AI Magazine Volume 26 Number 3 (2005) (© AAAI)

# VModel

## A Visual Qualitative Modeling Environment for Middle-School Students

*Kenneth D. Forbus, Karen Carney, Bruce L. Sherin, and Leo C. Ureel II*

■ Learning how to create, test, and revise models is a central skill in scientific reasoning. We argue that qualitative modeling provides an appropriate level of representation for helping middle-school students learn to become modelers. We describe Vmodel, a system we have created that uses visual representations and that enables middle-school students to create qualitative models. Software coaches use simple analyses of model structure plus qualitative simulation to provide feedback and explanations. This system has been used in several studies in Chicago public school classrooms, using curricula developed in collaboration with teachers. We discuss the design of the visual representation language, how Vmodel works, and evidence from school studies that indicate it is successful in helping students.

Modeling is a central skill in scientific reasoning. Learning to formulate, analyze, test, and revise models is a crucial aspect of understanding science and is critical to helping students become active, lifelong learners. Supporting students in articulating models of a domain and refining the models through experience, reflection, and discussion with peers and teachers can lead to deeper, systematic understanding of science (Collins 1996). However, modeling is often treated as an art. Since modeling formalisms have traditionally been associated with creating mathematical models and deriving numerical results, they have been relatively inaccessible to younger students, such as middle-school students.

Qualitative reasoning (QR) formalisms provide the expressive power needed to capture the intuitive, causal notions of many human mental models (Forbus and Gentner 1997), including expressing aspects of modeling not handled by traditional formalisms, such as conditions of applicability and other types of modeling knowledge. However, the predicate calculus-based formalisms typically used in QR work provide a serious entry barrier to their use by children.

Our solution to this problem is to develop a visual representation language, based on concept maps, that provides a student-friendly way to express qualitative models and software that helps students create and refine qualitative models. This article describes the visual language and our modeling software Vmodel, which uses it. We start by motivating our approach. The design and software architecture of Vmodel is outlined next, focusing on its ontology, supports for the modeling process, and coaching. Evidence that suggests Vmodel helps students learn from classroom experiments is summarized. We close by describing related work and future plans.

## Approach

Why do we want students to become modelers? First, models provide a means to externalize thought. External representations help reduce working memory load, allowing students to work through more complex problems. External representations also help them present their ideas for discussion and collaboration, an important aspect of learning among young students (Brown and Campione 1996).

Second, the process of modeling itself is valuable. It forces students to articulate relationships between entities and dependencies between their beliefs. This is important both for understanding the phenomenon being modeled but also in developing a broader understanding of complex, interrelated systems, an important goal in the AAAS standards (AAAS 1989). The idea that a change in one variable may have far-reaching and unforeseen consequences is missing from many science curricula. The qualitative causal relationships developed by the QR community provide an appropriate level of expressive power for achieving this goal.

Third, modeling provides students with practice in using formal representations, a skill they need for mastering mathematics and programming. It provides them with a source of experience in the creative, liberating power of a technical vocabulary. Qualitative modeling, being grounded in everyday experience, can provide a bridge to more complex symbolic formalisms.

Given that we want to make students into modelers, how do we do it? Graphical external representations, coupled with computer support, can provide powerful ways for people to communicate and collaborate. Many graphical external representations have been created to aid students. They can be grouped into three families: concept map notations, dynamical systems notations, and argumentation environments.

*Concept map notations.* Concept maps (Novak and Gowin 1984, Cañas et al. 1995) describe structural and functional properties and relationships between entities and ideas. Typically, concept maps express global or time-invariant information.

*Dynamical systems notations.* Forrester's version of system dynamics (Forrester 1996), Bond graphs, and software systems such as STELLA[1] and Model-It (Jackson et al. 1996) provide graphical languages for expressing differential equations for continuous systems.

