Interactive Learning Environments, 2013
Vol. 21, No. 4, 371–413, http://dx.doi.org/10.1080/10494820.2013.803125

Model construction as a learning activity: a design space and review

Kurt VanLehn*

Computing, Informatics and Decision Science Engineering, Arizona State University, Tempe, AZ 85284, USA

(*Received 20 February 2013; final version received 3 May 2013*)

Modeling is becoming increasingly important both as a way to learn science and mathematics, and as a useful cognitive skill. Although many learning activities qualify as "modeling", this article focuses on activities where (1) students *construct* a model rather than explore a given model, (2) the model is expressed in a *formal language* rather than drawings, physical objects or natural language texts and (3) the model's predictions are generated by *executing* it on a computer. Most research on such learning activities has focused on getting students to successfully construct models, which they find very difficult to do. In the hope that new research can find ways to remove this bottleneck, this article attempts to list all the major ideas that have appeared in the literature and might be useful to those developing new learning activities involving model construction. The ideas are organized into a design space with five dimensions: (1) modeling language types, (2) ways for describing the systems that students should model, (3) instructional objectives and their corresponding assessments, (4) common student difficulties and (5) types of scaffolding.

Keywords: modeling; model construction; scaffolding; interactive learning environments; interactive learning activities; constructive learning

1. Introduction

The literature on educational uses of modeling includes several recent reviews (Clariana & Strobel, 2007; Clark, Nelson, Sengupta, & D'Angelo, 2009; Doerr, 1996; Hopper & Stave, 2008; Jacobson & Wilensky, 2006; Jonassen & Strobel, 2006; de Jong & van Joolingen, 1998; Löhner, 2005; Penner, 2001; Stratford, 1997). Although this article is a review in that it presents no new work, it is not intended to be an exhaustive list of relevant studies. Instead, it is intended to be an exhaustive list of the major ideas that have appeared in the literature and might be useful to new research. The ideas are organized into a five-dimensional design space. That is, the discussion separately addresses five questions that must be addressed in designing and evaluating instruction that uses model construction:

(1) What type of model should I have students construct?
(2) How can students find out about the systems they will model?
(3) How can students' learning be assessed?
(4) What difficulties in learning can be anticipated?

*Email: kurt.vanlehn@asu.edu

(5) What scaffolding can be used to help students learn while modeling?

This organization was chosen simply to encourage more studies of model construction. As argued in the next section, model construction is becoming increasingly prominent as an instructional objective in science, mathematics and engineering, but we have still not found efficient, effective methods for teaching model construction. As this review demonstrates, the design space of model construction activities is vast and relatively unexplored, so further experimentation may find instructional methods that work better than those that have been studied so far.

2. Model construction is important

Model construction has been important and ubiquitous in recent US standards for K-12 science, mathematics and engineering. Standards for science instruction (National Research Council, 2012) contain over 350 occurrences of "model" and "modeling". The report has only seven strands that are threaded throughout the standards, and model construction is one of them. There are sections in the report devoted exclusively to defining model construction and describing standards for proficiency in modeling. In many of the report's exemplary learning progressions, modeling activities are mentioned explicitly.

In the Common Core State Mathematics Standards (CCSSO, 2011), modeling is one of only seven mathematical practices. Unlike the content topics (e.g. fraction addition and the Pythagorean theorem), modeling is not allocated a standard of its own but is instead indicated by placing stars on content topics taught at the high-school level where modeling should be applied. For example, there is a star on "Create equations and inequalities in one variable and use them to solve problems", whereas there is no star on "Solve systems of linear equations exactly and approximately (e.g. with graphs)". Of the 117 high school topics, 45% involved modeling.

Although there currently are no standards for K-12 engineering, a National Academy of Engineering committee studying K-12 engineering education highlighted the importance of modeling in such instruction when they said, "Mathematical analysis and modeling are essential to engineering design…" (Katehi, Pearson, & Feder, 2008, p. 25).

Several theoretical frameworks suggest that modeling is important for achieving deep understanding. For instance, Penner (2001) argues that constructivism strongly implies that model construction will be effective if not essential to understanding science.

Studies of professional and/or highly educated decision-makers suggest that they need to learn to model systems. In particular, their informed, experienced judgments were not as accurate as simple models (Booth Sweeney & Sterman, 2000; Kainz & Ossimitz, 2002; Moxnes, 2000).

3. Model construction vs. other modeling activities

In the science, mathematics and engineering community, the term "modeling" usually means constructing an expression in a formal language that denotes properties of a given system. Often the formal language is mathematical equations, a computer programming language or a mixture of the two languages. The process of model construction involves not only constructing an initial model, but also testing its predictions against the thing being modeled, which will be called "the system". If the predictions are inaccurate, the model must be revised, which is often called "debugging" the model. For the class of models considered here (which will be defined more precisely later), generating the

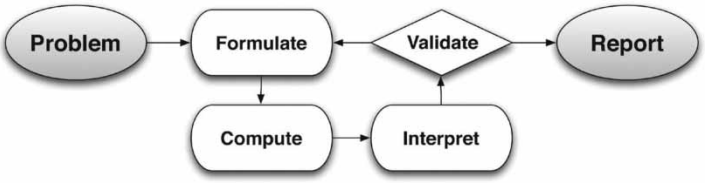

Figure 1. Modeling is the process of constructing and debugging a model (CCSSO, 2011).

model's predictions involves some kind of numeric or symbolic computation that can be done by a computer. Thus, the overall process is a cycle of several subprocesses, as shown in Figure 1, which is borrowed from the Common Core Mathematics Standards (CCSSO, 2011). In short, in the professions, "modeling" usually means "model construction".

In the science education community, the term "modeling" is often used more broadly to encompass any educational activity that involves a model. A particularly common activity is to give students a model of a system and ask them to understand it. Students control the model by manipulating sliders or other controls, and they observe its behavior via gauges, graphs, animations or other displays. Often this activity is called *model exploration*.

The activities of model construction and model exploration are very different (Alessi, 2000b). Model construction requires knowing the modeling language and executing the processes shown in Figure 1. Model exploration requires neither knowing the modeling language nor executing the problem solving process of Figure 1. Consequently, model-construction activities have a steeper entry cost than model-exploration activities, because they require learning the modeling language and the skills involved in the cycle of Figure 1.

One would hope that that this extra cost is paid back with larger learning gains, and there is some evidence of this. Hashem and Mioduser (2010, 2011) compared four instructional activities involving NetLogo models: one was model construction and the other three were different types of model exploration. The model-exploration groups had 2 hours of training, whereas the model construction group required 48 hours of training in order to gain sufficient skill in NetLogo programming. As expected, the model-construction group scored higher on the post-test as well on measures of post-intervention interviews. Although NetLogo is much more complicated than most other modeling languages and thus took a long time to learn, the experiment demonstrates some extra benefit for the extra cost.

The rest of this article describes a design space for learning activities based on model construction. The intention is to list major ideas that have appeared in the literature and might be usefully combined or reworked to form new instructional activities. The ideas are organized into five more-or-less independent dimensions, each discussed in its own section (Table 1).

4. Modeling languages

Because model construction is so different from model exploration, this review covers only model-construction activities. Such activities can be classified along two dimensions: (1) the type of model being constructed by the students and (2) the way the system is presented to them. This section considers model types; the following section discusses methods for presenting systems.

Table 1. A design space for learning activities involving model construction.

1. Selected modeling languages
   a. Constraints on system states
   b. System dynamics
   c. Agent-based
2. Methods for presenting systems
   a. Succinct text
   b. Resources
   c. Simulation
   d. Virtual labs and virtual field studies
   e. Real labs and real field studies
3. Instructional objectives and assessments
   a. Improved model-construction skill
   b. Improved domain knowledge
   c. Improved understanding of a particular system
   d. Concept inventories and mental modeling
   e. Understanding the role of models
   f. Less easily assessed objectives
4. Common student difficulties
   a. Poor problem solving strategies
   b. Difficulties understanding the modeling language
   c. Difficulties understanding the presentation
5. Types of Scaffolding
   a. Tutoring
      i. Feedback/hints on the model
      ii. Feedback/hints on the student's process (meta-tutoring)
      iii. Concrete articulation strategy
      iv. Decomposition into subsystems
      v. Reflective debriefings
      vi. Answering student questions
   b. Clarifying the modeling language
      i. Notation for models
      ii. Grounding the symbols
      iii. Comparing predictions to system's behavior
      iv. Students explaining their model
   c. Gradually increasing complexity
      i. Hiding model features
      ii. Qualitative model construction as scaffolding
      iii. Model progressions
   d. Other scaffolding
      i. Teachable agents and reciprocal teaching
      ii. Mental execution of models
      iii. Test suites
      iv. Generic schemas
      v. Gamification

Several frameworks for defining and classifying types of models have been proposed (Clariana & Strobel, 2007; Harrison & Treagust, 2000; Hestenes, 2007; Stratford, 1997). Collins and Ferguson (1993) developed a particularly exhaustive taxonomy of "epistemic forms". An epistemic form is a language or syntax for writing models, and it is not a model itself or a theory. Although "epistemic form" is an admirably precise term, it did not catch on. Thus, "modeling language" will be used here in place of "epistemic form". Only some of the modeling languages are used in school science classes (Harrison & Treagust, 2000).

Table 2. A taxonomy of modeling languages from Collins and Ferguson (1993).

Structural analyses: Decompose the system into parts and describe relationships among parts
• *Spatial decompositions,* e.g. anatomical diagrams or circuit diagrams.
• *Temporal decompositions* or *stage models,* e.g. a series of states or phases.
• *Compare and contrast* listings, e.g. of the solar system versus the Bohr atom
• *Cost-benefit analysis*. Comparing and contrasting two economic alternatives.
• *Primitive-elements game*, e.g. how quarks etc. combine to make protons, electrons, etc.
• *Cross-product* or *table game,* e.g. the periodic table of elements.
• *Tree-structure* or *hierarchy game*, e.g. biological taxonomies
• *Axiom systems*, e.g. Euclid's geometry or Peano's arithmetic.
Functional analyses: Determine the causal or functional relationships among elements of a system
• *Critical-event analysis,* e.g. of an airplane crash or the invention of the printing press.
• *Cause-and-effect analysis*. A sequence of events, each causing the ones after it.
• *Problem-centered analysis*. A problem's solution has side effects which pose problems that…
• *Multicausal analysis* or *AND/OR graphs*.
• *Form-and-function analysis*.
Behavior analyses: Describe the dynamic behavior or process of the system.
♥ *Constraint systems.* They determine the space of possible states of the system.
♥ *System-dynamics models*. They determine how the system changes over time.
♥ *Aggregate-behavior models*, e.g. of slime molds. Often called emergent or agent-based.
• *Situation-action models*, e.g. production systems, Markov models, finite state transition nets
• *Trend* and *cyclical analyses*, e.g. finding leading indicators for picking stocks

Table 2 shows the Collins and Ferguson taxonomy of modeling languages. Collins and Ferguson divide modeling languages into those for analyzing the structure, function and behavior of systems. The structure of a system is like its anatomy; structure models describe the parts of a system and their relationships. The behavior of a system is how it changes over time. The function of a system refers to the purpose of the system and how that purpose is achieved. Although there are other ways to divide up the analyses of systems – for example, de Kleer and Brown (1981) use just structure and function, and Weld (1983) uses role, function, structure and mechanism – this three-way division of structure, behavior and function is often found in the literature (Erden et al., 2008; Goel, Rugaber, & Vattam, 2009; Hmelo-Silver & Pfeffer, 2004).

This review will address only a small number of the languages shown, namely three of the modeling languages used for analyzing the behavior of systems. These languages are marked in two with a ♥ bullet. Moreover, only executable languages will be reviewed. That is, after the user has constructed a model in the language, the user presses a "run" button and the modeling software calculates predictions about the behavior of the system. Because these languages are all similar to a degree, there is a chance that this review can find commonalities in their instructional objectives, assessments, obstacles and scaffolding. Now let us consider in more detail each of the three modeling language types to be reviewed.

