LevelSpaceGUI - Scaffolding Novice Modelers' Inter-Model Explorations

Arthur Hjorth Bryan Head Corey Brady Uri Wilensky

{arthur.hjorth, bryan.head}@u.northwestern.edu {cbrady, uri}@.northwestern.edu

+1 773 943 0013 +1 206 504 0137 +1 847 371 1985 + 1 847 372 2524

Center For Connected Learning and Computer-Based Modeling
Departments of Learning Sciences and Computer Science
Northwestern Institute on Complex Systems
Northwestern University
2120 Campus Drive
60208 Evanston, IL, USA

ABSTRACT
We present an interface for programming relationships between two or more NetLogo [18] models running concurrently. The interface is designed specifically to help high school aged novices explore and define computational relationships between agent-based models, and to investigate how prompting learners to reason about the relationships *between* complex systems may change how they reason about the systems individually.

Categories and Subject Descriptors
H.5.m [**Information interfaces and presentations (e.g., HCI)**]: Miscellaneous.

General Terms
Design, Languages, Modeling, Systems

Keywords
Agent-based modeling, education, novice programming, complex systems, complexity, multi-level, science education

1. INTRODUCTION
The last two decades have seen an increased focus on methods for both studying complexity [8, 26, 27] and researching how learners make sense of complex systems [7, 22]. One strand in complexity research has focused on agent-based descriptions of systems. Within this perspective, many core scientific phenomena in a variety of domains can be understood through a complexity lens using computational simulations of the interactions of many individual "agents" [1, 12, 21]. Modelers can give instructions to thousands of independent agents, all operating concurrently. This makes it possible to capture complex system behavior by "growing it" [2, 17] from the behavior of these system elements and to explore the connection between the micro-level behavior of individuals and the macro-level patterns that emerge.

Research on learning about emergence from an agent-based perspective begins from the observation that reasoning about complexity involves coordination between (at least) two "levels" of experience. This approach posits that difficulties arise when learners mis-apply intuitions developed and found effective at one level of experience, to another level [11, 14, 24]. Agent-based modeling languages like NetLogo [18] have been designed in part to address this challenge, supporting the development of learners' intuitions about complex systems. By allowing learners to bridge their understanding of how individual entities behave with their observations of how the systems act in the aggregate, learners are able to overcome many of these problems [10, 11, 14, 19, 23]. Moreover, we have found that applying an agent-based modeling perspective can provide immediate benefits for scientific understanding as well as supporting learners in applying a complexity lens to phenomena in other domains [3].

This line of inquiry has typically focused on learners reasoning about the interactions *within* individual systems. By describing just one phenomenon, the agent-based model has acted as a focusing device, excluding or simplifying factors not directly relevant to the dynamics of the phenomenon. A potential downside to this approach, of course, is that phenomena could be seen as isolated or disjointed, when they may in fact be highly interconnected and interdependent at a "higher" meta-systemic level.

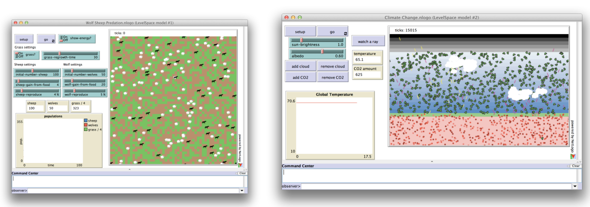

Figure 1: LevelSpace connects NetLogo models of phenomena like ecosystems and climate change.

We hypothesize that providing support for learners to reason *between* systems may not only be valuable in itself but may also be a fruitful approach to getting them to reason in greater depth about each system. This hypothesis was substantiated by a pilot study that we conducted with an early prototype of our design in which we found that learners asked new questions of individual systems when asked to connect them to another system [4]. We wish to extend this line of inquiry to include reasoning about the

IDC '15, June 21-25, 2015, Medford, MA, USA
ACM 978-1-4503-3590-4/15/06.
http://dx.doi.org/10.1145/2771839.2771884

interactions *between* systems by enabling learners to connect different agent-based models and program interactions between them at both the agent and aggregate levels.