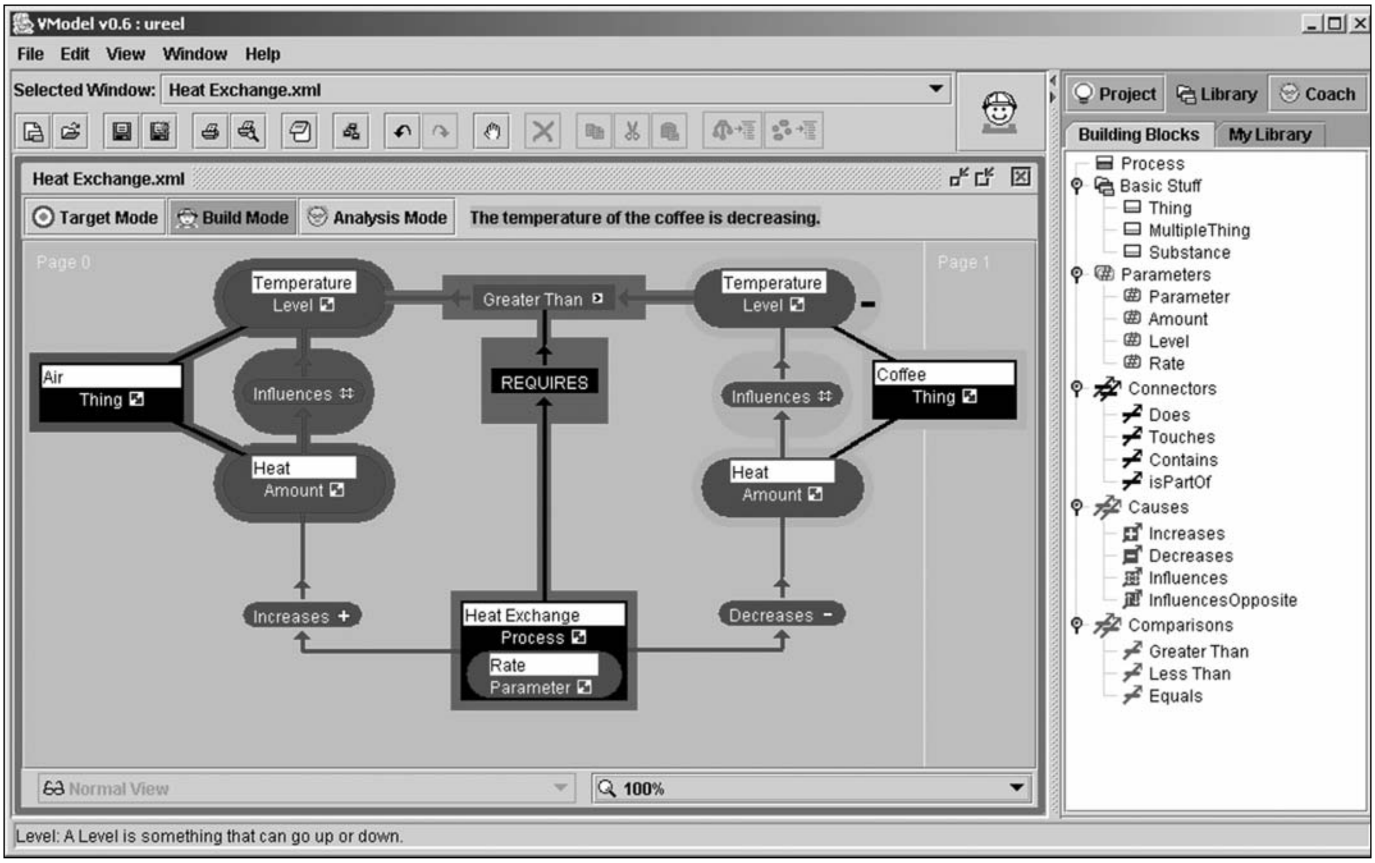
*Argumentation environments.* Belvedere (Suthers and Jones 1997) and the Collaboratory Notebook (Edelson, Pea, and Gomez 1996) are examples of software that uses graphical conventions to help students gather, create, and reason about evidence and arguments for and against hypotheses.

Each of these notations expresses some aspects of modeling, but none of them alone is sufficient. For example, concept maps adhere to minimal structural or semantic requirements. Although in theory they can be used to express anything, the lack of standardized semantics makes it difficult for students to understand each other's concept maps and extremely difficult to create software that detects whether or not maps are well formed (Cañas et al. 1995, Reichherzer et al. 1998). Dynamical system notations do not express the conditions under which a given model is applicable. Argumentation environments treat as atomic what would be whole complex structures in other notations, which limits their ability to scaffold students in detailed explorations.

Even taken together, these notational families and curricula based on them neglect two key issues in understanding the art of modeling.

The first key issue is *the importance of broadly applicable principles and processes.* Existing educational modeling systems treat each modeling task as a new problem, with no connection to other situations. This misses any opportunity to help students see that the same principles and processes operate across a broad range of situations. For example, the basic idea of heat flow is relevant to chemistry, biology, atmospheric physics, and many other areas that, on the surface, appear unrelated. Existing modeling systems do not help students see the importance of creating a systematic body of knowledge (as scientists do), as opposed to a series of ad hoc explanations concerning specific systems.

The second key issue is *qualitative understanding of behavior.* Modeling systems tend to be numerical (for example, STELLA), although they sometimes include a qualitative layer on top to simplify model creation (for example, Model-It). Understanding numerical data plots as depicting behavior is an important skill, but providing all the data needed to run a numerical model can distract students from understanding the causal phenomena in the situation. In addition, the level of mathematics necessary to make a model of any reasonably complex phenomenon may be out of reach for a young student. Mathematics shifts the representation away from the basic level entities a student might naturally attend to. For example, a mathematical model requires students to simultaneously represent all influences as a single equation or set of equations. This may be beyond the student. In addition, a student may not be able to easily interpret outcomes couched in mathematical terms or to debug wrong models.



The temperature of the coffee is decreasing.

Level: A Level is something that can go up or down.

*Figure 1. The Vmodel Interface.*

The theories, representations, and reasoning techniques developed in qualitative reasoning research provide most of the pieces needed to address these problems. Enabling and encouraging students to create their own domain theories should help them understand the broad applicability of scientific principles and processes. Qualitative modeling provides both the formalisms for expressing intuitive, causal models and the reasoning techniques needed to generate predictions and explanations from them for helping students see the consequences of their ideas. Making these formalisms available through a visual notation makes this power accessible to young students.