4.1 *Constraint systems*

Constraint systems predict the possible states of the behavior of a system. Thus, if one quantity of the model is perturbed, a constraint system predicts how the rest of the system must change in order to remain in a legal state. The model consists of a set of variables and a set of constraints on the values of those variables. The constraints are usually mathematical equations or a set of qualitative equations. If there are N variables, then the equations determine which points in the N-dimensional space correspond to possible system states.

If one of the variables represents time and the equations are temporal differential or difference equations, then the model is classified as a system-dynamics model rather than a constraint system. System-dynamics models are described later in this section.

For *quantitative* constraint systems, several instructional systems have been built for algebraic model construction (McArthur et al., 1989) and arithmetic model construction (Marshall, Barthuli, Brewer, & Rose, 1989). They usually have students create graphs and tables, then draft equations which are then solved by the system or by the student. When a tutoring system is involved, it monitors the student's progress in developing the equations, offers feedback and hints when asked, and offers help when it detects that the student is floundering. When no tutoring system is involved, the students typically enter equations into a spreadsheet or a computer algebra system such as MatLab or Mathematica. The program solves the equations, allowing the student to focus on constructing the appropriate equations.

For *qualitative* constraint systems, node-link diagrams are used to represent models in most languages, including these major ones:

- Betty's Brain (Leelawong & Biswas, 2008),
- Garp and DynaLearn (Beek, Bredeweg, & Lautour, 2011; Bredeweg, Liem, Beek, Salles, & Linnebank, 2010; Bredeweg, Linnebank, Bouwer, & Liem, 2009),

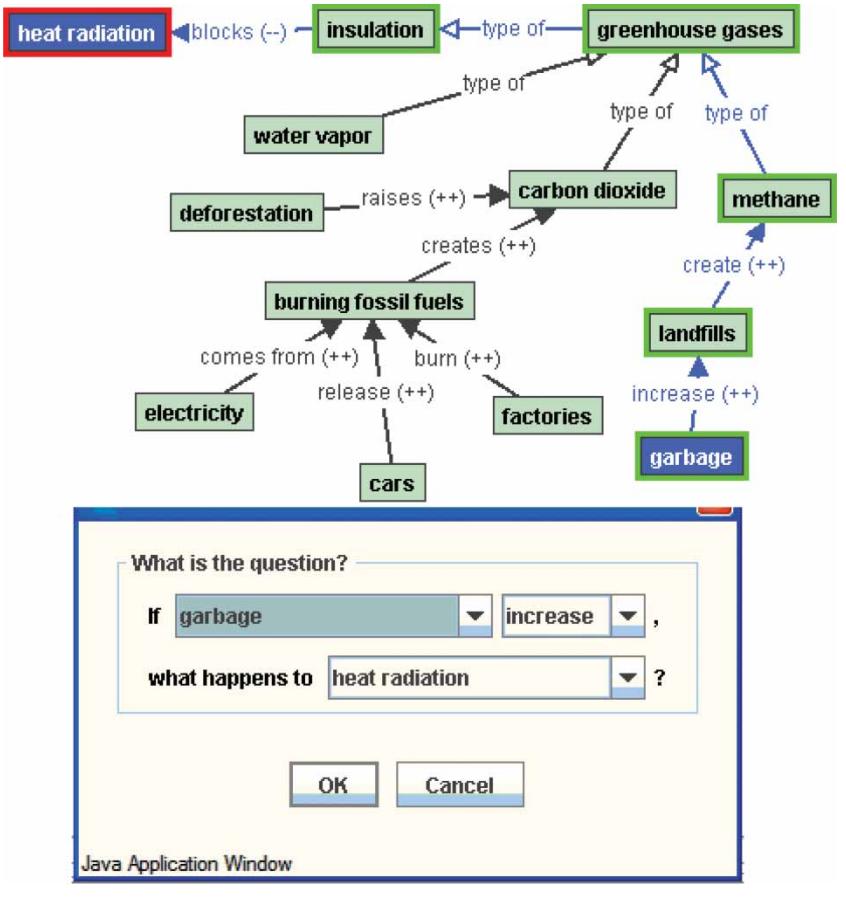

Figure 2. A model and query from Betty's Brain. Reprinted from Leelawong and Biswas (2008), with permission of IOS Press.

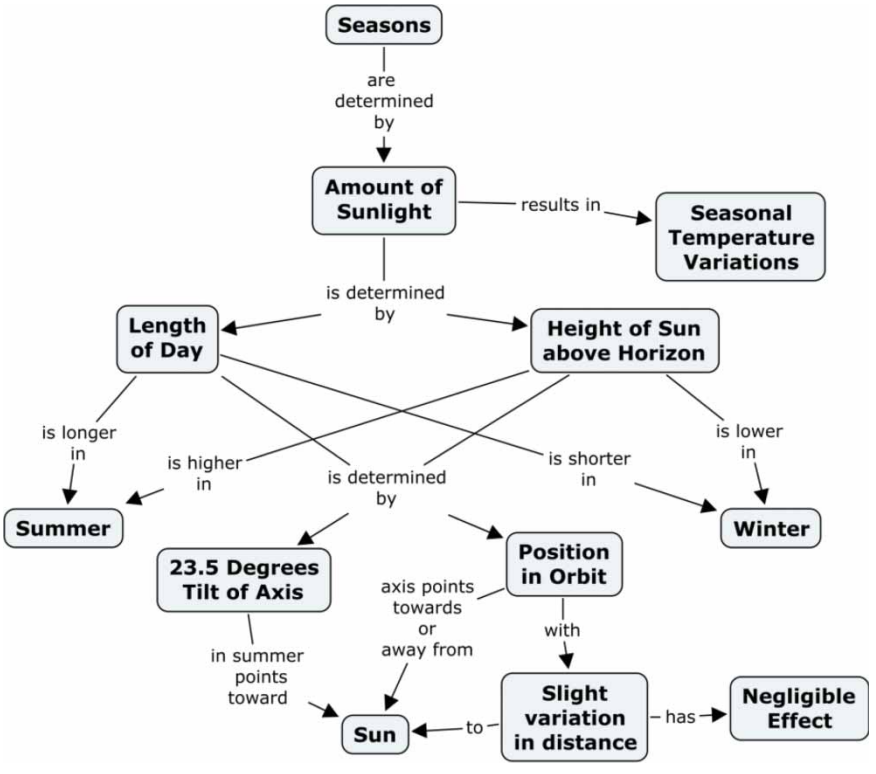

Figure 3. A concept map. Reprinted from Novak and Canas (2008), with permission of the authors.

- IQON and VISQ (Miller et al., 1993) and
- Vmodel (Forbus, Carney, Sherin, & Ureel II, 2005).

Figure 2 shows a qualitative diagram drawn by a user of Betty's Brain (Schwartz et al. 2009). The ++ label indicates that the two variables are directly related (as one increases, so does the other), while a – – label indicates that they are inversely related (as one increases, the other decreases). In Figure 2, the user is asking what happens to one variable when another variables' value is changed. This is one way to access the predictions of the model. Another is to have the software report all the variables whose values are increased or decreased when a certain variable's value is modified. Some qualitative diagram languages have other labels than + and −, and can do other types of predictions as well.

Qualitative diagrams should not be confused with concept maps even though they look similar. A typical concept map language allows the user to enter anything on the links, as shown in Figure 3. A concept map is essentially a collection of propositions such as "Seasons are determined by amount of sunlight". A concept map is a model, but it is not an executable model. That is, there is no way for the concept map itself to answer questions such as "what would happen if the tilt of the earth's axis was zero?" Activities where student construct a concept map are excluded from this review.

4.2 *System-dynamics modeling languages*

A system-dynamics model predicts the behavior of a system over time. A set of temporal differential equations can be used as a system-dynamics model. However, for students who

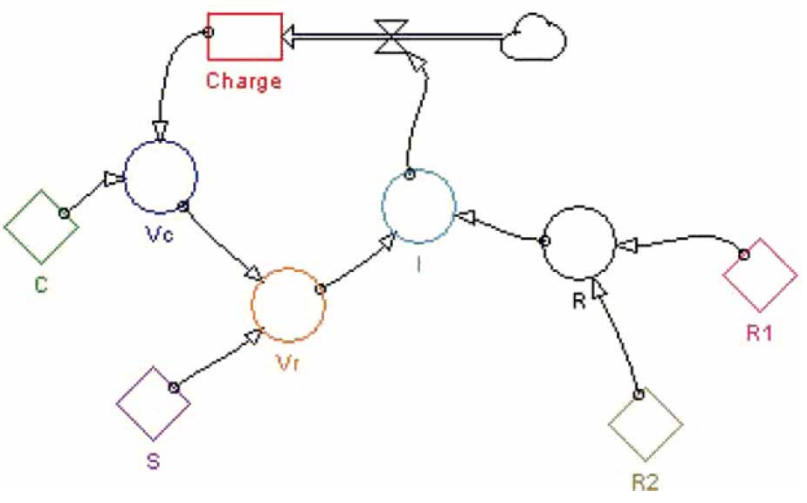

Figure 4. A system-dynamics diagram. Working counterclockwise from the top, the equations inside each rectangle and circle are as follows: $Charge_t = Charge_{t-1}+I_t$; $Vc_t = Charge_t/C$; $Vf_t = S\text{-}Vc_t$; $I_t = Vf_t/R$; $R = 1/(1/R1 + 1/R2)$. Reprinted from Mulder et al. (2010), with permission of Taylor & Francis Ltd.

are not familiar with calculus, a common system-dynamics modeling language is a quantitative diagram. A quantitative diagram summarizes a system of equations as a node-link diagram. Typically, each node represents both a variable and a function for computing its value. The inputs to the function are represented as incoming links. To avoid cluttering the diagram, the functions can be seen only by clicking on the node. Figure 4 shows an example from Co-Lab (Mulder, Lazonder, & de Jong, 2010). It represents time discretely rather than continuously as differential equations would because that makes the mathematics accessible to students who have not taken calculus. There are several general purpose, industrial strength tools for drawing such quantitative diagrams, including Stella (http://www.iseesystems.com/), Powersim (http://www.powersim.com/), Dynasys (http://www.hupfeld-software.de/pmwiki/pmwiki.php) and Vensim (http://www.vensim.com/). Tools intended primarily for education include:

- Modellingspace (Avouris, Margaritis, Komis, Saez, & Melendez, 2003),
- CoLab (van Joolingen, De Jong, Lazonder, Savelsbergh, & Manlove, 2005) and
- Model-It (Metcalf, Krajcik, & Soloway, 2000).

Although diagrams work well for small models, it can be difficult to follow links when the diagrams become more complex. However, system-dynamics models can be written as text, and this may make them easier to understand when there are a large number of variables. The caption of Figure 4 shows how its diagram can be expressed as a set of equations. Modellus is modeling tool based on equations as the model representation (Teodoro & Neves, 2011). Qualitative diagrams can also be expressed as a set of qualitative constraints expressed as text.

4.3 *Agent-based modeling languages*

An agent-based model is for modeling the emergent behavior of systems. The model consists of a large number of agents. Each agent is simultaneously executing a simple program.

