates the visualizations and controls necessary to observe and modify the parameters of the visualization at runtime. For each component variable that is designated to have a control (as shown in Figure 3), a slider is added on the left. Each variable that is monitored is given a monitor on the right and a line on the graph. Finally, the individual agents are added to the visualization for appropriate tracking and movement. After this, the simulation is ready to be run by the user whenever the user wants.

TESTING & ANALYSIS

MILA–S was pilot tested with about 50 7th grade life science students in a middle school in the near Atlanta, USA. The pilot study of MILA–S took place after more rigorous testing of other pieces of the MILA learning environment. In the larger experiment, about 250 students spent two weeks using MILA without the simulation generation component. During this time, they investigated the sudden death of thousands of fish in a lake in downtown Atlanta. Students were asked to use MILA to investigate this system and explain what had caused so many fish to suddenly die with no external signs of disease or trauma. For these two weeks, students posed multiple hypotheses, consulted several sources, participated in laboratory procedures regarding acidity and chemical contamination, and researched similar events in other ecological systems. During this time, students also used three static NetLogo simulations. These three simulations were not generated based on students' causal models, but rather were written by domain experts prior to the use of the system. Specifically, these simulations covered fish breeding behaviors, algae behaviors in different seasons, and the interaction between fish, algae, oxygen, and carbon dioxide.

The middle school teachers teaching the unit then selected about 50 students for the pilot study with MILA–S. These students were asked to construct a causal model that, when simulated, would give the same results as those that were actually observed in the Atlanta lake in which thousands of fish had died. In this way, students began engagement with MILA–S with two weeks of experience with conceptual modeling in MILA, manipulating NetLogo simulations, and the ecosystem which they aimed to explain. For this pilot experiment, the students worked in small teams of two to three. While students were only given 50 minutes to use MILA–S during this study, their prior experience with the learning environment and content knowledge led to quick engagement with the assignment, and all teams were able to construct and simulate non-trivial models by the end of the session. In fact, most teams were able to engage in at least one cycle of model construction, simulation use, model revision, and simulation reuse.

During interaction with MILA–S, screenshots were taken of students' workspaces, microphones were used to record students' conversations, and researchers watched in person for student behaviors. This data, along with the final projects that students constructed, log data from the

software on the construction process, and feedback from students entered into the software's notepad, was used to conduct the following analysis.

Testing Analysis

The goal of this pilot study was to evaluate the usability and usefulness of MILA–S. Thus, evaluation and analysis has taken a qualitative design-based research approach, aimed at discerning the natural behaviors in which students engage while using MILA–S as well as the challenges or opportunities for future development of the process. Thus, analysis attempts to answer three questions: (1) How did students interact with MILA–S; (2) What difficulties did they encounter with MILA–S; and (3) What opportunities are there for future development of MILA–S?

Student Behavior

Students were given relatively limited advanced instruction on the process of model construction and simulation. As referenced previously, the goal provided to students was to create a simulation (by first creating a causal model) that led to the same results as the actual phenomenon. Students were told that 'same results' can mean two things: at a basic level, it means that all the fish in the simulation die as they did in the lake; at an advanced level, it means that all the fish die *and* other observed phenomena, such as the rising level of carbon dioxide, the falling level of oxygen, and the rising population of algae, also occur, with minimal coincidence of absent phenomena, such as a sudden change in weather patterns.

Given only this guidance, the majority of student teams engaged in a (surprisingly) sophisticated process of modeling and simulation. Teams began by modeling simple relationships they already believed to be true from their previous research on the system in the earlier unit; a few teams started by opening their previous projects as well to keep available for consultation and information. All teams generated their first simulation early in the engagement; no teams attempted to model the entire system accurately before attempting to run a simulation.

The projects of one particular team are used earlier in this paper in Figure 1 and Figure 2 to illustrate the workings of MILA–S. Figure 1 shows the initial model that this team constructed before every running a simulation; Figure 2 shows the first simulation this team ran after constructing that initial model. This team then ran their simulation a few more times to discern repeated patterns; they observed specifically that fish populations rise and fall with the rise and fall of oxygen concentration, and that oxygen concentrations fall and rise with the rise and fall of fish populations, creating a feedback cycle. However, they also observed that this cycle never caused the fish to die out altogether, as happened in the lake. They speculated that this relationship alone could never cause the fish to die off altogether because the oxygen would always rebound when the fish population dropped. They also noted that their simulation did not capture other substances believed to be significant, especially algae and carbon dioxide. Equipped with this feedback, they set about revising their model.