## The Design and Architecture of VModel

We have created a visual modeling tool for conceptual modeling that combines good ideas from concept maps and dynamical systems notations into a qualitative modeling environment. The basic organization uses concept maps, but with some very strong restrictions. As usual, nodes represent entities and properties of entities. However, each node has a specified type, such as Thing or Process, drawn from the system's ontology. As usual, links represent relationships. However, the labels used on links are drawn from a fixed set of relationships.

These restrictions provide a clearer semantics than traditional concept maps have. In traditional concept maps, any path of whatever length is intended to be a proposition, that is, a natural language statement that is true about what is being described. With our restrictions, links in our concept maps can be identified with propositional logic statements involving binary relations.[2] This makes software coaching more feasible than in traditional concept maps. Figure 1 shows the interface and a model.

These constraints address the trade-off between providing freedom of expression versus scaffolding for students. Providing their own names for entities and properties enables students to express their ideas more accurately (for example, "temperature" versus "hotness" versus "cold"). Requiring students to select a general type for entities and properties helps coaching software figure out which is which and avoids the need to do natural language understanding on the students' typed phrases. Re-

| Entities | Relationships |
|---|---|
| Processes: | Connectors: |
| Process | Does |
| | Touches |
| Basic Stuff: | Contains |
| Thing | isPartOf |
| Multiple-Thing | |
| Substance | Controllers: |
| | Requires |
| Parameters: | |
| Parameter | Comparisons: |
| Amount | Greater than |
| Level | Less than |
| Rate | Equals |
| | Causes: |
| | Increases |
| | Decreases |
| | Influences |
| | InfluencesOpposite |

*Table 1. The Vmodel Ontology.*

stricting links to a fixed set of relationships provides a powerful scaffold for students, ensuring that their ideas are at least in the ballpark in terms of form of argument.[3] It also forces students to enter a community of modelers, enabling their ideas to be compared and contrasted with those of others more easily.

The other basic feature of our design is the use of a *model library.* All of the models students build are included in their model library. The model library also contains extensions they create to the entity ontology, based on their modeling of specific situations, that is, a simple form of domain theory. This encourages students to generalize across situations. A student's model library thus represents the student's evolving understanding, providing a portfolio and support for reflection.

The underlying ontology used in Vmodel is that of qualitative process theory (Forbus 1984). Systems and phenomena are modeled via sets of entities with continuous parameters, whose relationships are expressed using a causal, qualitative mathematics, where processes provide an explicit notion of mechanism. Although QP theory was designed to capture everyday adult intuitions about the physical world, creating a student-friendly visual notation was one of the major design challenges in Vmodel.

In the remainder of this section, we describe Vmodel's ontology, the model library, workflow, and coaching.

## The Vmodel Ontology

Table 1 illustrates the basic ontology used in Vmodel. The contents are more or less familiar in AI representation practice, although some of the names have been changed to be child-friendly, based on our classroom experiences.

There are three kinds of entities. Processes represent the QP theory notion of physical process. Basic Stuffs represent objects (Thing, for example, an elephant), groups (Multiple-Thing, for example, a population), and substances (for example, water) in the world. Parameters represent the QP theory notion of quantity. As a form of scaffolding, we include subclasses for extensive parameters (Amount), intensive parameters (Level), and rates (Rate), since these are fundamental distinctions that middle-school students are unclear on, but must master. An unspecified type (Parameter) is also provided to support brainstorming.

The astute reader will notice that there is no relationship in the ontology presented to students that links a quantity to an entity. In Vmodel such links are unlabeled, because (1) they are extremely common, (2) we never found a concise term and explanation that didn't confuse students, and (3) students are perfectly happy not articulating this connection, given the intimate nature of the relationship between an entity and its continuous properties. An unlabeled line is used to relate an entity and a parameter of it, which in the interface is introduced by a right-click menu on the entity itself.

Connectors provide a small vocabulary of relationships to express configural information (for example, Touches, Contains, isPartOf). The relationship Does is used when a process is considered to be part of some entity (for example, a stomach does digestion). The conditions for a process are expressed using Controllers, that is, the Requires relationship. The Requires relationship links a physical process to a relationship (figure 1).

Value information about parameters is expressed through comparisons, that is, Greater Than, Less Than, and Equals. Given the curricula we have examined, we have not yet seen a requirement for $\leq$ or $\geq$.

Causes express QP theory's qualitative mathematics of influences. Influences are particularly appropriate because of their compositional nature: Each influence can be stated as a link, and the set of such links provides the set of influences on a parameter in a student's model.

Finding student-friendly names for these relationships took considerable experimentation. Here are the names we have found that work reasonably well.

*Increases/Decreases:* Indicates an integral connection between two parameters, that is, heat flow decreases the heat of its source and increases the heat of its destination. These are QP theory's direct influences, with Increases $\equiv$I+ and Decreases $\equiv$I-.