Our demo shows our design of a graphical programming interface, LevelSpaceGUI, based on a novel and powerful NetLogo extension called LevelSpace [5] in which both researchers and learners can connect NetLogo models, program interactions between them, and explore the results as the models run together. LevelSpaceGUI is an application that provides a lower threshold graphical interface to LevelSpace using the eXtraWidgets extension [13] to dynamically add, remove, and modify interface elements (or "widgets") in NetLogo.

1.1 DESIGN CONTEXT

Over the next two years we will be iteratively developing curricular activities based on LevelSpaceGUI in two different contexts—after-school computer clubs and classrooms—both at the high school level. The design literature on reasoning about complex systems emphasizes the necessity of a deep understanding of the behavior of the agent-level entities in the system, described by agent-rules, which in turn implies the importance of actively model*ing*, rather than simply interacting with pre-programmed models. If learners do not know the inner workings of a model, including the entities, their properties, and their rules of interaction, it will be much harder for them to make these connections. Of course, this requires a level of familiarity with programming that many learners in our contexts may not possess. Other NetLogo-based programming environments for novices such as DeltaTick [25], NetTango [6], or Modelling4All [9] enable such scaffolding for novices, but none currently allow cross-model linking. We built LevelSpaceGUI specifically to enable modeling across linked systems by novices.

2. DESCRIPTION OF OUR DESIGN

LevelSpaceGUI is itself implemented as a NetLogo model. It allows users to load two or more other, separate NetLogo models and program connections between them. Our hope is to scaffold learners during this process in two different ways: first, by always offering learners access to potentially relevant data structures from the models that they are connecting. While working on connecting models, LevelSpaceGUI makes the programmatic components of the model (procedure/method names, global variable names, breed names, breed-specific variable names, etc.) easily accessible to learners. (Figure 2). Additionally, learners can construct new components with which inter-model relationships can be defined.

Second, we scaffold learners during inter-model programming by constraining the combinations of what can interact with what and by restricting in a few different ways: First, we introduce static types to help prevent runtime errors. NetLogo is a dynamically typed language, which in spite of all of its benefits means that syntactically there are almost no constraints on what can interact with what, which in turn can lead to runtime errors. We therefore introduced three different, static types: *extended agents*, *commands* and *reporters*.

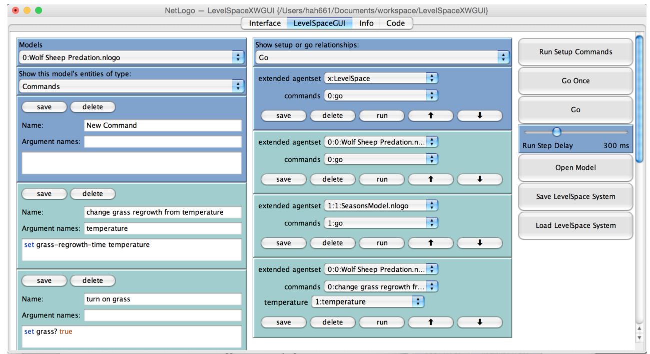

Figure 2: Screenshot of LevelSpaceGUI. Commands, reporters, and extended agents are on the left. Runtime interactions between models are in the middle. Control buttons are on the right. Blue boxes are part of the 'workspace' and signify that the command or the inter-model relationship has not yet been saved.

Agents in our design have been extended to include anything that can act; both NetLogo's native types of agents (turtles, links, and patches), and the *models themselves* – that is, we have represented models themselves as first-class agents. Commands are anything that run code and change the state of any of the loaded worlds, without returning a result value. Reporters are anything that returns some sort of data, whether at the level of a model or at the level of an individual agent inside a model. By introducing static types, we minimize the risk that learners get runtime errors when they are running their LevelSpace models.

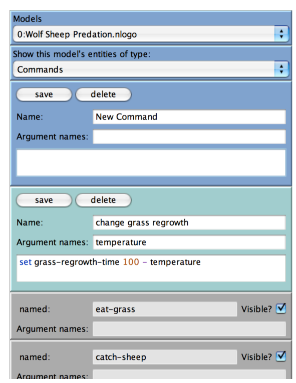

Figure 3: The interface allows learners to "inspect" the models that they open. Gray commands are inbuilt, green ones are user-created.