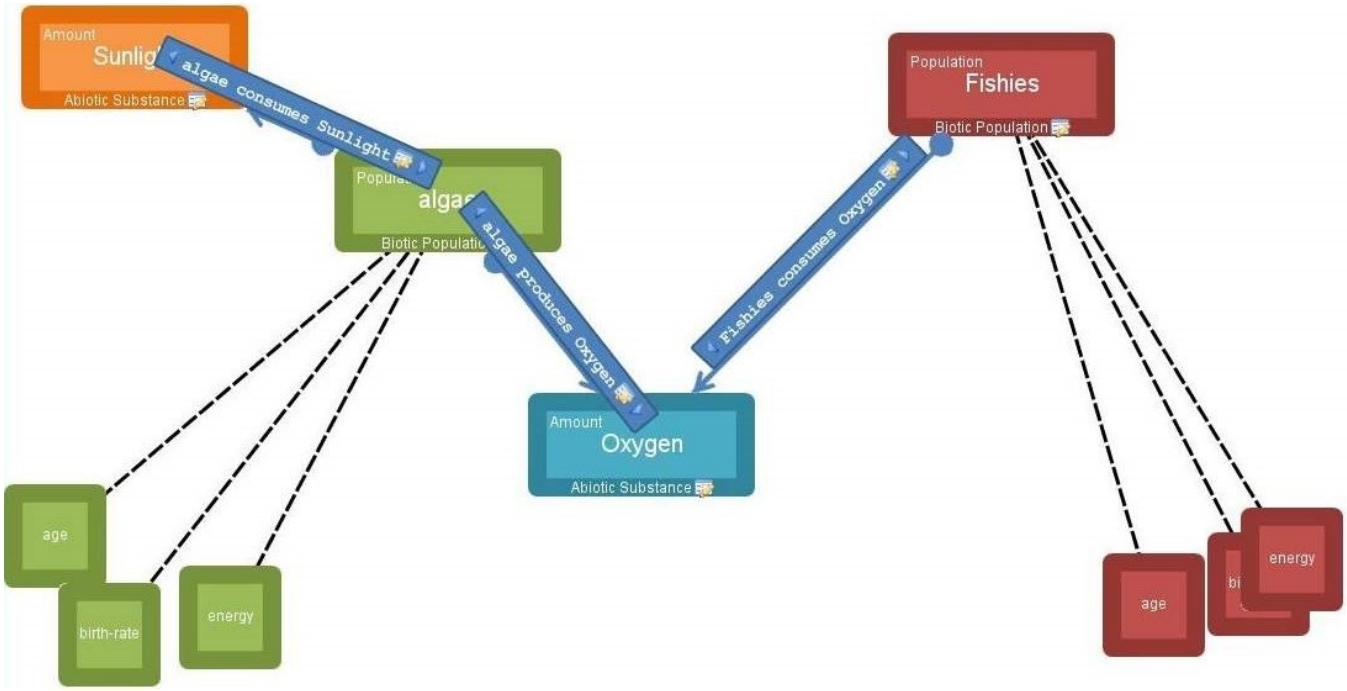

Figure 4: The revision of the model shown in Figure 1 constructed by the group after receiving feedback from the simulation shown in Figure 2.