*Influences/InfluencesOpposite:* Indicates functional dependence between two parameters, that is, the heat of something determines its temperature. These are QP theory's indirect influences, with Influences $\equiv \propto_{Q+}$ and Influences Opposite $\equiv \propto_{Q-}$.

In early classroom trials, we experimented with an extended vocabulary that included relationships that didn't make a commitment to the sign (that is, $\propto_{Q*}$) and relationships that didn't make a commitment to the direction of causal relationship between two parameters. The intent was to allow students to first articulate and then refine their partial understandings. In practice, students using weaker relationships rarely refined them, hence their elimination.

Vmodel is designed to be very general in order to handle a wide variety of phenomena and so be relevant to a broad range of middle-school topics. Consequently, we allow certain aspects of the ontology to be extended. For example, teachers and curriculum developers are allowed to extend the vocabulary of Connectors as desired. We can allow this since Vmodel's reasoning software does not exploit the specific semantics of these vocabulary items. However, the rest of the relational vocabulary is fixed, since these terms are used in Vmodel's reasoning. Teachers and curriculum designers can also "seed" the model library with new entities and processes, using the same capabilities available to students, as described in the next section.

## The Model Library

The true power of modeling arises when students can use concepts they developed in earlier exercises to tackle more complex problems. Few modeling tools explicitly support this kind of reuse and abstraction, but we believe that it encourages the systematization of a student's knowledge. For example, maintaining a library of models and abstractions derived from them should facilitate transfer of ideas from one situation to another.

The model library contains two kinds of information. First, it includes a portfolio of all of the modeling projects a student (or a group of students) has tackled. The ability to refer back to previous exercises promotes reflection. The second kind of information in the model library is a catalog of entities and processes that the student has extracted from previous models to use as parts for future models. Adding an entity to the model library simply copies a template containing that entity, all of its quantities, and all relationships between those properties. Using the new entity in a subsequent model simply copies that piece of map into the current model. Adding a process to the model library causes a general template to be created. The participants of the process are identified by using any Requires links and causal relationships involving the process's rate parameter, and these participants are replaced by variables. When this template is used in a new model, these variables appear as diamonds, which can be dragged onto other elements of the model to "wire in" the process instance into the model. Thus students can accumulate a general library of physical processes.

One simplification we have made to keep Vmodel usable by middle-school students is to allow only a single level of extension to the process and entity ontologies. Ideally, one might want to have a full lattice of classes, with multiple layers. We have found that middle-school students are struggling with the idea of representing continuous properties by using parameters. Thus it seems unwise to burden them with additional subtleties at this stage.

### Supporting the Modeling Process

Modeling involves several different activities. We communicate and support this by organizing the software into several modes, each of which corresponds to a modeling activity. These are stating the problem, modeling, and analysis. We describe each in turn.

**Problem Statement.** Students are always asked to identify how one specific continuous parameter of the system will change (that is, increase, decrease, or remain constant), called the *modeling target.* A model is successful when its qualitative simulation produces the predicted result and yields a plausible explanation. We discovered through experimentation that such structuring is very important for students. Otherwise, they tend to treat modeling as a "brain dump," putting in everything that they can think of.

**Modeling.** The Build mode in the interface is where students create and edit their concept map, using the representations outlined above. Their specific prediction about a change in a parameter, based on their model target, is made here as well. Most of the students' time is spent in Build mode.