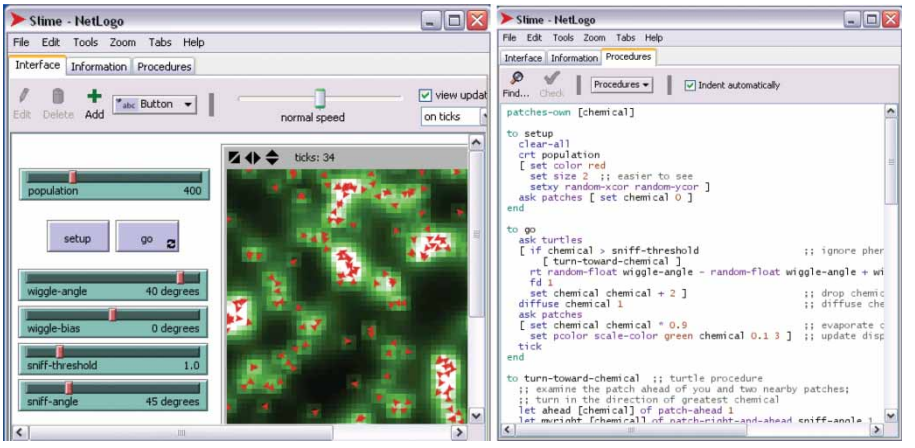

Figure 5. A NetLogo model of slime mold growth.

The program is expressed in a computer programming language, such as Logo or Java. Some modeling languages are:

- NetLogo (http://ccl.northwestern.edu/netlogo/)
- AgentSheets (Repenning, Ioannidou, & Zola, 2000)
- Worldmaker (Boohan, 1995).
- OOTLS (Neumann, Feurzeig, & Garik, 1999)
- CTiM (Basu et al., 2012).

Different types of agents run different programs. For instance, Figure 5 shows a NetLogo model of slime molds. The left panel shows the interface, which has some buttons and sliders for controlling values inside the program and an animation of the agents. The right panel shows the code for the agents' programs. Some agents represent slime molds, and they run one program. Other agents represent patches that the slime molds move across and deposit chemicals on; the ground agents run a different program from the slime mold agents. At this writing, NetLogo is the most widely used agent-based modeling language for education.

Agent-based modeling overlaps with system-dynamics modeling in that both languages can be used to model dynamic systems. For instance, a predator-prey ecosystem can be easily modeled in both NetLogo and Stella. One study (Thompson & Reimann, 2010) has compared these two modeling paradigms, but the small sample size and relatively short period of the instruction prevented drawing any conclusions about their relative effectiveness.

5. Methods for presenting systems

Although the modeling language has a major impact on the students' thinking and learning, the method of presenting the system to be modeled is also important. Here are five major ways to present systems to the students:

- *Succinct text*: The system is described in text that succinctly mentions most of the relevant entities and relationships, and seldom mentions irrelevant ones. For instance,

one might say, "In 2010, the population of Phoenix was 1,445,632 and was increasing at 9.4% per year. Construct a model that predicts the population from 2010 to 2040."

- *Resources*: The student is given a task, such as "Predict the population of Phoenix in from 2012 to 2040", and a set of resources comprising text, tables, charts, pictures, videos, diagrams, etc. The resources contain the relevant information about the system along with irrelevant information. Examples of this mode of presentation are the studies of Betty's Brain (Biswas, Jeong, Kinnebrew, Sulcer, & Roscoe, 2010; Leelawong & Biswas, 2008; Schwartz, Blair, Biswas, Leelawong, & Davis, 2008; Schwartz et al., 2009).
- *Simulation*: The student is given a simulation without being able to view the computation that drives it. In most cases, the student can control the simulation. For instance, the student could be given only the interface to the slime mold simulation, shown on the left of Figure 5 which can be controlled with the buttons and sliders. Examples of this method of presentation are studies using CoLab (Basu et al., 2012; Bravo, van Joolingen, & de Jong, 2009; van Joolingen et al., 2005; Löhner, Van Joolingen, & Savelsbergh, 2003; Löhner, Van Joolingen, Savelsbergh, & Van Hout-Wolters, 2005).
- *Virtual labs and virtual field studies*: The student is given a simulation of a laboratory with appropriate apparatus and supplies or is allowed to explore a virtual world using appropriate instruments. The students' task is to set up experiments and run them. For instance, to study slime mold growth, the student would have to manipulate simulated Petri dishes, pipettes and photographic measurement tools. Although such virtual laboratories exists (Yaron, Karabinos, Lange, Greeno, & Leinhardt, 2010), it seems that they have not yet been used as a presentation method for model-construction activities.
- *Real labs and real field studies*: Although the model is constructed on a computer, the system is presented in reality, such as visiting a stream and taking water samples, then measuring pH with the litmus paper. Examples of this method of presentation are (Metcalf et al., 2000).

For every choice of presentation type and every choice of modeling language, there is a feasible model-construction activity. For instance, if we choose "simulation" as the method for presenting the system and "system dynamics" as the modeling language, then we have a learning activity where students can manipulate the simulation in order to understand how it works, then construct a system-dynamics diagram. The student's goal is to create a model whose predictions match the behavior of the simulation. This is the activity most frequently done with Co-Lab (van Joolingen et al., 2005).

Mathematical "word problems" correspond to activities where the presentation of the system is succinct text and the modeling language is systems of equations. However, these would be word problems done with a computer system where students enter the equations into a computer (including calculators) and the computer solves them. Many students find numerical answers for word problems using informal strategies that do not involve writing equations (Koedinger, Alibali, & Nathan, 2008; Koedinger & Nathan, 2004), so they are not doing model construction.

6. Instructional objectives and assessments

Many claims have been made about the instructional benefits of model construction, and several types of assessment have been used for testing these claims. The subsections that

follow each discuss a common instructional objective and its assessment. A final section briefly mentions other benefits of model construction that are not easily tested with assessments.

6.1 *Improved model-construction skill*

Model-construction skill is operational knowledge about the modeling language, the system presentation and the method for getting from the presentation to the model. This is a kind of "sense making" (Quintana et al., 2004) or "scientific reasoning". For instance, if the presentation method involves experimentation, then the control of variable strategy ("vary one thing at a time") is part of model-construction skill. If the experimental observations have a chance of error, then model-construction skill includes knowing that one should do repeated tests or samples. If the modeling language is systems of equations, then model-construction skill includes knowing that when the number of equations is the same as the number variables, then the solution may be a single point. If a qualitative diagram has a multi-link path between two variables, then model-construction skill should include knowing how to determine from the + and – labels on the links whether the variables are directly or inversely related.

The most common procedure for assessing model-construction skill uses the same system presentation method, the same modeling language and different domain knowledge and different systems from the ones used during instruction. For instance, several studies of Betty's Brain (Biswas, Leelawong, Schwartz, & Vye, 2005; Biswas et al., 2010; Leelawong & Biswas, 2008) trained students by having them create models of the oxygen cycle in an water-based ecosystem and then assessed them by having them create models of the nitrogen cycle in a land-based ecosystem. In both activities, the systems were presented with resources and the modeling language was the qualitative diagram language shown in Figure 2.

A less common assessment of model-construction skill involves varying the modeling language. For instance, van Borkulo, van Joolingen, Savelsbergh, and de Jong (2012) trained students in using the quantitative modeling language shown in Figure 4, then tested them using a qualitative modeling language similar to the one shown in Figure 2. This could be a considered a far transfer test of model-construction skill.

Two dependent measures are commonly used:

- *Product*: One measures the quality of the models that the students' construct, typically by scoring the number of elements of the model that are correct or acceptable.
- *Process*: One measures the quality of the students' behavior when constructing a model, typically by coding videos or log data for events that are clearly unacceptable, such as starting to construct a model before paying any attention to the presentation of the system, or running a deterministic modeling language twice without changing the model between runs.

6.2 *Improved domain knowledge*

Domain knowledge refers to general concepts and principles that apply more widely than just the system being addressed by the model-construction activity. Model-construction activities can impact this knowledge in three ways: by adding new domain knowledge, by making existing knowledge more operational and robust, and by removing misconceptions. Let us consider each instructional objective in turn.

One goal of inquiry learning is that students develop hypotheses about a particular system and that the hypotheses turn out to be general domain principles. For instance, the Thinker Tools sequence of model-exploration activities led some sixth graders to discover general principles governing velocity, acceleration, mass and force (White, 1993). Such an instructional objective can be measured with a conventional assessment, such as asking the students to state their newly won beliefs about the domain or asking them questions about them.

Another instructional objective is for students to convert their inert, fragile prior domain knowledge into operational, robust domain knowledge. That is, instead of discovering new hypotheses, students are told the relevant scientific knowledge *before* or *during* the model-construction activities. For instance, the system can be presented in the context of a set of resources that present the target domain knowledge, as in the Betty's Brain studies (Chin et al., 2010). The appropriate assessments are to see if students can apply the domain concepts and principles in near and far transfer situations. Retention tests may also be used to measure robustness.

Students often have incorrect beliefs about the physical world, which are often called misconceptions or alternative conceptions. Misconceptions often can be detected during interviews with students or using multiple-choice tests whose foils offer responses consistent with the false beliefs. Using both methods, Lee, Jonassen, and Teo (2011) showed that having students construct models in Model-It reduced their misconceptions compared to a control group of students who received a more standard inquiry instruction over the same period. Their task domain was the water cycle, and observed misconceptions include (p. 54): "When the plant gives out carbon dioxide or oxygen, they would condense on the transparent plastic bag and water droplets formed."

6.3 *Improved understanding of a particular system*

Some systems are so important that it is worth the student's time to come to a deep understanding of them. For instance, the effect of greenhouse gases on global warming may be worth understanding in detail.

Whereas model-construction skill and domain knowledge are assessed by using different systems during the assessment from those used during the training, if the goal is to deeply understand a particular system, then that system must be used in both training and assessment phases. Because the system is the same in both training and assessment, it would be wise to use a different task for assessment than for training. If the assessment had students construct a model of the target system (e.g. global warming), and they had already constructed such a model during training, then the assessment task might tap only shallow, episodic memories rather than a deep understanding of the target system. Thus, this instructional objective might be best assessed using tasks that are not model construction but nonetheless tap the same deep understanding of the system as model construction. Table 3 lists candidate assessment tasks culled from a classic edited volume on mental models (Gentner & Stevens, 1983) and from a review of assessments used in the system-dynamics literature (Hopper & Stave, 2008).

Developing scoring methods for such assessments requires considering the *use-specificity* of transfer (Singley & Anderson, 1989). The basic idea is that if an assessment task requires, say, 100 knowledge components and the model-construction task involves only 60 of them, then the expected best score on the assessment task would be only 60%, assuming that the score is linearly related to the number of knowledge components that have been mastered. Determining the number of shared knowledge components often requires a

Table 3. Methods for assessing students' understanding of a system.

Assessments involving prediction.
° Qualitative relationships. Ask the student questions of the form, "If X increases, then will Y increase, decrease or be unaffected?"
° Sketching graphs. Ask the student to sketch the relationship between two variables, or the value of a variable over time.
° Classifying behavior. Ask the student whether the system, or a variable of the system, exhibits oscillation, damped oscillation, asymptotic growth, unbounded growth, emergence, or some other class of behavior that has a name known to the student.
Assessments involving faults.
° Response to a fault. When a fault is inserted into the system, how will that affect its behavior?
√ Recognizing faulty behavior. When given a system behavior or given a system whose behavior can be tested, can the student determine if the system is operating normally or whether it has a fault?
√ Troubleshooting. When given a system whose behavior is faulty, can students find and correct the fault?
Assessments involving redesign.
√ Extensions. Can the student modify the system to add a new capability or behavior?
° Optimization. Can the student modify the system to exhibit the same behavior while reducing the cost, error rate, complexity, latency or some other valued measure?
Assessments involving exposition.
° Explanation. How well can the student explain or describe the system's behavior?
° Teaching. How well can the student teach the system's behavior to another student?
° Analogies. How well can the student construct or judge analogies to the system?
Assessments involving operating a device.
° Plan a sequence of operations to minimize time or risk of error.
° Choose which of several methods for operating the device is best for a particular situation.
° Control the output of a device despite fluctuations in its inputs.
° Design or optimize a general procedure for operating the device.
° Estimate the time or number of operations required to operate the device.
° Non-standard operation. When the normal operating procedures are blocked or fail, invent a sequence of operations that will work.
Retention.
° In addition to performing one of the assessments above immediately after the students' have engaged in a model construction activity, the assessment is conducted several days or weeks later.

separate study. For instance, consider a study of the transfer between tasks (Kessler, 1988): (1) constructing a computer program, (2) debugging a given computer program and (3) calculating the output of a given program with a given input by mentally executing it. These three activities are rather similar to constructing a model, debugging a model and making predictions using a model. Kessler thoroughly trained students on one of the three tasks, and then tested them on all three tasks. He found large amounts of transfer between constructing and debugging a program, but little transfer between mentally executing a program and the other two tasks. This makes sense given that constructing and debugging programs seem intuitively to share many steps, whereas mentally simulating the execution of a program hardly shares any steps with the other two. If the analogy between program construction and model construction holds, then use-specificity of transfer predicts little transfer from model construction to mental prediction of system behaviors, which is the first set of tasks in Table 3.