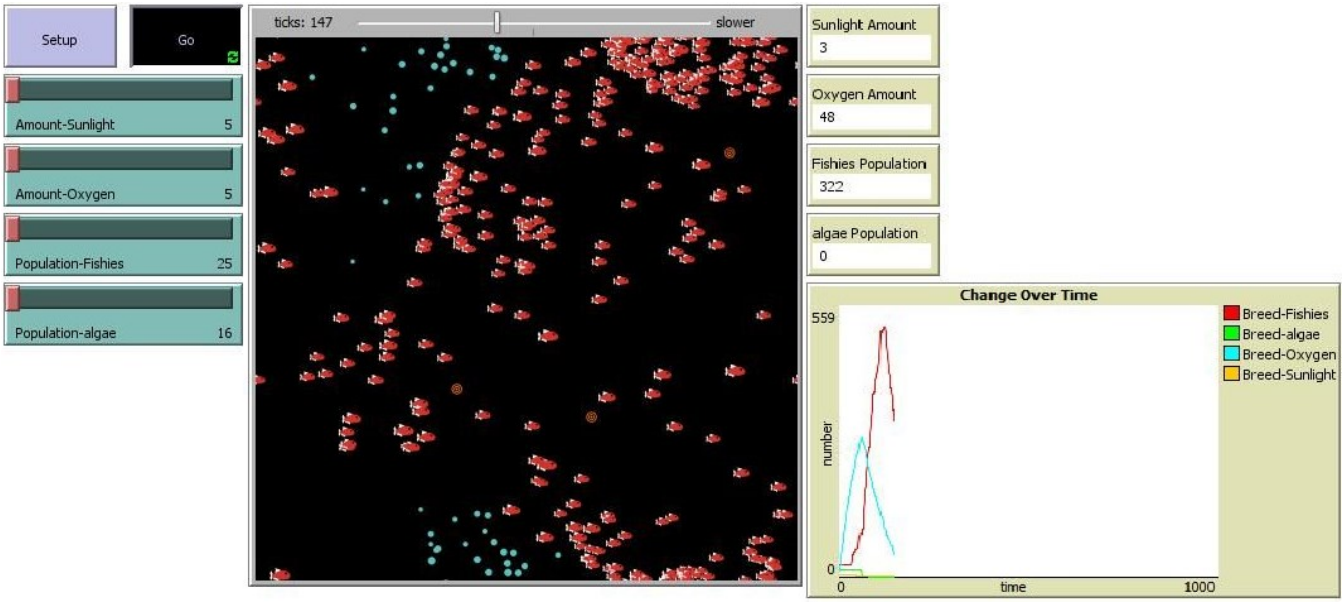

Figure 5: The simulation generated from the model shown in Figure 4.

The initial revision of the team's model is shown in Figure 4. Based on their observations, students revised their description of the way in which oxygen is generated in the system: instead of viewing it simply as being created by sunlight, they included the biotic component algae as the consumer of sunlight and producer of oxygen. Because algae is a biotic component, the simulation generated from this model inferred certain biological principles about algae, most notably that in the presence of excess sunlight, it would grow in population. Subsequently, in the presence of excess oxygen, fish, too, would grow in population given the absence of an additional limiting energy source (such as food). Then, as noted in the simulation shown in Figure 5, the spike in fish population causes the oxygen concentration to drop. Algae, serving as the intermediary in generating oxygen, does not create oxygen fast enough to prevent the fish population from crashing to extinction. The screenshot provided in Figure 5 depicts the start of this crash.

Throughout this process, students demonstrated a mature understanding and execution of the feedback cycle between conceptual modeling and simulation. Students naturally, with no specific guidance in this direction from teachers or researchers (except what was provided during the initial study preceding this study), began using observed results from the simulations to inform their continued construction of their models. In this way, students demonstrated the four cycles of model construction suggested by the literature [13]: they constructed a model initially, used the model through the simulation, evaluated the model through the results of the simulation, and revised the model for another iteration of the cycle. The process seen in this team was seen in the majority of teams as well based on preliminary analysis of model construction logs and screenshots of interaction behaviors.

Broader Trends

To better understand the process of model construction across the various student teams, a case study analysis was conducted on the course of model construction and revision. Three significant trends were observed. First, for a majority of teams, the initial phase of model construction consisted primarily of testing out the connections between conceptual model and simulations than explicitly trying to accurately model the system. Multiple teams constructed models and made changes early on not necessarily in an attempt to more accurately describe the system, but rather to understand the way in which certain connections in the models would manifest in the simulations. For most (but not all) teams, this process preceded actual attempts to accurately model the system, and many teams completely started over once they had attained a decent working knowledge of the connection between the models and the simulations.

A second trend marked a qualitative difference in the modeling process present in the earlier engagement with MILA and that present during engagement with MILA–S. Models constructed previously in MILA could be considered "retrospective" in that they attempted to specifically capture a particular series of events that had actually transpired in the lake. Models constructed in MILA–S, on the other hand, were largely "prospective" in that they aimed to provide not only an account of what had actually happened, but also an account of what would have happened under different conditions. Although a subtle

difference, this shift represented a significant fundamental change in the exercise's purpose, from modeling a series of events to modeling a collection of relationships.