**Analysis.** The Analysis mode is where students analyze their model through simulation and re-

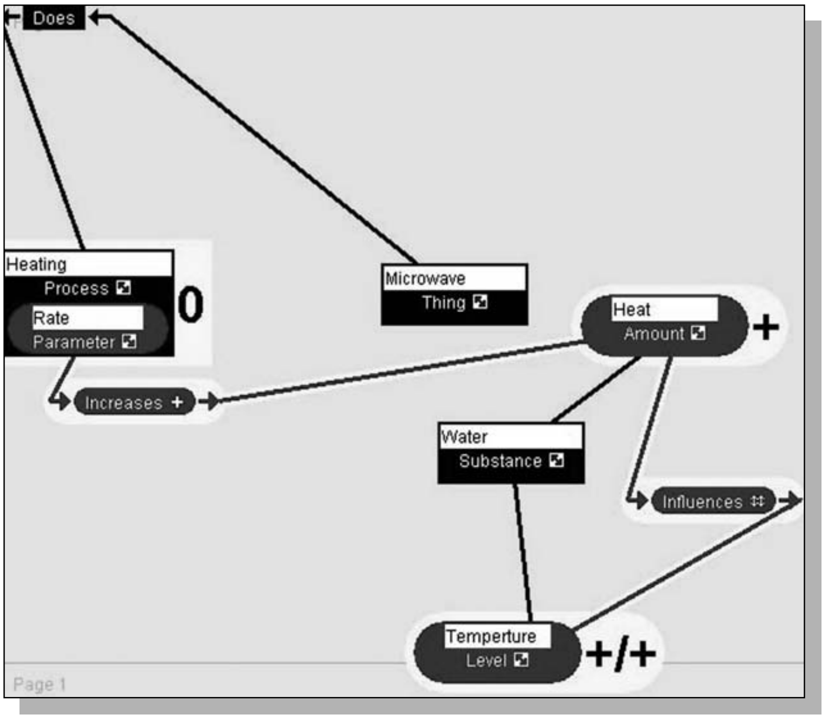


*Figure 2. Qualitative Simulation of a Student Model.*

flection. Upon entering Analysis mode, the students' model is analyzed by means of qualitative simulation. The qualitative simulation method we use is extremely simple. Every process instance in a model must be active. Therefore we can identify all directly influenced parameters by finding all the increases/decreases links, and resolve them using the laws of QP theory. If there is an ambiguity, due to conflicting direct influences, it is resolved by asking the student for the outcome. Influence resolution continues by propagating changes through indirect influences, until directions of change for all parameters have been found. (Uninfluenced parameters are of course unchanging, as per QP theory.)

While simple, this qualitative simulation method is very appropriate for middle-school modeling tasks. We deal with only a single (tacit) qualitative state, avoiding the complexity of branching behaviors and envisioning. All rates are tacitly assumed to be positive. These assumptions suffice for the vast majority of the examples that middle-school students deal with. Resolving ambiguous influences by asking the students is straightforward and easy for them to understand. Unfortunately, this means the software cannot support discussion of relative magnitudes and when effects can be neglected. On the other hand, we have seen little need for such discussions so far.

The results of qualitative simulation are provided to the student in terms of a textual explanation. For example, this explanation is generated from a model of heating water in a microwave oven:

> There is the process of Heating which
>
> > Increases the Heat of Water that is INCREASING and which
> >
> > Influences the Temperature of Water that is INCREASING

The generated text serves several pedagogical purposes. First, it encourages students to name and decompose their entities properly, since the templates include conventions such as "<quantity> of <entity>." This overcomes a student tendency to lump things together by relying on their language skills (for example, "Trixie eating does eating" grates, even though the fragment of concept map that gives rise to the statement at first did not.) Second, it illustrates the process that students should follow when reasoning with causal models on their own. This point is reinforced by animation used during qualitative simulation, where the quantity currently being considered is highlighted and intermediate results are displayed as graphical annotations on the student's model. Figure 2 illustrates the end state of a VModel qualitative simulation. In this student model, the "+/+" indicates that the students predicted the parameter would increase, and the model successfully explains this prediction.

## Supporting Students in Modeling

We have built in a number of supports to help students learn modeling. The most basic is an embedded HTML-based help system, which provides answers to questions about the software, modeling (for example, How do I know when I am done?), and the representation system. Most of the interface design was driven by scaffolding needs. For example, when a Process is created, it always includes inside itself a Rate parameter, since every process must have one (see figure 1). Similarly, the use of "skins" to provide visual cues for grouping helps deal with complex maps. Each entity is assigned a unique color, and the skin grows to include every quantity of it. Moreover, the skin for a process expands to include the Requires link and the antecedent relationship it links to.

Through several years of experimentation in schools, we found two kinds of software coaching to be useful. The first concerns structural properties of models, and the second concerns the qualitative simulation of the model. We discuss each in turn.

**Structural Coaching.** Students learning the visual notation often have trouble with its syntax and grammar, that is, what kinds of relationships can be used to connect what sorts of elements. We call the coach for this the Builder. The Builder uses a simple set of rules to identify when the students are using modeling elements incorrectly, when a more appropriate modeling element is available, or when there are missing elements or inconsistencies in their model. For example, two of the Builder's rules, stated in English, are: "A direct influence relationship must be connected between the rate of a process and any type of parameter" and "No type of parameter can have both a direct influence and an indirect influence as input."

In addition to scaffolding the student in modeling, the Builder also acts as a gatekeeper for Analysis mode. That is, the student cannot proceed until the Builder is satisfied. This prevents the qualitative simulator from having to deal with models that do not satisfy the constraints of QP theory (which of course is a different issue from whether or not the model's contents are correct).

The Builder indicates when it finds a problem in two nonintrusive ways. First, an icon always depicts the state of the Builder. The default icon is smiling, but as the student's model accumulates errors, it is replaced by a puzzled-looking icon. If a large number of errors are made, a frowning icon is used. Second, parts of the model that are implicated in problems are highlighted in red. Students can query the coach as to the nature of the problem(s) by clicking the coach icon or by right-clicking the indicated modeling element. The coach responds by displaying its analysis of the current state of the model. It is important that the coach be nonintrusive because editing a model often involves temporary inconsistencies.

**Analysis Coaching.** When the student enters Analysis mode, another coach, the Professor, becomes available. The Professor provides help in interpreting the results of a student's simulation. For example, if there is a mismatch between the student's prediction and the results of the simulation, this is noted as part of the text generated along with the simulation, for example,

> Model Check
>
> <u>Level</u> <u>Temperature</u>
>
> You predicted that Temperature would be DECREASING but instead it is INCREASING

with the offending parameter highlighted in red (underlined here). Like the Builder, the Professor is depicted by icons indicating various degrees of contentment or alarm, depending on the student's results.