Second, we constrain the reporters that are available as arguments depending on which extended agent is running the command. For instance, each model will contain different *breeds* of agents, each of which has their own variables. Wolves and sheep in the Wolf-Sheep Predation [14] NetLogo model, for instance, have variables like energy and x- and y- coordinates, and these are available only when wolves or sheep are chosen as the extended agent to run a command (Figure 4). Our hope with this design decision is to help learners better understand what kind of information is available to each extended agent, and thus to enable them to build multi-level models from the agent-perspective.

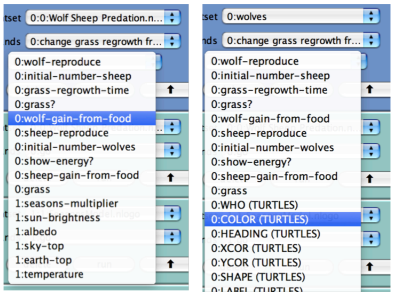

Figure 4: Side by side comparison of available arguments when respectively a model or lower-level agents run commands. Breed-specific arguments show up only on the right side.

Finally, our intention in visually separating the 'parts' of a model (left-hand column) from its runtime (center column) is to make this distinction conceptually clearer to learners. In addition, in the center column, we separate out inter-model commands that are run at setup from those that are run during runtime, for the same reason.

2.1 Example: Interactions between a model of Climate and a model of an Ecosystem

To illustrate the kinds of connections and entities that learners might create, we will give an example of how two models in the NetLogo library, Climate Change (CC) [15] and Wolf Sheep Predation (WSP) [20], can be connected. Briefly, CC shows how the interaction between photons from the Sun, infrared energy, clouds, and greenhouse gases such as CO_2 and methane interact to produce the greenhouse effect. WSP shows population dynamics in a two-tiered ecosystem in which wolves eat sheep, sheep eat grass, and grass grows back after a certain period of time.

So, how might these two models be connected? This question has guided our early implementations of LevelSpaceGUI with learners. In working closely with the WSP model, learners identified a wide array of external factors that could affect the Wolf-Sheep-Grass Ecosystem. The list below is aggregated from student ideas generated while exploring the WSP model:

- Seasons, weather, climate, or the sun's effect on the grass
- Rain/drought/floods
- Fire, tornados, natural disasters
- Diseases
- Human effects (hunters, farmers, shepherds, poachers)
- Other animals, in particular other predators for wolves and/or sheep
- Animals' group behavior (e.g., herding, flocking)
- Manmade (fences/walls) or natural (rivers/mountains) barriers to movement
- Lifespan of sheep and wolves (youth, old age)

The theme of weather and climate factors was salient in student thinking, which suggested exploring possible links between WSP and CC. Other interests led to links between WSP and other models, such as Fire [16]. In the rest of this section, we follow out one pathway for exploring connections between WSP and CC.

First, the rate at which grass in the WSP ecosystem grows could be a function of, among other possible weather-related factors, temperature. To create this relationship, a learner would first select the WSP model, then choose "Commands". This would show a list of all commands currently in the WSP model. She can then fill in the blue box— first naming her new command (e.g., 'change grass regrowth'); then writing the function that she believes would describe the relationship between the growth rate of grass and temperature; and finally deciding what arguments this function would take (Figure 5a). She would then create the relationship between CC and WSP by creating a new relationship in the center column, choosing the CC model first because it is the agent that causes this change to happen, then select her newly constructed command, and finally choose which of the parameters from CC would be passed on to their command as arguments–in this case, 'temperature' (see Figure 6, bottom block).

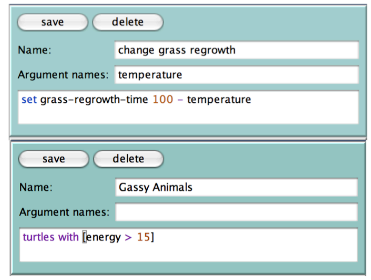

Figure 5: Two examples of the interface for creating LevelSpace-commands, reporters, and extended agents. 5a (top): creating a command for changing how fast grows back; 5b (bottom): creating an extended agent consisting of 'gassy animals.'