Finally, a third trend reflected the unique way in which conceptual models and simulations, when used together, allow students to execute the documented process of model construction [1, 5]. The literature largely breaks the process into phases of model construction, use, evaluation, and revision. Examining students' model construction activities under this framework, students can be seen using the conceptual models and generated simulations as two versions of the same underlying model. Model construction is performed in the conceptual model, while model use and evaluation are conducted using the generated NetLogo simulation. Students use these simulations to see what their model predicts would have happened under the circumstances observed in the system, then evaluate how well those predictions match the actual observations. Based on this evaluation, students then revise the original model to more accurately capture the system and repeat the cycle.

Challenges

The trajectory of model construction, use, evaluation, and revision shown in Figure 1, Figure 2, Figure 4, and Figure 5 demonstrates only approximately half of this team's interaction. Similarly, many other teams reached this stage with plenty of time to continue forward. However, a challenge arose in nearly all teams at approximately this point. MILA–S provided an effective framework at simulating the interactions between a small number of components and their variables; however, the systems that students were examining involved several more components than these, along with multiple relationships between their variables. Upon reaching a level of complexity slightly above what is shown in Figure 4, the NetLogo simulations generated by MILA–S stopped providing meaningful feedback to students. The number of agents would explode based on the multiple consumption and production relationships at play, slowing the simulation down and rendering the visualization elements indistinguishable. Repeated runs of the same simulation with the same initial parameters generated wildly varied responses as the number of agents and methods exacerbated the influence of random chance on the simulation's outcomes. These challenges are documented in similar attempts to create bridges between conceptual models and formal simulations [10, 11].

It is still likely that with the proper parameters and relationships, MILA–S could still have generated usable simulations that gave meaningful feedback; the challenge that arose is that most executions of the simulations gave limited or no feedback as to the changes that needed to be made to more closely replicate the real phenomenon. Too much noise existed in these simulations to continue to facilitate the evaluation and revision process. Future work in this direction, then, needs to improve the intelligent generation of simulations based on these conceptual models. Additional reasoning can be used to scope and limit the simulation space to remove the majority of this noise and emphasize the useful patterns. Specifically, this additional intelligence can emphasize a natural notion of carrying capacity to prevent the populations in the simulations from exploding. It can also generate more abstract and scalable relationships in the simulation by moving away from a one-to-one matching between conceptual relationships and simulation methods.

Opportunities

In addition to these positive takeaways, this work also presents two key opportunities for future development. First, the during model construction, use, evaluation, and revision process is similar to other processes encountered in other disciplines. Computer programming, for example, involves constructing, executing, observing, and revising a program. Similarly, industrial design uses a prototyping approach in which the designer constructs a prototype, tests it with users, evaluates their results, and revises it accordingly. Thus, there is a significant opportunity to use this approach to teaching scientific inquiry to teach about underlying general design process. Indeed, in a sister project, we have developed an interactive development environment for designing game-playing agents based on the same process of rapid modeling, simulation, evaluation, and revision [12].

Secondly, the opportunity presented by bridging the gap between conceptual models and simulations is not restricted solely to science education; it is an opportunity for real-world scientific pursuits as well. This curriculum was sculpted to teach this process precisely because it is so authentically analogous to scientific research [7], and the need for a bridge between conceptual models and computational simulations exists in the real domain as well. By enhancing this framework with added flexibility to define new interaction prototypes, an ontology of a broader assortment of variables, and a more intelligently restricted simulation space, MILA–S and similar tools can provide valuable information for real research in various domains.

CONCLUSIONS

This paper has presented the design of an interactive system called MILA–S for generating agent-based simulations from conceptual models of ecological systems. MILA–S not only enables construction of causal models of components and mechanisms in an ecosystem, but it also takes as input the causal model and autonomously generates an agent-based simulation that shows the temporal evolution of the system according to the causal model. The user needs to simply use a visual syntax for generating causal models and the interactive tool automatically generates the corresponding simulation. Further, because the simulation directly corresponds to the causal model, the

results of the simulation directly evaluate the model and point to the revisions needed in the model.