Although predicting transfer accurately requires a detailed cognitive task analyses and studies of both the model-construction activity and a given assessment task, Table 3 has "√" bullets next to the assessments that seem most likely to overlap with model construction. These may be the most appropriate assessments if the goal is to determine which instructional treatment causes the best understanding of the system.

6.4 *Concept inventories and mental modeling*

Concept inventories are non-quantitative assessments that are often used in quantitative science and mathematics courses (Pfundt & Duit, 1998). They often have questions that require students to construct a mental model, execute it, and thus predict the behavior of a system. For instance, here is a question from the famous Force Concept Inventory (FCI) (Hestenes, Wells, & Swackhamer, 1992):

> Two metal balls are the same size, but one weighs twice as much as the other. The balls are dropped from the top of a two story building at the same instant of time. The time it takes the balls to reach the ground below will be:
>
> (A) About half as long for the heavier ball.
> (B) About half as long for the lighter ball.
> (C) About the same time for both balls.
> (D) Considerably less for the heaver ball, but not necessarily half as long.
> (E) Considerably less for the lighter ball, but not necessarily half as long.

To answer such questions, students should construct a mentally held model of the system and then execute it in order to predict its behavior (Ploetzner & VanLehn, 1997). As another example, Figure 6 shows a question from the concept inventory of Booth Sweeney and Sterman (2000). It also requires students to construct a mentally held model of the system and execute it in order to predict the system's behavior. Like many concept inventory tasks, students sketch their answer rather than selecting it from a set of choices.

Because correctly answering a concept inventory question requires both constructing a mental model and mentally executing it, one must be careful in interpreting scores. When students score well on a concept inventory, then they probably have mastered both skills. However, lack of mastery of just one skill suffices for generating low scores. In particular, even if the instruction is completely successful at teaching model-construction skills and yet provides no practice at mentally executing models, then scores on the concept inventories will probably be low.

To make this issue more concrete, let us consider a specific analysis of transfer from instruction based on model construction to assessment based on the FCI. Ploetzer and VanLehn (1997) constructed rule-based cognitive models of both quantitative and qualitative physics problem solving. The rules for quantitative problem solving were validated against verbal protocols of students studying examples and solving ordinary quantitative physics homework exercises; they represented what students could reasonably learn from such instruction. The rules for qualitative problem solving were constructed to both answer the FCI questions and to be as similar as possible to the rules for quantitative problem solving. If correctly answering FCI questions required only skill at constructing models, then one would expect 100% of the FCI rules to correspond to quantitative problem solving rules, which include rules for constructing equation-based (constraint system) models. In fact, only 36% of the FCI rules corresponded to quantitative problem solving rules. Of the FCI rules that did not match quantitative problem solving rules, most involved deriving predictions from models, which is the mental equivalent of executing the model. This analysis suggests that success on the FCI requires both skills (mental construction and execution of models) but that only one of these skills is taught in conventional physics instruction.

Some educators consider mental modeling, and the correct answering of concept inventory questions in particular, to be an essential instructional objective, perhaps even more important than formal model construction. In such classes, the instructors should probably

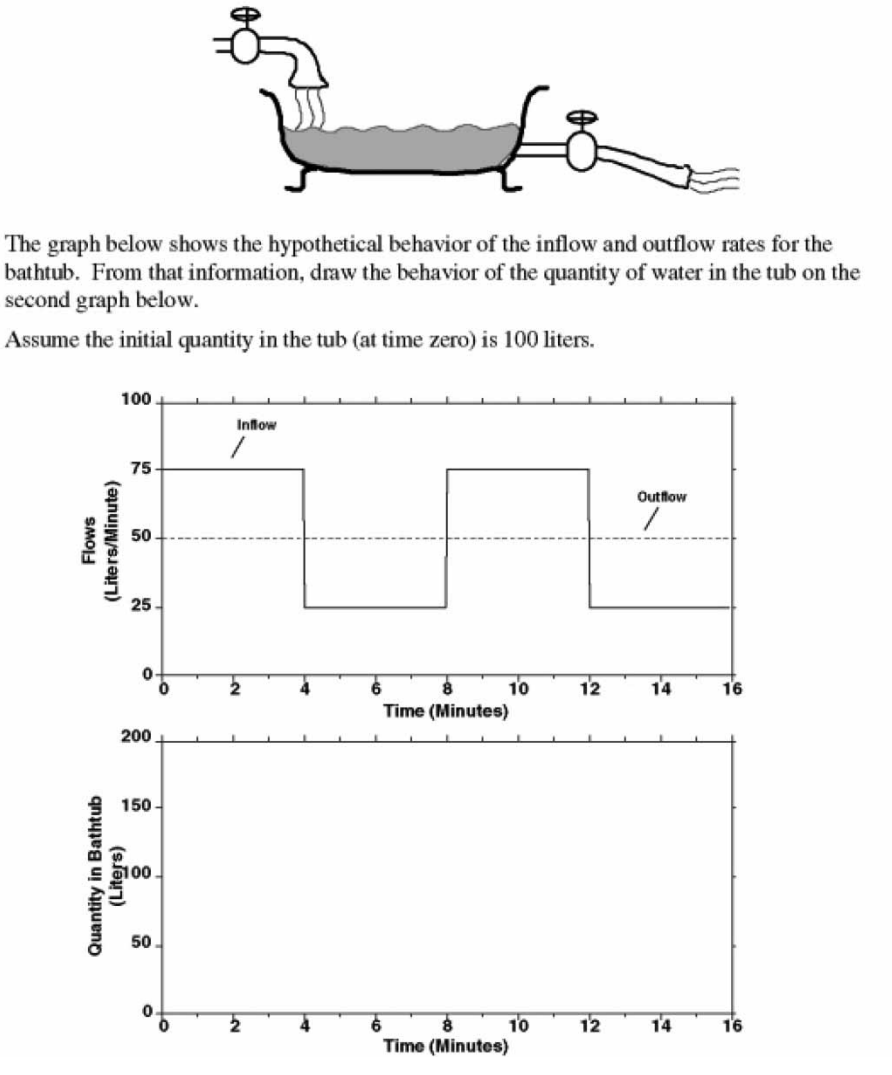

Figure 6. A question from a system-dynamics concept inventory. Reprinted from Booth Sweeney and Sterman (2000), with permission of John Wiley and Sons.

give the students considerable practice in mentally constructing and executing models using problems similar to those on the concept inventories. VanLehn and van de Sande (2009) sketch such a curriculum for physics. Löhner (2005, chapter 5) presents preliminary evidence about the effects of adding model execution exercises to a model-construction learning activity.

6.5 *Understanding the role of models*

Model-construction activities allow students to learn about the role of models in science, business, government and other domains. For instance, students can learn:

- to distinguish the model's structure from its predictions about the system's behavior;
- that validating a model means testing that its predictions match known, well-understood system behavior;
- that models can explain poorly understood systems by matching the observed behavior and offering predictions about unobserved behavior;
- that models can make predictions about the behavior of systems that have not yet been observed or even built;
- that models can help decision-makers by allowing them to compare the costs and benefits of alternatives by modeling each;
- that models can be valid with respect to known observations but nonetheless make incorrect predictions about future observations;
- that models can be modular so that even the large, complex models used for, e.g. weather, economics and nuclear explosions can be understood in terms of submodels which are more easily understood and validated and
- about parsimony, sensitivity, calibration, non-identifiability and many other epistemological concepts.

Such epistemological knowledge is declarative knowledge, so it can be assessed with questionnaires, interviews and journals (Crawford & Cullin, 2004; Schwarz & White, 2005; Treagust, Chittleborough, & Mamiala, 2002). Schwarz et al. (2009) define two learning progressions for modeling epistemology and show how students' drawings can be used to assess their location along each progression.

6.6 *Other less easily assessed instructional objectives*

There have been many claims about the benefits of model construction, including some that would be rather difficult to assess. This section lists a few.

- When students construct an executable model, they have constructed a machine that thinks. Moreover, they have told it exactly how it should think. The mere fact that they can do this is a powerful idea (Papert, 1980).
- Model construction may lead students to take ownership of their ideas and knowledge (Penner, 2001).
- The models can become an object of discussion and communication among students and thus help them practice evidence-based discussion skills (Schwarz & White, 2005; White, 1993).
- Model construction may help students see the "big picture" and overarching patterns (Mandinach & Cline, 1994).
- Students may more clearly distinguish observed phenomena from underlying causal processes and make stronger connections between them (Doerr, 1996).
- Students may develop intuitions, predilections and skills at understanding complex phenomena in general (Hogan & Thomas, 2001; Mandinach & Cline, 1994; Schecker, 1993; Steed, 1992).

7. Common student difficulties

Students have many difficulties with model construction, and this section lists some of the most commonly mentioned ones. They are grouped, somewhat arbitrarily, into difficulties

with the overall problem solving method, difficulties with the modeling language and difficulties understanding the presentation of the system.

7.1 *Poor problem solving strategies*

The cognitive process that students should ideally do is diagrammed in Figure 1. It is included in the Common Core State Standards in part because there is wide agreement that this build-test cycle is the ideal way to construct a model.

More specifically, students should build an initial version of the model while being sure to include the quantities or behaviors that they want to explain. They should then test the model by executing it and comparing its predictions to the behavior of the system, in so far as they know what that behavior is. If the model's predictions do match the behavior of the system, then students should test different cases or starting values in order to insure that a match is obtained over a variety of test cases. If they find a mismatch between the models' predictions and the system's behavior, then they need to debug the model by locating and fixing the part of the model responsible for the incorrect prediction. They should then resume evaluating the model.

When a system is complicated enough, students should decompose the system into parts and perform the build-test cycle described above on one part after another. Before beginning, students should identify their goal such as modeling a particular behavior, quantity or relationship. This goal should drive their decomposition of the system. As they iteratively develop the model, they should reflect often on their goal in order to prevent building unnecessary parts of the model.

Unfortunately, many students do not enact the cognitive processes described above. This section lists common ways that student's fail.

The goal of model construction is to construct a parsimonious model in the given modeling language such that the model's predictions match the behavior of the given system. However, students often appear to have a different goal in mind, such as:

- to produce a single, numerical answer (Booth, 1988; Hogan & Thomas, 2001; Miller et al., 1993). This misconception may have been encouraged by early experiences with algebra story problems;
- to construct a model that displays everything they believe to be true about the system, without any regard to what predictions the model makes (Hogan & Thomas, 2001). In fact, they may never execute the model. This misconception may be encouraged by early experience with concepts maps and
- to get the predictions of the model to match the behavior of the system without considering the domain or plausibility. As Löhner et al. (2005, p. 456) put it, "they spend their effort trying to match their model output to the system simulation output, rather than trying to explain the underlying mechanism."

Even when students understand that the goal of model construction is to create a model that explains the behavior of a system, they often fail to adequately test and debug their model. Some noted examples of poor testing and debugging are:

- When students execute their model, they sometimes do not compare its predictions to the system's behavior (Metcalf et al., 2000). Thus, they do not notice when the model is making incorrect predictions.
- Students sometimes test their models on only one or two cases (Alessi, 2000a).