Second, the greenhouse gases in the atmosphere come from, amongst other sources, animal flatulence. A learner might hypothesize that animals with full stomachs are gassier than other animals. So she might decide that only the animals who are most full should participate in this interaction, by first choosing the WSP model, then "Extended Agents", name their collection of agents 'gassy animals', and then choose all turtles[1] satisfying some criterion, e.g. having an energy value greater than 15 (Figure 5b). She would then create a new relationship between the 'gassy animals' and the 'add-greenhouse-gas' command that already exists in the CC model, and that was 'imported' to the interface as part of the initial import of the elements of the model (Figure 2). The end result is the LevelSpace go-procedure seen in Figure 6.

This example illustrates a few important features of our design: Understanding the scope of variables is potentially difficult for novices to begin with. This difficulty can be exacerbated by having many different, concurrent models and their respective agents each with their own sets of global and agent-specific variables. LevelSpaceGUI helps learners by only populating the dropdown menu with commands that that particular extended agent is actually able to run, and only allows variables as arguments that these agents have "knowledge of" and access to.

For instance, the extended agent called 'gassy animals' contains a reference to 'energy'. This variable is only accessible to 'turtles' in the WSP model, and not to the turtles in the CC model. Similarly, when CC calls the learner-constructed command 'change grass regrowth', which is a command specific to the WSP model, it essentially asks the WSP model to run this command. The command's code (Figure 5a) contains a reference to 'grass-

[1] Generic mobile agents in NetLogo are called 'turtles'

regrowth-time', a global variable that only WSP has access to, but it is being changed as a result of a variable in another model. By constraining users' references to model- or agent-specific reporters and commands, our hope is to both prevent runtime errors, and to encourage novices to think about the scope of variables.

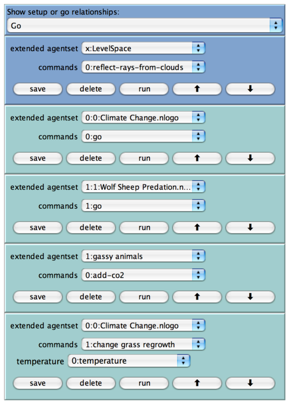

Figure 6: The go-procedure with relationships between WSP and CC.

In our early iterative design work with learners, we have already begun to see evidence in favor of our guiding hypothesis, namely that providing support for learners to reason *between* systems may not only be valuable in itself but may also be a fruitful approach to getting them to reason in greater depth about each system. In particular, not only have we found that students are able to conceptualize links between NetLogo models that they have studied individually, but also making these links can lead them to reflect more deeply on these systems. For instance, after reasoning about climatic effects on WSP, students became more attentive to the agent-based rules of interactions between wolves and sheep. An initial hypothesis—that barren ground or longer grass might make it easier or harder for wolves to hunt sheep—was disconfirmed by looking more deeply into the NetLogo code of WSP. In another example, students became unsatisfied with "grass regrowth time" as a simple means of expressing seasonal effects. Instead, they wished to explore the effects of snow-covered areas in winter, or water-covered areas in rainy seasons. Indeed, these reflections (involving how to remove land areas temporarily from the grass-regrowth cycle) connected the inquiry of students who were connecting WSP with CC with lines of thought being pursued by students connecting WSP with the Fire model.

Our empirical studies are in early stages, but these early implementations suggest that, conceptually, linking models and thinking between models are generative acts that support powerful ways of thinking about the systems involved.

3. DEMONSTRATION AT IDC2015

In our workshop, attendees will connect models from the NetLogo models library (or their own!). We also welcome pedagogical and design-related discussions about inter-model reasoning, and how to design and study curricular activities that foreground the particularities of thinking between models.

4. ACKNOWLEDGMENTS

This work is supported by The National Science Foundation grant #1441552. The opinions expressed here are solely those of the authors.

5. REFERENCES

[1] Epstein, J.M. 2006. *Generative social science: Studies in agent-based computational modeling*. Princeton University Press.