## Classroom Experiences

The classroom portion of this research has been conducted as part of the National Science Foundation's Center for Learning Technologies in Urban Schools (LeTUS), a partnership involving Northwestern University, University of Michigan, and the Chicago and Detroit public school systems. The center developed inquiry-based middle-school science curricula. These curricula were developed in work circles, a collaborative organization that involves researchers, teachers, and school administrators in developing and adapting materials for classroom settings. This joint development arrangement provided invaluable feedback and supports experiments in a variety of urban schools.

We helped Chicago public school (CPS) teachers develop two curricula. One curriculum concerns heat and temperature, in which students consider alternate energy resources for homes.[4] In collaboration with Marcia Linn's group at Berkeley, we added complementary simulation-based activities to their successful thermal curriculum (Linn and Songer 1991). These activities use self-explanatory simulators (Forbus and Falkenhainer 1990) to allow students to explore the outcomes of design choices in making a solar house. The other curriculum concerns ecosystems, using as a hook the creation of a life-support system for a Mars colony. This provides an arena for students to explore the requirements of life and how ecosystems work, using simulation experiments.

Our design process for the visual notation was guided by pencil and paper studies we carried out in CPS classrooms. The first pilot studies with the software took place in winter and spring of 2001, in a bilingual magnet school in Chicago. Eleven sixth graders (aged 11 and 12) worked together, in groups of two or three, creating models related to the Mars curriculum. The task was initially posed in terms of relationships between two species (for example, grass and antelopes), and students were asked to extend their models to handle new species. The second and third studies (fall 2001, winter 2002, spring 2003) also used the Mars curriculum. In a seventh-grade classroom, students completed a combination of classwide and small-group modeling exercises to plan an ecosystem from scratch. Students working in small groups were asked to discuss and justify their models in front of the entire class. Data was collected through videotaping, clinical interviews, and portfolios of models constructed.

Since this article is about the design and architecture of Vmodel, we focus on results that bear on these choices. In particular, between the second and third studies, we added most of the modeling process supports (for example, the model target) and the qualitative simulator.

*Naturalness of the visual language.* After some tuning, student questions tend to focus on the contents of the models, rather than on how to use the primitives of the modeling language. However, that does not mean that students always used these concepts correctly. In the first two studies, 67 percent (198/296) of boxes and 44 percent (113/258) of arrows were used correctly. In the third study, correct box use rose to 74 percent (222/298) and correct arrow use rose to 96 percent (221/230).

*Structure of causal chains.* Prior to introducing stronger coaching, student models exhibited a variety of flaws in their causal models. Causal paths were often short, with several disconnected paths rather than a single path explaining a change. Students would often leave out processes and attempt to specify direct entity-to-entity relationships rather than articulate the specific quantities involved. After introducing stronger coaching, student causal paths became longer and more focused on explaining changes specified in the target.

These observations suggest that qualitative simulation and coaching are actually providing significant value in Vmodel. Is Vmodel helping students learn modeling? Here the analysis of data is still in progress, but the results so far are very encouraging.

*Generalization in modeling.* Even though there have not been many opportunities yet in the curriculum to make generalizations based on their models, we are finding that students do indeed start exploiting abstractions. One student for instance, used a model of gazelles and grass to model the interaction between lions and gazelles by making the appropriate substitutions. This is not an easy thing for students to grasp. In one classroom discussion, students were working out what to name a model they had created that described an astronaut's weight gain or loss in terms of the astronaut's caloric intake and exercise and metabolic needs. Could the model be used to explain more than just astronauts? How could it be made truly general, if it wasn't already? One girl ventured "I think it should be called 'the calorie cycle' because you could take out ASTRONAUT and replace it with DOG, and that model would explain both." That is the kind of insight that we want all students to attain.

Unfortunately, we are also finding some places where further improvements could be made.

*Comparison and reflection were lacking.* Students often viewed their model as finished when "both coaches were happy." This is not surprising given how work is structured in middle-school classrooms. Similarly, students sometimes viewed reusing models from their library as "cheating."

*Causal map redundancy.* Students are able to use physical process descriptions correctly. However, they sometimes include extra statements that summarize the result of influence chains in a particular situation. For instance, they might have that the rate of predation of gazelles by lions is influenced by the population of lions and that the rate of predation decreases the population of gazelles, which is a correct partial description of predation. But they will often add an additional relationship between population of gazelles and population of lions, in an attempt to summarize the effect of the mechanism.

We will discuss our plans for tackling these problems next.

## Plans for Additional Coaching

We are designing an analogy-based (Forbus, Gentner, and Law 1995) coaching facility to provide feedback based on normative models. (Normative models are authored using the same software, so that teachers and curriculum developers can add content without our being involved.) The candidate inferences of a comparison of a student model to a normative model can provide suggestions for what a student might want to think about in order to improve the model (Collins and Stevens 1982).

On the other hand, there are infrastructure difficulties. Students working in schools with network connections will be able to use an email facility built into the software. Unfortunately, many classrooms are not so endowed, and in those cases we resort to carrying a floppy or laptop back with us to get the data or asking the teachers to email student work to us later using their personal accounts. Even if every classroom were networked, we would continue to use email, rather than web-based, coaching for scalability. One does not expect an instant reply to an email message, whereas a web server that does not respond instantly is annoying. Our collaborating teachers find this model reasonable because students work only a small part of any typical day on science activities, and that time is spent in discussions as well as computer work.