- To fix incorrect predictions, students sometimes add "fudge factors", which are formulas, constants or logical conditions designed to artificially fix the problem, and not to model the system (Alessi, 2000a).
- Students sometimes make unmotivated changes to the model in the vague hope that the results will come out right (Alessi, 2000a; Löhner et al., 2005).

As mentioned earlier, when a system is complicated enough, students should use a decompositional method. That is, in order to create a model of a whole system, students should divide the system into loosely coupled parts, build *and test* a model of each of the parts, then integrate the parts' models into a whole. Unfortunately, students often create a large model before testing it (Metcalf et al., 2000). They then have trouble finding out why it makes poor predictions.

Students often copy models from the instructional materials, perhaps thinking that it will be easier to modify an existing model (which, unfortunately, they may not fully understand) rather than start from scratch (Alessi, 2000a). Students often try to incorporate the formulas of previous science and mathematics classes (which they often do not fully understand) instead of doing true system analysis (Alessi, 2000a).

7.2 *Difficulties understanding the modeling language*

This section lists a few of the observed difficulties students have in understanding the language in which models are constructed, and hence the models that they have constructed. Although the language may be formal mathematical expressions or formal diagrams, it is still a "foreign" language and quite non-trivial for students to learn (Koedinger et al., 2008).

Students often lose track of what the symbols in their models denote. For instance, Paige and Simon (1966) found that algebra students would report a negative number for L, where L should denote the length of a board. If students had kept track of the denotation of L, they would have realized immediately that a negative length is impossible. When building a model and especially when testing a model, it is essential that students keep track of what the model's symbols denote.

When students are constructing a model that is intended to be understood by someone else, there is an additional problem, which is called "grounding". The term comes from the literature on analyses of natural language dialogues, and it denotes the psycholinguistic process where two people come to a mutual understanding of what a phrase or term means (Clark & Brennan, 1991). Often this process proceeds smoothly and invisibly, but it occasionally breaks down. For instance, the term "next week" often denotes different weeks for different participants in a conversation. Similarly, a variable labeled "birth rate" in a model can denote different quantities to the student and the teacher. The problem is exacerbated when the student is trying to communicate with a computer tutor, which is handicapped by lack of linguistic skill. If help is offered by a teacher, computer tutor or peer, and the participants have failed to ground all the formal terms in the model, then the help can be extremely confusing.

Students often get confused when modeling languages use symbols for both objects and quantitative properties of objects. When students are given a list of terms such as "water", "tanks", "water pressure" and "water level in a tank", they have trouble identifying which terms are variables (Kurtz dos Santos & Ogborn, 1994).

Students are often confused when the modeling language allows a variable's value to be specified using a differential. Most system dynamic models have two kinds of relationships between variables. One can be expressed by an equation of the form $x = f(V1 \ldots Vn)$, where

$V1 \ldots Vn$ are variables not including x, and f is a mathematical function not including derivatives or integrals. The other kind of relationship is expressed by an equation of the form $dy/dt = \Sigma \pm v\_i$ which says that the change in y over time is a weighted sum of variables, where the weights are either +1 or −1. In graphical languages for system dynamic models, the variable y is called a "stock" and the variables v_i are called "flows". Students often have difficulties understanding stocks and flows (Booth Sweeney & Sterman, 2000; Cronin, Gonzalez, & Sterman, 2009; Hopper & Stave, 2008; Kainz & Ossimitz, 2002; Metcalf et al., 2000; Sterman & Booth Sweeney, 2002). Kurtz dos Santos and Ogborn (1994) suggest that flows that have constant values are much less confusing than flows whose values vary over time.

Students who are learning algebra sometimes have difficulty in understanding the formal language. As a vivid illustration, Booth (1988) listed several misconceptions about algebra notation found during interviewing students in the 8th to 10th grades:

- Thinking that implicit multiplication is concatenation, so when y is 4, 5y is 54.
- Thinking that coefficient-variable terms, such as 5 m, act like number-unit terms. Thus, 5 m means "5 meters," so "5 y" would denote 5 yogurts or 5 yams or a set of 5 other objects or units whose first letter is "y".
- Thinking that the equals sign is a command to produce an answer.
- Thinking that different variables must stand for different numbers. Thus, $a + b = a + c$ can never be true because b and c must stand for different numbers.

It is likely that other formal modeling languages, even the graphical ones, also present difficulties and afford misconceptions. For instance, after a 3-hour pilot experiment in the author's lab where students used a stock-and-flow language similar to the one of Figure 4, students reported during focus groups that they did not understand the difference between the single-line arrow and the double-line arrow. Löhner et al. (2003) found that students had trouble entering equations that they knew about into a graphical modeling language. Heffernan and Koedinger (1997) found that students could solve parts of algebra word problems but could not compose the results algebraically, suggesting a lack of fluency in the language of algebra.

7.3 *Difficulties understanding the presentation*

As mentioned earlier, there are a variety of methods that instructors can employ for presenting a system to the student. This section just considers the two most widely used methods: experimentation with simulations and reading text.

When the presentation of the system requires experimentation in order to uncover its behavior, there are a variety of obstacles associated with inquiry. For instance, students' experiments may be either motivated or unmotivated. A motivated experiment is done in order to test a specific hypothesis, whereas an unmotivated experiment is sometimes characterized as guesswork. For instance, in one model-construction study, 39% of the students' experiments were designed with no hypothesis in mind (Löhner et al., 2005). Low domain knowledge is often associated with frequent unmotivated experiments (Lazonder, Hagemans, & de Jong, 2010; Mulder et al., 2010). This is just one example of the many difficulties that students have when attempting to understand a system by experimenting with it.

When the presentation of a system is text, then superficial reading is a common problem. For instance, keywords such as "per" or "altogether" can often be used to help translate a word problem into an equation (Paige & Simon, 1966). By the time, students

have reached high school and are solving algebra story problems, the use of such superficial reading strategies may have diminished. Koedinger et al. (2008) found that accuracy for story problems that used operator words (e.g. "subtracted" and "multiplied") was not higher than for story problems that used other words instead. Heffernan and Koedinger (1997) found that simplifying the language made little difference on the error rate of algebra word problems.

Nonetheless, there are certain phrases in English that invite superficial reading. For instance, Clement (1982) found that university engineering students usually translated "there are six times as many students as professors in this university" into 6S = P. This misconception was dislodged neither by hints nor by using diagrams.

System-dynamics problems often involve quantities that are ratios and that vary over time, and the English phrases that express such concepts are often awkward and difficult to understand. For instance, a variable named "rabbit birth rate" might be defined as either "the number of rabbits born per year per rabbit in the population" or "the ratio of baby rabbits born in a year to the number of rabbits in the population that year." Faced with such convoluted language, many students may opt for a superficial reading assuming that it will be easier to debug the resulting model than to try to untangle the language.

The remarks made so far apply to all text, including presentations comprising a short paragraph of text that presents all and only the information needed for constructing the model. Another method of presentation is to give students a set of resources (e.g. multimedia; a list of web pages) with the relevant information about the system embedded in considerable irrelevant information. Ideally, students would alternate between constructing the model, discovering that they needed to know something about the system, searching the resources for the desired information, and resuming construction of the model. Although fast learners using Betty's Brain did exhibit that pattern of reading, slow learners spent large amounts of time reading the resources in an unfocused fashion (Segedy, Kinnebrew, & Biswas, 2012).

8. Types of scaffolding

As the preceding section illustrated, model construction is difficult, so considerable research has investigated methods for accelerating students' learning. These methods for helping learners along are often called "scaffolding", because they can be removed when the learner has reached mastery. That is, scaffolding is not an intrinsic part of the model-construction activity, but is something added to the activity.

Because many creative forms of scaffolding have been invented, it is difficult to organize this section. The list of scaffolding types has been grouped into (1) those that require tutoring or similar interactivity from the system, (2) those that involve clarification of the modeling language, (3) those that involve gradually increasing the complexity of the model-construction activities, and (4) all the rest.

8.1 *Tutoring*

A typical computer tutor knows the solution but gives the students only prompts, hints and feedback about it. This kind of scaffolding is covered in the first two subsections below. The remaining subsections cover other interactive methods of scaffolding, such as answering questions asked by students. The scaffolding methods described in this section often require artificial intelligence or other sophisticated computation.

8.1.1 *Feedback and hints on the student's model*

Several systems give feedback and hints to students on the model. Bravo et al. (2009) extended CoLab to do such tutoring. When the student clicked on a "Check" button, the system compared the student's model to a reference model, noted the differences between them, selected one, and told the student about it. A formative evaluation of the system was done, but the system was not compared to the standard version of CoLab, which lacks such tutoring.

In an early version of Betty's Brain, students could ask Mr Davis (an agent portraying a tutor) for help after Betty took a quiz, and he would give hints (Biswas et al., 2005). If students asked for enough hints, Mr Davis would eventually reveal exactly how to change the model. Unfortunately, students tended to ask for hints rapidly, ignoring them until Mr Davis indicated how to fix the model. This form of help abuse, sometimes called *gaming the system*, is quite common in other tutoring systems as well (Baker, Corbett, Koedinger, & Wagner, 2004; Muldner, Burleson, van de Sande, & VanLehn, 2011). This led Biswas et al. to modify Mr Davis so that he gave hints on the *process* of model construction rather than the *product*, i.e. the student's model. That form of scaffolding is covered next.

8.1.2 *Feedback and hints on the student's process (meta-tutoring)*

Whereas the preceding section discussed feedback and hints on the students' model as they are constructing it, this section discusses feedback and hints on the students' behavior. Feedback on the model often has to mention domain knowledge. For instance, if students have constructed a model that says that all rainfall becomes runoff, then the system might say, "Does any water soak into the ground? Is runoff really equal to rainfall?" On the other hand, feedback and hints about the student's behavior seldom need to mention domain knowledge. For instance, if students continue to add to their model without testing it, then a possible hint is, "It's a good idea to run your model and check its predictions even before it's finished." Because domain knowledge is not mentioned, feedback and hints on process are sometimes called "meta-tutoring" because "tutoring" is generally reserved for domain-specific feedback and hints.

Betty's Brain was equipped to do both tutoring and meta-tutoring (Leelawong & Biswas, 2008). It could comment on the student's model or it could comment on the student's behavior. Perhaps, the most powerful example of the latter was that Betty would refuse to take a quiz until the student had tested her first. That is, students had to ask specific questions of their model (e.g. if air temperature goes down, what does body temperature do?) before the model could be given a test suite of instructor-generated questions. In a series of studies (Leelawong & Biswas, 2008; Tan, Biswas, & Schwartz, 2006; Tan, Wagster, Wu, & Biswas, 2007; Wagster, Tan, Biswas, & Schwartz, 2007; Wagster, Tan, Wu, Biswas, & Schwartz, 2007), meta-tutoring was found to be significantly more effective than the same system with meta-tutoring turned off.