Preliminary results from a pilot study in middle school science indicate that MILA–S is both useful and usable as an interactive learning environment. MILA–S was usable in that the students in our study had little difficulty in using the interactive tool for model construction, use in simulation, evaluation of the simulation results in comparison to the real world, and revising the model accordingly. Further, MILA–S was useful in that the above process of modeling led the students to significantly better causal models. On one hand, this integration of conceptual modeling and executable simulations brings authentic scientific practice to middle school classrooms. On the other, this intelligent generation of agent-based simulations from causal models can inform the construction of similar systems for use in real scientific practice.

ACKNOWLEDGMENTS

We are grateful to Vickie Bates and Lara Catall, the teachers in this study, as well as to Kristina Strickland, Angela Gula, and Connie McCrary for the support provided. We also thank our colleagues at the Design & Intelligence Laboratory for many discussions about this work, especially Rochelle Lobo, David Majerich, Spencer Rugaber, and Swaroop Vattam.

REFERENCES

1. Clement, J. (2008). *Creative Model Construction in Scientists and Students: The Role of Imagery, Analogy, and Mental Simulation*. Dordrecht: Springer.
2. Darden, L. (1998). Anomaly-driven theory redesign: computational philosophy of science experiments. In T. W. Bynum, & J. Moor (Eds.), *The digital phoenix: how computers are changing philosophy*, (pp. 62–78). Oxford: Blackwell.
3. de Jong, T., & van Joolingen, W. R. (1998). Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research, 68*(2), 179-201.
4. Goel, A., Rugaber, S., & Vattam, S. (2009). Structure, Behavior & Function of Complex Systems: The SBF Modeling Language. *AI for Engineering Design, Analysis and Manufacturing,* 23: 23-35.
5. Halloun, I. (2007). Mediated Modeling in Science Education. *Science & Education, 16*(7), 653-697.
6. Jackson, S., Krajcik, J., & Soloway, E. (2000) Model-It: A Design Retrospective. In M. Jacobson & R. Kozma (editors), *Innovations in Science and Mathematics Education: Advanced Designs for Technologies of Learning* (pp. 77-115). Lawrence Erlbaum.
7. Joyner, D., Majerich, D., & Goel, A. (2013). Facilitating authentic reasoning about complex systems in middle school science education. In *Proc. 11th Annual Conference on Systems Engineering Research, Procedia Computer Science 16:* 1043-1052. Elsevier.
8. Machamer, P., Darden, L., Craver, C. (2000) Thinking about Mechanisms. *Philosophy of Science, 67*(1): 1-25.
9. Nersessian, N. (2008). *Creating Scientific Concepts*. Cambridge, MA: MIT Press.
10. Norling, E. (2007). Contrasting a system dynamics model and an agent-based model of food web evolution. In *Procs. Multi-Agent-Based Simulation VII,* LNAI 4442: 57-68, Springer.
11. Parunak, H., Savit, R., & Riolo, R. (1998). Agent-based modeling vs. equation-based modeling: A case study and users' guide. In *Procs. Multi-Agent Systems and Agent-Based Simulation*, LNAI 1534: 10-25, Springer..
12. Rugaber, S., Goel, A., & Martie, L. (2013) GAIA: A CAD Environment for Model-Based Adaptation of Game-Playing Software Agents. In *Procedia Computer Science 16*:29-38.
13. Schwarz, C., Reiser, B., Davis, E., 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*, *46*(6), 632-654.
14. Vattam, S. S., Goel, A. K., & Rugaber, S. (2011). Behavior Patterns: Bridging Conceptual Models and Agent-Based Simulations in Interactive Learning Environments. In *Proc. 11th IEEE International Conference on Advanced Learning Technologies (ICALT-2011), July 2011.* 139-141. IEEE.
15. Vattam, S., Goel, A., Rugaber, S., Hmelo–Silver, C., Jordan, R., Gray, S, & Sinha, S. (2011) Understanding Complex Natural Systems by Articulating Structure-Behavior-Function Models. *Journal of Educational Technology & Society,* 14(1): 66-81.
16. White, B, & Frederiksen, J. (1990) Causal Model Progressions as a Foundation of Intelligence Learning Environments. *Artificial Intelligence*, 42(1): 99-157.
17. 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.
18. Wilensky, U., & Resnick, M. (1999). Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and Technology*, 8, 3-19.
19. Winsberg, E. (2001). Simulations, models, and theories: Complex physical systems and their representations. *Philosophy of Science*, S442–S454.