[2] Epstein, J.M. and Axtell, R.L. 1996. *Growing Artificial Societies: Social Science From the Bottom Up*. Brookings Institution Press MIT Press.

[3] Goldstone, R.L. and Wilensky, U. 2008. Promoting Transfer by Grounding Complex Systems Principles. *Journal of the Learning Sciences*. 17, 4, 465–516.

[4] Hjorth, A., Brady, C., Head, B. and Wilensky, U. 2015. Thinking Within and Between Levels: Exploring Reasoning with Multi-Level Linked Models. *Exploring the material conditions of learning: opportunities and challenges for CSCL* (Gothenburg, Sweden).

[5] Hjorth, A., Head, B. and Wilensky, U. 2015. *LevelSpace NetLogo Extension. Center for Connected Learning and Computer-Based Learning. Evanston, IL.*

[6] Horn, M. and Wilensky, U. 2011. NetTango: A Mash-‐Up of NetLogo and Tern.

[7] Jacobson, M.J. and Wilensky, U. 2006. Complex systems in education: Scientific and educational importance and implications for the learning sciences. *Journal of the Learning Sciences*. 15, 1, 11.

[8] John, H. 1998. *Holland, Emergence: from chaos to order*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

[9] Kahn, K. and Noble, H. 2009. The Modelling4All project a web-based modelling tool embedded in Web 2.0. Rome, Italy, 1–6.

[10] Levy, S.T. and Wilensky, U. 2009. Crossing Levels and Representations: The Connected Chemistry (CC1) Curriculum. *Journal of Science Education and Technology*. 18, 3, 224–242.

[11] Levy, S.T. and Wilensky, U. 2008. Inventing a "Mid Level" to Make Ends Meet: Reasoning between the Levels of Complexity. *Cognition and Instruction*. 26, 1, 1–47.

[12] Macy, M.W. and Willer, R. 2002. From factors to actors: Computational sociology and agent-based modeling. *Annual review of sociology*, 143–166.

[13] Payette, N. 2014. *eXtraWidgets NetLogo extension. Centre for Research in Social Simulation, University of Surrey, Guildford: UK*.

[14] Sengupta, P. and Wilensky, U. 2009. Learning Electricity with NIELS: Thinking with Electrons and Thinking in Levels. *International Journal of Computers for Mathematical Learning*. 14, 1, 21–50.

[15] Tinker, R. and Wilensky, U. 2007. NetLogo Climate Change model. http://ccl.northwestern.edu/netlogo/Models/ClimateChange. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[16] Wilensky, U. NetLogo Fire model. 1997. http://ccl.northwestern.edu/netlogo/Models/Fire. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[17] Wilensky, U. 2001. Modeling nature's emergent patterns with multi-agent languages. *Procedings of EuroLogo*.

[18] Wilensky, U. 1999. NetLogo. http://ccl.northwestern.edu/netlogo. Center for connected learning and computer-based modeling. *Northwestern University, Evanston, IL*, 49–52.

[19] Wilensky, U. 2003. Statistical mechanics for secondary school: The GasLab multi-agent modeling toolkit. *International Journal of Computers for Mathematical Learning*. 8, 1, 1–41.
[20] Wilensky, U. 1997. Wolf Sheep Predation model. http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation. Center for Connected Learning and Computer-Based Modeling.
[21] Wilensky, U., Brady, C. and Horn, M. 2014. Fostering Computational Literacy in Science Classrooms. *Communications of the ACM*. 57, 8, 17–21.
[22] Wilensky, U. and Jacobson, R. 2014. Complex Systems in the Learning Sciences. *The Cambridge handbook of the learning sciences*. Cambridge University Press.
[23] Wilensky, U. and 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.
[24] Wilensky, U. and Resnick, M. 1999. Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and Technology*. 8, 1, 3–19.
[25] Wilkerson-Jerde, M.H. and Wilensky, U. 2010. Restructuring change, interpreting changes: The deltatick modeling and analysis toolkit. *Proceedings of constructionism*.
[26] Wolfram, S. 2002. A New Kind of Science, Champaign, IL: Wolfram Media.
[27] Bar-Yam, Y. 1997. *Dynamics of complex systems*. Addison-Wesley Reading, MA.