For people unconstrained by such computational resource limitations, we are planning a knowledge entry and analysis tool based on VModel with supercharged modeling and analysis capabilities. We plan to merge VModel-like concept mapping with our open-domain sketching system, sKEA (Forbus and Usher 2002). To this we will attach a domain-independent, mixed-initiative Socratic tutor, which will provide students and engineers with a companion with whom they can engage in a dialogue for assistance in model development and analysis.

## Related Work

We know of two projects that are very similar to ours in both spirit and approach. One is the UWF Quorum project (Cañas et al. 1995, Reichherzer et al. 1998), which used concept maps to let student express and share a wide variety of ideas, both within their schools and with students in other countries. Quorum's success encouraged us to consider the use of restricted concept maps as a visual notation for qualitative modeling. Their use of an "artificial idiot," the Giant, is a different approach to coaching than ours, where we use a simple version of the persona effect (Lester et al. 1997) to make feedback more palatable for students. The second is "Betty's Brain" at Vanderbilt (Biswas et al. 2001), where they are using qualitative representations in concept maps to foster learning. The task they use, of "teaching" Betty (their software) by building concept maps so that Betty can produce explanations, is inspired. Their domain-specific qualitative modeling framework uses tables for composing discrete values to provide qualitative simulation. Our qualitative modeling framework is richer, incorporating physical processes and a student-extendable ontology. We also support the creation of new abstractions from student models, unlike the Vanderbilt software.

## Discussion

This article summarizes our effort to create a representational system with computer support that enables middle-school students to learn to be modelers. By using qualitative representations as a formal semantics for a restricted concept map language, we have created a visual notation for building models in terms of conceptual knowledge, rather than requiring mathematics as traditional modeling systems do. Vmodel provides scaffolding that helps students learn to build causal models with it. Using software coaches to help students avoid the modeling equivalent of grammatical errors and using qualitative simulation help the students reason through their models.

Our school studies provide evidence that we have been successful in helping students become modelers. We are particularly encouraged by spontaneous uses of the visual qualitative language by students outside of the software; it suggests that this way of looking at the world is becoming a habit of mind, something that they, not the software, own. While Vmodel is aimed at middle-school students, we believe that this combination will be very useful for a wide range of students. One of our hypotheses, which cannot be tested until Vmodel spreads further through the schools, is that having better conceptual models will facilitate later learning of mathematics and mathematical modeling.

Educationally, bringing modeling into science curricula more broadly has the potential to help students become full-fledged modelers, engaged in the joy of unraveling complex phenomena rather than frustrated by memorizing mountains of isolated facts.

Our vision for modeling environments is that they should become as natural for modelers as word processors are to writers and spreadsheets are to accountants. Ideally, they can be used in all stages of modeling, from gathering and summarizing the phenomenon to be explained to initial model formulation to refinement through testing the consequences of the model against data. Most modeling tools focus on later, quantitative-oriented tasks in the modeling process. Given our target audience of middle-school students, Vmodel focuses exclusively on conceptual modeling. But one can imagine using it as part of a larger suite of tools that encompass the whole modeling process, for both learning and professional work.

## Acknowledgements

This research is supported by the National Science Foundation under the REPP program. Rob Harris provided additional programming assistance. We have had the invaluable support of several Chicago public school teachers in developing curricula and thank them for allowing us to use their classrooms for testing purposes. These teachers are Beverly Miller, Deborah Rogers-Green, Adam Dorr, Carol Scafide, Kiesha Korman, Carlos Rodrigues, and Judy Whitcomb.

## Notes

1. High Performance Systems, STELLA software, www.hps-inc.com/.

2. This restriction to binary relations entails no loss of expressivity for two reasons. First, higher-arity relationships can be expressed by means of reification. Second, the use of compositional relationships such as influences enables the assembly of more complex statements by underlying reasoning systems.

3. For example "Pixies do it" and other anthropomorphic arguments are simply not expressible.

4. See M. C. Linn's Houses in the Desert project,. http://wise.berkeley.edu/WISE/about/houses/.

## References

AAAS. 1989. *Science for All Americans: A Project 2061 Report on Literacy Goals in Science, Mathematics, and Technology.* Washington, DC: American Association for the Advancement of Science.

Biswas, G.; Schwartz, D.; Bransford, J.; and the Teachable Agents Group at Vanderbilt University. 2001. Technology Support for Complex Problem Solving: From SAD Environments to AI. In *Smart Machines in Education: The Coming Revolution in Educational Technology,* ed. K. Forbus, and P. Feltovich. Menlo Park, CA: AAAI Press.

Brown, A. L.; and Campione, J. C. 1996. Guided Discovery in a Community of Learners. In *Classroom Lessons: Integrating Cognitive Theory and Classroom Practice,* ed. K. McGilly, 229–270. Cambridge, MA: The MIT Press.