Although Betty's Brain taught students a strategy for model construction, it did so in a piecemeal fashion as hints from Mr Davis or comments from Betty. Other model-construction systems offered more explicit scaffolding. For instance, the final version of Model-It (also called TheoryBuilder (Metcalf, 1999)) had several features intended to guide students' process of constructing a model (Metcalf et al., 2000). The model editor had four modes, which were selected by clicking on one of four buttons labeled Plan, Build, Test and Evaluate. The Build mode was the actual model editor. The other modes presented forms to be filled in by the student. Figure 7 shows the form for the Test mode. Students were given feedback and hints, which they could suppress if they desired, when they attempted to

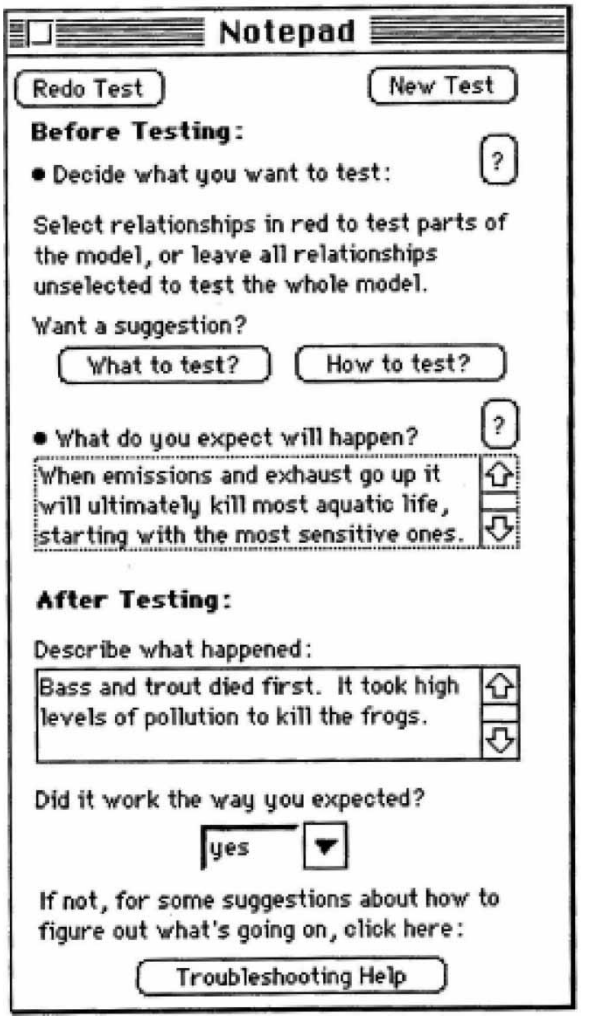

Figure 7. Scaffolding for the test mode of Model-It. Reprinted from Metcalf (1999), with permission of the author.

bypass a suggested activity. The Carnegie Learning's Algebra Cognitive Tutor (www.carnegielearning.com) and the Word Problem Solving Tutor (Wheeler & Regian, 1999) also scaffolded problem solving strategies via lightweight constraints, implemented with a small amount of didactic instruction followed by feedback and hints.

The meta-tutoring of Betty's Brain placed only weak constraints on students' behavior. The four phases of Model-It placed somewhat stronger constraints on students' behavior. At the far end of this progression is procedural scaffolding, which places very strong constraints on student behavior. The basic idea of procedural scaffolding is to teach students to temporarily follow a specific procedure for constructing a model. Although the procedure is not required by the task and there are many other ways to successfully problems, the procedure is used as a temporary scaffolding to guide students who might otherwise be quite lost. A physics tutoring system, Pyrenees (Chi & VanLehn, 2008, 2010) had students start by identifying the quantity, such as the final velocity of a falling object, that the physics

problems asked them to calculate. They were then asked to identify a physics principle, such as the definition of kinetic energy that contains it. Then they wrote the equation down. Next, for each unknown quantity in the equation, they repeated this process, treating the unknown as the sought quantity. In this fashion, they constructed a set of equations that modeled the given system. Procedural scaffolding significantly improved students' model construction in the task domain where it was required. Moreover, when procedural scaffolding students transferred to a new task domain where procedural scaffolding was not required, they again learned significantly faster than the control students.

Procedural scaffolding was used by Marshall et al. (1989) to scaffold arithmetic story problem solving, and by AMT (VanLehn et al., 2011) to scaffold construction of system-dynamics models. In all three systems, students were first given feedback and hints that kept them following the procedure, and later the feedback and hints were turned off thus allowing them to try solving problems without following the procedure. That is, the procedure was "faded out".

Whereas giving students feedback and hints on their models appears to have problems, giving them feedback and hints on the process of constructing a model has been shown to be effective in a several studies. Such "meta-tutoring" is one of a small number of scaffolding method that seems reliably useful.

8.1.3 *Concrete articulation strategy*

One method for helping students write equations with variables is to first ask them to write several versions of the equation with specific numerical values for the quantities that will eventually be variables. For instance, suppose students are asked to write an equation to model the following system: "Cathy took a *m* mile bike ride. She rode at a speed of *s* miles per hour. She stopped for a *b* hour break. Write an expression for how long the trip took." Before asking students to write an abstract algebraic expression, this scaffolding method asks students to solve several concrete, arithmetic expressions e.g. "Suppose Cathy is going at 20 miles per hour, rides 100 miles and takes a 2 hour break. Write an arithmetic expression for how long the trip took."

This scaffolding method has been used in several algebra tutoring systems (Heffernan, Koedinger, & Razzaq, 2008; Koedinger & Anderson, 1998; McArthur et al., 1989). Heffernan, who coined the name "concrete articulation strategy", found that it increased learning compared to the same tutoring system without the method. The method is used in the algebra tutors of Carnegie Learning (www.carnegielearning.com), and could probably be applied to many types of model construction as well.

8.1.4 *Decomposition into subsystems*

One form of scaffolding is to suggest a decomposition of the system so that one can focus on one subsystem while temporarily ignoring the rest of the system (Ramachandran & Stottler, 2003). For instance, consider modeling this system:

> Alan, Stan and Doug are picking cherries. Doug picks 7 gallons more cherries than Stan, and Stan picks half as many cherries at Alan. If Alan picks A cherries, how many does Doug pick?

If the student has trouble writing an equation for this, the tutoring system can suggest, "Well, suppose that Stan picks S cherries. What expression would you write using S for the number of gallons of cherries that Doug picks?" The tutoring system has decomposed the system into a Stan-Doug part and a Stan-Alan part, and asked the student to focus on just one part.

This scaffolding is used by several algebra word problem tutoring systems (Heffernan, 2001; Heffernan et al., 2008; Ramachandran & Stottler, 2003), including commercial ones (e.g. http://www.pearsonhighered.com/educator/mylabmastering/products/index.page). As typically deployed, students first attempt to provide an answer to the given problem. If they then ask for help, the tutoring system decomposes the problem into a series of subproblems which it poses to the student.

8.1.5 *Reflective debriefings*

One method of scaffolding is to wait until students have finished a model-construction activity, then ask them reflection questions such as, "What did you learn?" or "What parts of the task were confusing?" (Buckley et al., 2004). One can also ask qualitative questions such as, "What would be different if the mass were increased substantially?"

Such reflective debriefings were used in a series of studies with a physics tutoring system (Connelly & Katz, 2009; Katz, Allbritton, & Connelly, 2003; Katz, Connelly, & Wilson, 2007). Using both human tutors and computer tutors, it was found that the tutoring system's effectiveness was increased by adding reflective debriefings. This seems likely to work with any model-construction instructional system.

8.1.6 *Answering student questions during model construction*

A simple scaffolding method for human instructors to implement is to answer questions raised by students as they are constructing models. For example, Corbett and his colleagues augmented a model-construction system to answer questions typed by students in a chat window (Anthony, Corbett, Wagner, Stevens, & Koedinger, 2004; Corbett et al., 2005). If a student had not asked questions recently, the system would prompt with, "Do you want to ask a question?" Students can ask Betty's Brain to explain why a given variable's value increases, and it will present the chain of logical inferences that led to this conclusion (Leelawong & Biswas, 2008). DynaLearn both explains its reasoning and answers other questions as well (Beek et al., 2011).

Unfortunately, students tend not to ask substantive questions. Betty Brain constantly nagged students to ask for explanations because students tended not to do so (Leelawong & Biswas, 2008; Segedy et al., 2012). In one study of Corbett's system, students asked about 14 questions per hour. Most questions were about the user interface and about the correctness of answers. Students seldom asked about mathematical processes, knowledge or principles. Of the 431 questions asked during the whole study, only one was about mathematics principles. Corbett et al. concluded that a major problem with this form of scaffolding is that students do not ask many deep questions.

This is typical of students. Even when working with a human tutor (not necessarily on a modeling activity), students rarely ask substantive questions (Core, Moore, & Zinn, 2003; Graesser, Person, & Magliano, 1995). Thus, the empirical record suggests that question-answering fails as scaffolding simply because students do not ask enough questions while they are learning.

8.2 *Clarifying the modeling language*

As mentioned earlier, students have many difficulties understanding the meaning of their models. They often just push the formal symbols (e.g. nodes and links) around without

any regard to what their manipulations mean. This section describes several methods for combating this kind behavior, which Löhner et al. (2005) call "jigsaw puzzling".

8.2.1 *Notation for models*

As mentioned earlier, students often ignore the meaning of modeling language elements (e.g. variables, equations) as they try many possible ways of combining them to make the model's predictions match the system's behavior (Löhner et al., 2005). One way to scaffold model construction is to design the notation for the model so that the elements display their semantics more clearly.

Several model-construction systems supplement or replace equations with other notations. For describing a functional relationship, Model-It (Metcalf et al., 2000) had students construct a sentence (see Figure 8, top pane) or fill in a table (Figure 8, middle pane). The system would draw a graph representing the functional relationship as described by the student. For describing simple differential equations (i.e. $x' = y$), students constructed a sentence (Figure 8, lower pane) without graphical representation of the relationship. Notice that students had space for entering unconstrained text for explaining the relationship they had selected. Before writing an equation, Kid's World (McArthur et al., 1989) had students first express relationships between quantities by drawing an x-y graph or by filling in a table. CoLab (van Joolingen et al., 2005) allows students to express relationships in three forms: text, graphs and mathematics (Figure 9).

Although constraint-based models all provide notation for variables and for constraints among variables, some also provide explicit representations of *objects* and *processes*. Formally speaking, an "object" is merely a set of variables and a "process" is merely a set of constraints. They serve only to organize the model and play no role in the underlying mathematical structure, which is a system of variables and constraints. Nonetheless, objects and processes may add conceptual clarity to a model. For instance, Model-It lets the user define "stream" as an object and define properties of stream such as turbidity and nitrate_concentration. The properties are variables. Prometheus (Bridewell, Sanchez, Langley, & Billman, 2006), VModel (Forbus et al., 2005) and Homer (Bredeweg & Forbus, 2003) let the user define processes, such as "exponential growth" or "heat flow", and indicate a set of constraints that define it. Figure 10 shows a model in VModel, where the rectangular nodes represent both objects (e.g. wall, inside, outside) and processes (e.g. heat flow out), and the oval nodes represent quantities.

Because many students find it easy to construct concept graphs, some constraint-based languages provide a similar notation for expressing extra relationships among quantities, objects and processes. For instance, Betty's Brain allows students to indicate that carbon dioxide is a type of greenhouse gas (Figure 2). Constraints among quantities can also be given names, such as "releases" or "burns", which describe the relationship but play no role in the mathematics of the constraint.

Although many innovative designs for notation have been implemented, only one study has compared notations for model construction. Löhner et al. (2003, 2005) compared a graphical notation and a traditional equation-based notation. Although many differences in students' behavior were apparently caused by the difference in notation, the authors did not draw a conclusion about which notation was better overall.