Cañas A. J.; Ford K. M.; Brennan J.; Reichherzer T.; and Hayes P. 1995. Knowledge Construction and Sharing in Quorum. Paper presented at the Seventh World Conference on Artificial Intelligence in Education. Washington, DC, August 16–19.

Collins, A. 1996. Design Issues for Learning Environments. In *International Perspectives on the Design of Technology-Supported Learning Environments*, ed. S. Vosniadou, E. D. Corte, R. Glaser, and H. Mandl, 347–362. Mahwah, NJ: Lawrence Erlbaum.

Collins, A., and Ferguson, W. 1993. Epistemic Forms and Epistemic Games: Structures and Strategies to Guide Inquiry. *Educational Psychologist* 28(1): 25–42.

Collins, A., and Stevens, A. L. 1982. Goals and Strategies of Inquiry Teachers. In *Advances in Instructional Psychology,* Volume 2, ed. R. Glaser, 65–119. Hillsdale, NJ: Lawrence Erlbaum.

Edelson, D. C.; Pea, R. D.; and Gomez, L. M. 1996. The Collaboratory Notebook: Support for Collaborative Inquiry. *Communications of the ACM* 39(4): 32–33.

Forbus, K. 1984. Qualitative Process Theory. *Artificial Intelligence* 24(1–3): 85–168.

Forbus, K., and Falkenhainer, B. 1990. Self-Explanatory Simulations: An Integration of Qualitative and Quantitative Knowledge. In *Proceedings of the Eighth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Forbus, K., and Gentner, D. 1997. Qualitative Mental Models: Simulations or Memories? Paper presented at the Eleventh International Workshop on Qualitative Reasoning, Cortona, Italy.

Forbus, K.; Gentner, D.; and Law, K. 1995. MAC/FAC: A Model of Similarity-Based Retrieval. *Cognitive Science* 19(2) (April–June): 141–205.

Forbus, K., and Usher, J. 2002. Sketching for Knowledge Capture: A Progress Report. In *Proceedings of the International Conference on Intelligent User Interfaces* (IUI'02). New York: Association for Computing Machinery.

Forrester, J. W. 1996. System Dynamics and K–12 Teachers. Lecture presented at the University of Virginia School of Education, May 30.

Jackson, S.; Stratford, S. J.; Krajcik, J. S.; and Soloway, E. 1996. A Learner-Centered Tool for Students Building Models. *Communications of the ACM* 39(4): 48–50.

Lester, J.; Converse, S. A.; Kahler, S. E.; Barlow, S. T.; Stone, B. A.; and Bhoga, R. S. 1997. The Persona Effect: Affective Impact of Animated Pedagogical Agents. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 359–366. New York: Association for Computing Machinery.

Linn, M. C., and Songer, N. B. 1991. Teaching Thermodynamics to Middle School Students: What Are Appropriate Cognitive Demands? *Journal of Research in Science Teaching.* 28(10): 885–918

Novak, J. D., and Gowin, D. B. 1984. *Learning How to Learn*. New York: Cambridge University Press.

Reichherzer, T. R.; Cañas, A. J.; Ford, K. M.; and Hayes, P. J. 1998. The Giant: A Classroom Collaborator. Paper presented at the Fourth International Conference on Intelligent Tutoring Systems Workshop on Pedagogical Agents, 83–86. Berlin: Springer-Verlag.

Suthers, D., and Jones, D. 1997. An Architecture for Intelligent Collaborative Educational Systems. 1997. Paper presented at the Eighth World Conference on Artificial Intelligence in Education, Kobe, Japan, August 20–22.

**Kenneth D. Forbus** is the Walter P. Murphy Professor of Computer Science and Education at Northwestern University. His research interests include qualitative reasoning, analogy and similarity, sketching and spatial reasoning, cognitive simulation, reasoning system design, articulate educational software, and the use of AI in computer gaming. He received his degrees from the Massachusetts Institute of Technology (Ph.D. in 1984). He is a fellow of the American Association for Artificial Intelligence and an associate editor of *Cognitive Science* and serves on the editorial boards of *Artificial Intelligence,* AAAI Press, and the Adbivory Board of the *Journal of Game Development*. His e-mail address is forbus@northwestern.edu.

**Karen Carney** is completing her Ph.D. in the learning sciences at Northwestern University. She is also a visiting research associate at the Center for the Study of Learning, Instruction, and Teacher Development at the University of Illinois at Chicago. Carney's areas of interest include science learning, roles of educational technology in the classroom, and student use of representational formalisms.

**Bruce Sherin's** work focuses on the study of science learning, particularly as it occurs within novel technology-based learning environments. In early work, he engaged in the design and study of novel interventions for physics instruction, with particular emphasis on the role of symbolic representations. More recently, he received an NSF Early Career Award to develop new frameworks for understanding the learning that occurs in project-based science instruction.

**Leo Ureel** is a Ph.D. candidate in computer science at Northwestern University. He received his B.S. and M.S. degrees in computer science from Michigan Technological University. His research interests include Socratic thought, constructionism, intelligent tutoring systems, visual languages, and software metrics. His email address is ureel at northwestern.edu.