So far, the discussion has focused exclusively on notation for constraint-based and system-dynamics languages. There has been comparatively little experimentation with notations for agent-based languages. The major agent-based language, NetLogo, uses a dialect of

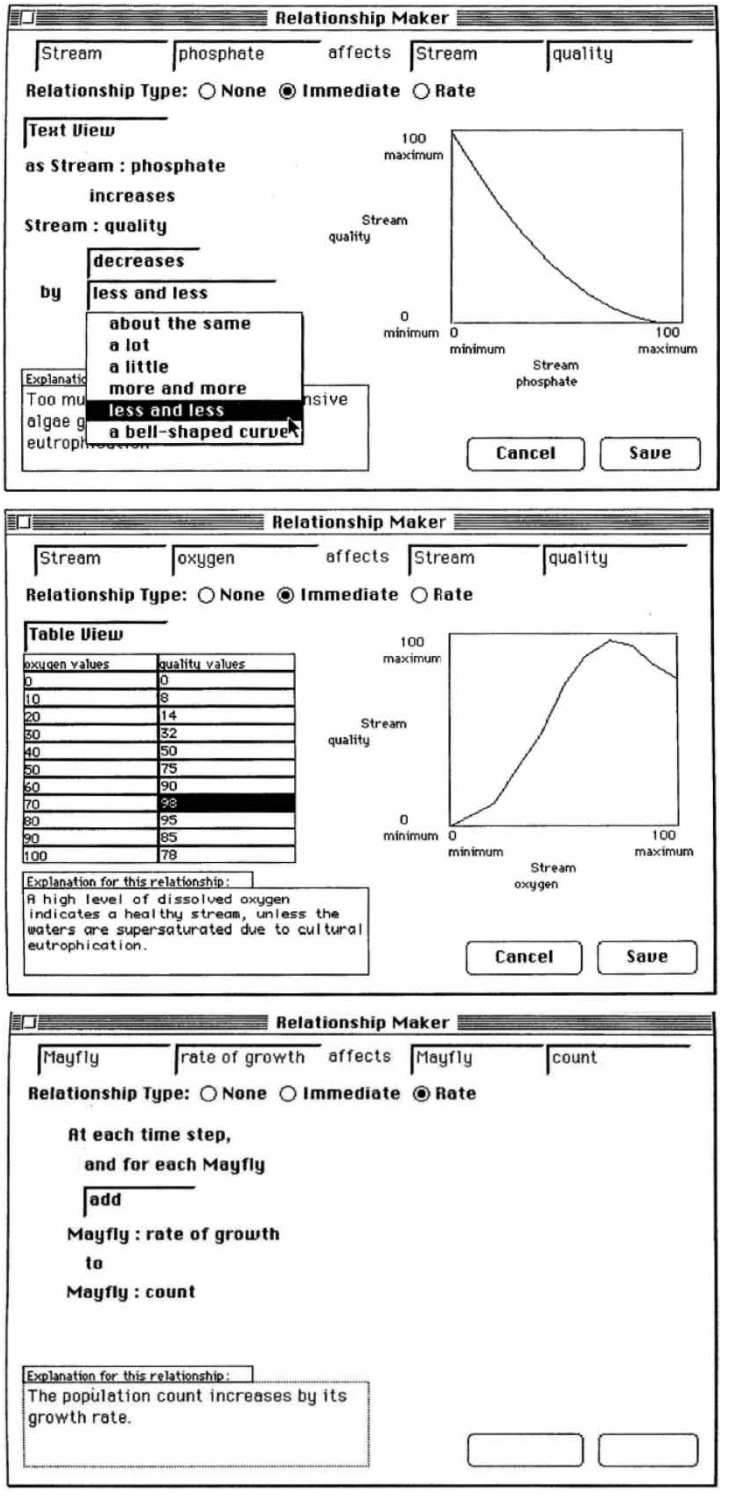

Figure 8. Methods of expressing relationships in Model-It. Reprinted from Metcalf (1999), with permission of the author.

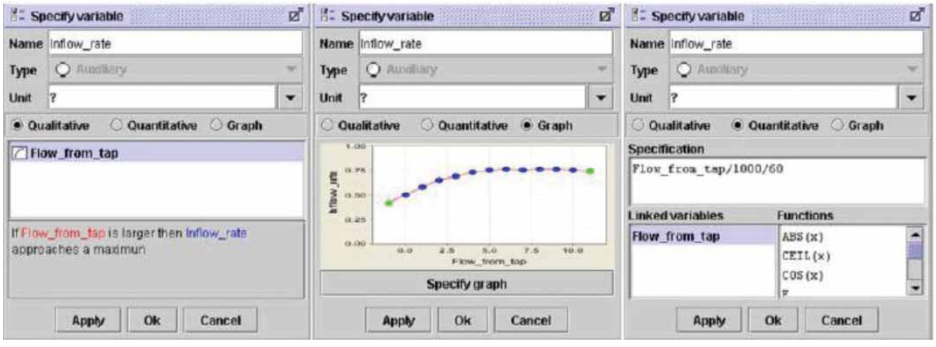

Figure 9. Methods for expressing relationships in CoLab. Reprinted from van Joolingen et al. (2005, Figure 4), with permission of Elsevier Limited.

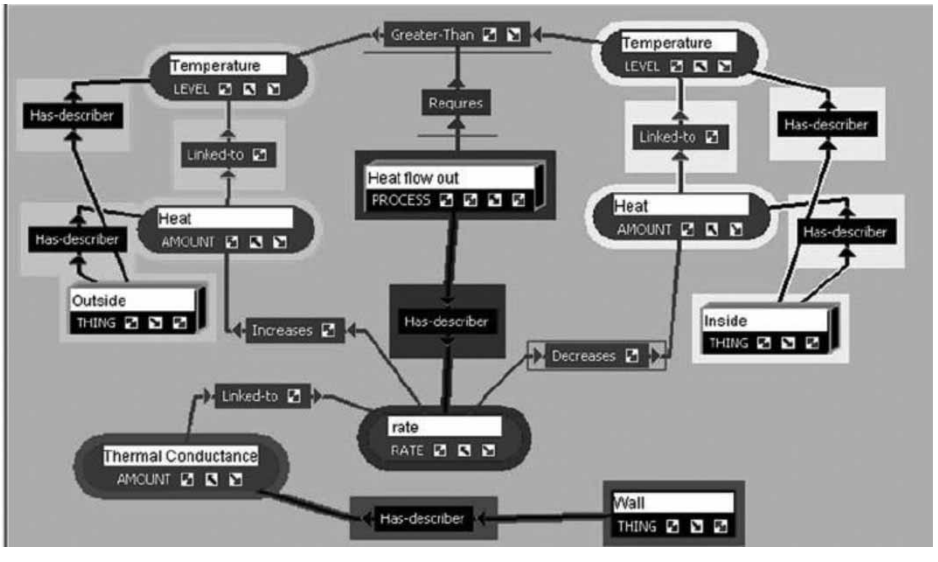

Figure 10. Objects, processes and quantities are all displayed in Vmodel. Reprinted from Bredeweg and Forbus (2003, p.38), with permission of the American Association for Artificial Intelligence.

the Logo programming language. CTiM provides a graphical programming language instead (Basu et al., 2012). The two notations have not yet been compared empirically.

8.2.2 *Grounding the symbols*

Models often have symbols that denote objects, properties or quantities in the system. "Grounding a symbol" is the process of getting the student and another agent (e.g. the teacher, another student or a tutoring system) to agree on what the symbol denotes. For instance, if the student chooses "birth rate" as a variable name in a system-dynamics model, it could denote either the number of births per unit time or the ratio of births to population. When the student is using a tutoring system for model construction that gives feedback on the students' model, then the student and the tutoring system need to ground symbols. That is, they need to mutually agree on what every symbol in the model stands for. There are several common methods for doing this:

- The software provides names for symbols and the students select those they wish to use in the model. The software designers try to choose names that are so clear that the students have no trouble understanding what they mean. This assumes that the students read the symbols' names carefully, which is not often the case, especially when the names become long.
- The students name the symbols. The software has to match the students' name to names it expects and knows the meanings of. In one study, the software's matches were correct 78% of the time (Bravo et al., 2009).
- The presentation of the system has elements that can be selected and used as names. For instance, if the presentation is text, then phrases in the text can be selected as names for variables (Marshall et al., 1989).

Model-it (Metcalf et al., 2000), ModelingSpace (Avouris et al., 2003) and Vmodel (Forbus et al., 2005) have students first define objects and then define variables as quantitative properties of the objects. Thus, students would first define "stream" as an object then define "phosphate concentration" as a quantitative property of it. Both objects and quantities were represented with student-selected icons. However, students often chose the same icons for different quantities.

In order to help ground variables, Model-It had students describe the range of a variable qualitatively (Metcalf et al., 2000). For instance, after typing in "% of tree cover" as the name, one student described the upper and lower ends of its range as "dense" and "sparse" (Metcalf, 1999). This qualitative labeling probably was more helpful than "100%" and "0%" for keeping in mind what the variable meant. ModelingSpace had a similar feature, plus the ability to display the qualitative value of a variable as an image (Avouris et al., 2003).

Some instructors feel strongly that grounding can be improved by requiring students to use units on the dimensional numbers that appear in equations and variable assignments, while other instructors believe that such attention to detail can be confusing, particularly when the dimensional number appears inside an equation (e.g. substitution converts $F = ma$ to either $F = 5a$ or $F = 5\text{kg}a$ or $F = (5\text{ kg})a$ or $F = 5\text{ kg }a$). Modeling languages also vary on their policies for use of units.

None of these scaffolding methods have been evaluated empirically. Moreover, similar methods have not been applied to agent-based models.

8.2.3 *Comparing the model's predictions to the system's behavior*

It is common for students to see the predictions of the model as being consistent with the behavior of the system when in fact the two behaviors are different. This section presents several methods for encouraging diligent comparison of predictions to behaviors.

For constraint-based models, one method involves asking students to manipulate the values of variables with sliders and observe the effects in real time on predictions of the system. For instance, Figure 11 shows a model in Q-Mod (Figure 25 of Miller et al., 1993). The vertical sliders inside the nodes function both as displays of the values of variables and as controls that can be dragged by students in order to modify the values of variables. The text inside the node displays the node's current value. This kind of user interface is widely used for model-exploration exercises, where students are given a model and asked to understand it by manipulating sliders and observing gauges.

For system-dynamics models, the same idea can be applied but it becomes more complicated. Such models predict the values of variables *over time*. Thus, the display of a variable's predicted behavior cannot show just a single value, but must instead show a graph of the

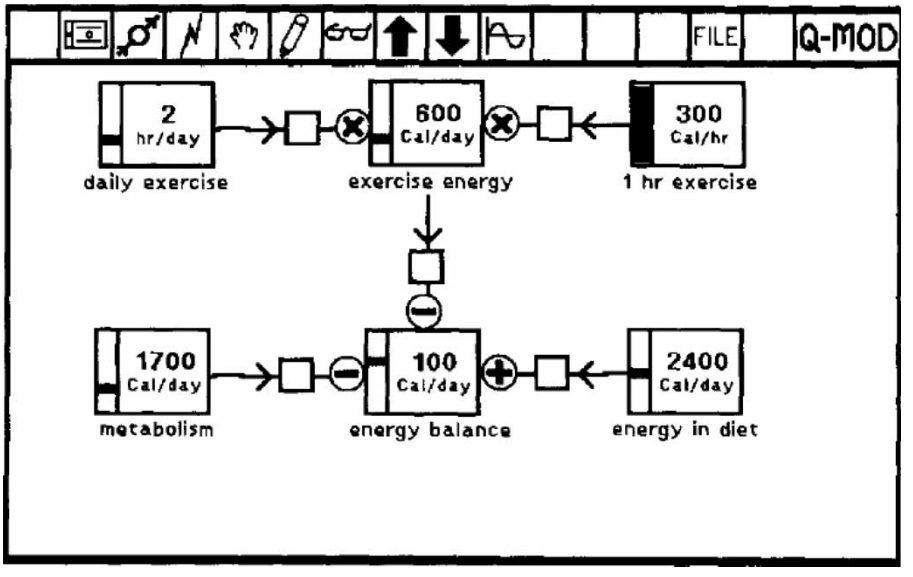

Figure 11. Sliders allow both controlling and displaying quantities. Reprinted from Miller et al. (1993, Figure 25), with permission of Elsevier Limited.

variables' value over time. When students manipulate a slider that controls a variable's value, the graphs of the other variables change in real time. This feature is available in many professional system-dynamics model-construction system, as well as Model-It (Metcalf et al., 2000).

For agent-based models, a standard practice is to display the predictions of the model as both graphs (e.g. the number of slime molds) and as an animation, as shown in Figure 5. This practice may help students interpret the graphs and compare the predictions of the model against the system's known behavior.

The basic idea of animating the model's predictions can be applied to certain constraint-based models. For example, Animate is a model-construction system where students are presented with succinct text and asked to construct equations in a graphical language (Nathan, 1998; Nathan, Kintsch, & Young, 1992). However, it also has an animation that is driven by the model. For instance, if the problem states that a helicopter leaves 2 hours after a train and overtakes it, and the student's model uses *t_train* and *t_copter* for the *duration* of the train's and the copter's trips, but the student states that *t_train* = *t_copter* – 2, then the animation shows the helicopter leaving *before* the train. This use of multiple representations was more effective than a no-feedback control (Nathan et al., 1992). More interestingly, it was also more effective than a traditional hint-sequence feedback (Nathan, 1998). While both forms of feedback got students to correct their sign errors and other expression errors, the animation feedback was better at getting students to understand the correction and thus they did better on post-test problems, which were similar to the training problems. Of all the methods for helping students compare the model predictions to system behavior, this is the only one to be evaluated empirically.

8.2.4 *Students explain the model in natural language*

Students sometimes have the basic idea for a model but have trouble writing it in the modeling language. For instance, suppose students are given a problem such as "Alan, Stan and

Doug are picking cherries. Doug picks 7 gallons more cherries than Stan, and Stan picks half as many cherries at Alan. If Alan picks A cherries, how many does Doug pick?" Some students can express their idea for solving the problem as a procedure, such as "Take half of A and add 7 to it". However, they do not know how to write that basic idea as an algebraic expression.

The tutoring system Ms Lindquist (Heffernan et al., 2008) sometimes asked students for their basic approach and gave them a sequence of menus to phrase their answers. For instance, the student might pick "Divide A by 2" from the first menu and "then add 7 to it" from the second menu.

Model-It collected natural language explanations in several places (Metcalf, 1999; Metcalf et al., 2000). For instance, the system prompted students to type in explanations of

- Their goal for the model and their ideas for building it;
- What predictions they expected the model to generate;
- What a quantity they defined represented
- What a constraint they defined represented

Although Model-It allowed students to turn off the prompting for such explanations, teachers often insisted students write good quality explanations.

Although this scaffolding has not been evaluated in isolation, a tutoring system that used it along with other scaffolding was effective when compared to classroom and other controls (Razzaq, Mendicino, & Heffernan, 2008).

8.3 *Gradually increasing the complexity*

There is no doubt that model-construction tasks are complicated. In addition to the modeling skills and domain knowledge that are the instructional objectives, students must learn the syntax of the modeling language, its semantics, the user interface for the editor, the user interface for executing the model and displaying its predictions, and many other things as well. Thus, one form of scaffolding is to introduce these details slowly. That is, the model-construction activities start simple and gradually become more complex. Several variants of this basic idea have been fielded so far.

8.3.1 *Hiding model features*

Modeling tools often have advanced features that can confuse novice users. To prevent such confusions, the initial user interface can hide the advanced features. As users become more skilled, they can access a property sheet that allows them to turn the features on. For example, although Model-It (Metcalf et al., 2000) lets students enter the graphical equivalent of a differential equation (see Figure 8, bottom pane), this capability is initially hidden. This form of scaffolding has not yet been evaluated.

8.3.2 *Qualitative model construction as scaffolding*

Modeling tools that use quantitative diagrams (e.g. Stella, CoLab and Model-It) often provide concept maps as well. DynaLearn (Bredeweg et al., 2010) allows students to start by creating a concept map of the system then enhance it as they move through a progression of five executable modeling languages.

Although instructors often recommend that students start model construction by forming a concept map, students seldom do. In one study, only 3 of 60 subjects ever used this feature even though they had been trained in its usage (Mulder et al., 2010).

This suggests that students should be *required* to do concept mapping before doing quantitative model construction. Mulder, Lazonder, de Jong, Anjewierden, and Bollen (2011) created a progression where the modeling language evolved from simple to complex, but the system stayed the same. Unfortunately, none of the measures of learning showed reliable differences, probably because the model-construction tasks were too challenging and few students made much progress through the sequences. Thus, the virtues of this form of scaffolding remain unknown.

Instead of using non-executable concept mapping, students could use an executable qualitative modeling tool to prepare for learning quantitative model construction. To explore this kind of scaffolding, Kurtz dos Santos and Ogborn (1994) had half their students work with a qualitative modeling tool, IQON, and half drawing similar diagrams on paper. Both groups then transferred to using a quantitative model-construction tool, Stella. The authors observed qualitative differences in the behaviors of the two groups, but stopped short of recommending one form of scaffolding over another.

8.3.3 *Model progressions*

The progressions discussed so far vary the complexity of the modeling language and tools. Another form of model progression varies the domain content and difficulty. For instance, several projects have developed a sequence of simulation-based problem solving activities and games that gradually increased in domain content and complexity (de Jong et al., 1999; Quinn & Alessi, 1994; Swaak, van Joolingen, & de Jong, 1998; White, 1984, 1993; White & Frederiksen, 1990). When these model progressions are compared to unordered sequences, the results were mixed. Sometimes the progression helped learning and sometimes it did not. However, all these model progression studies had students do model exploration. That is, they were not asked to construct models in a formal language.

8.4 ***Other scaffolding***

This section simply lists types of scaffolding that do not easily fit in the preceding three categories.

8.4.1 *Teachable agents and reciprocal teaching*

A widespread belief is that teaching is also a good way to learn. Thus, several projects have human students teach software that acts like a student (Biswas et al., 2005; Chase, Chin, Oppenzzo, & Schwartz, 2009; Chin et al., 2010; Gulz, Haake, & Silvervarg, 2011; Leelawong & Biswas, 2008; Matsuda et al., 2011; Obayashi, Shimoda, & Yoshikawa, 2000; Pareto, Arvemo, Dahl, Haake, & Gulz, 2011; Tan & Biswas, 2006; Tan, Wagster, et al., 2007; Wagster, Tan, Biswas, et al., 2007; Wagster, Tan, Wu, et al., 2007). The software that acts like a student is called a *teachable agent* or simulated student. Two modes of interaction are employed. One mode allows the human student to directly edit the knowledge of the teachable agent; the other mode requires the human student to tutor the teachable agent using natural language, examples and other communication methods that would be appropriate between human students. In the latter case, the teachable

agent's knowledge is not editable by the human student and is typically not displayed to the student either.

In some studies (Chan & Chou, 1997; Reif & Scott, 1999), the role of tutor vs. tutee switched back and forth between human and computer agent. This instructional method is called *reciprocal teaching* (Palinscar & Brown, 1984).

If the knowledge representation language used for the teachable agent is a modeling language, and the human student can edit the agent's knowledge, then teaching the agent is a kind of model-construction activity. Perhaps, the best known example of this class of system is Betty's Brain (Biswas et al., 2005; Biswas et al., 2010; Chase et al., 2009; Leelawong & Biswas, 2008; Schwartz et al., 2008; Schwartz et al., 2009; Tan & Biswas, 2006; Tan et al., 2006; Tan, Skirvin, Biswas, & Catley, 2007; Tan, Wagster, et al., 2007; Wagster, Tan, Biswas, et al., 2007; Wagster, Tan, Wu, et al., 2007). The users of Betty's Brain edit the qualitative diagrams shown in Figure 2. They are asked to pretend that the diagram is Betty's knowledge. Instead of executing the model and checking that its predictions match the behavior of a given system, students can either ask Betty a question (as shown in Figure 2) or have her take a quiz composed of a small number of such questions. Either way, the qualitative diagram is executed and its results are shown to the student. In the case of the quiz, the results are also marked as correct vs. incorrect depending on whether they match or do not match the behavior of the system. The system presented to the student as a set of resources describing, e.g. the oxygen cycle in streams and lakes.

Several studies have shown that the teachable agent cover story improves learning compared to a version of the system without the teachable agent cover story (Biswas et al., 2005; Chase et al., 2009; Pareto et al., 2011). In a particularly well-controlled experiment (Chase et al., 2009), students used Betty's Brain but were told that the model represented their own knowledge and not Betty's. The Betty agent was turned off but Mr Davis the tutor was still present. Students who thought they were writing models learned significantly less than students who thought they were teaching Betty. Chase et al. call this the Protégé effect. The Protégé effect can be enhanced by having the teachable agent make small talk with the human students during breaks in the instruction (Gulz et al., 2011). These studies suggest that teachable agents and reciprocal teaching can be quite effective at increasing learning during model construction.

8.4.2 *Having students execute models mentally*

One way to have students examine their model more closely is to have them execute it mentally. This can be done by asking qualitative questions such as, "If reflectance of the earth's atmosphere decreases, what happens to the earth's temperature?" Students can only reason for short distances (Kurtz dos Santos & Ogborn, 1994; Löhner, 2005). Nonetheless, this does raise their scores when quizzed on the structure of the model (Löhner, 2005, chapter 5), and could potentially raise scores on concept inventory questions, as discussed earlier.

8.4.3 *Test suites*

When there are multiple ways to test a model, students often do a poor job of selecting tests. They often focus on just a few tests out of the many possible ones (Alessi, 2000a). This suggests giving them test suites, which are composed by an expert to more thoroughly sample the space of all possible tests.

However, in one evaluation of Betty's Brain, test suites proved to be harmful. Students who had to generate their own tests performed better than students who were given test suites in the guise of a quiz (Biswas et al., 2005). Perhaps students could utilize test suites better if either they were given explicit instruction on how to use them, or the test suites were covertly designed to highlight just one bug in the model and students were prompted to find it.

8.4.4 *Generic schemas*

An analysis of algebra and arithmetic word problems indicates that a surprisingly small set of schemas suffices to characterize most of them (Marshall et al., 1989; Mayer, 1981). These schemas are not as abstract as mathematical relationships, yet they are general enough to cover multiple word problems. For example, Marshal et al. (1989) found that most single-step arithmetic problem could be characterized using only 5 schemas: Change, Group, Compare, Restate and Vary. Similarly, Mayer (1981) found that 90 schemas sufficed to cover a large corpus of high school algebra problems. For instance, there were 13 schemas for motion of an object including simple distance-rate-time, one object overtaking another, objects moving in opposite directions, an object making a round trip, and 9 others. There were nine schemas for work done per unit time: simple work problems, two processes working together, one process joining another to finish a job and six others. There were 8 schemas involving interest on investments, 5 schemas involving mixtures, 10 schemas involving travelling on a river and so on.

While analyzing videos of competent physics students working problems in pairs at the whiteboard, Sherin (2006) concluded that they use a type of schema he called *symbolic forms* to generate parts of models. For instance, if the students viewed the system as having balancing forces, then that would trigger their Balancing symbolic form, which suggested an equation of the form $\square = \square$. On the other hand, if they viewed the system as one force canceling out another, that would trigger the cancelation symbolic form, which suggested an equation of the form $0 = \square - \square$.

Schema libraries were provided by some model-construction environments (Bredeweg & Forbus, 2003; Bridewell et al., 2006; Marshall et al., 1989). A simple problem could be solved by copying a generic schema from the library and editing it to make it applicable to the given system. For more complex problems, students may combine multiple schemas to form the model. Some of these systems coached students on how to combine schemas in syntactically legal ways. Students were also able to add schemas to the library.

Unfortunately, schema-based scaffolding has not yet been evaluated. Such an evaluation could be done by comparing two versions of the model-construction activity: with and without the schema library.

8.4.5 *Gamification*

Gamification is adding game-like features to an educational activity. There are many ways that gamification can be done. For example, one mode of Betty's Brain put models in the role of players in a game show (Schwartz et al., 2009). The host of the game show would ask a player/model questions such as, "If the amount of fluoride in the water goes up, what happens to tooth decay?" Models that answered the question correctly would score points. The game show typically had four models that competed against each other. Although observation in the classroom leaves no doubt about the motivational benefits of the game show, its impact on learning has not been evaluated.

8.5 *Summary: which forms of scaffolding have demonstrated benefits?*

Of the forms of scaffolding reviewed here, meta-tutoring has the most empirical evidence for its benefits. Although giving hints and feedback on the models that students construct appears to be problematic, giving hints and feedback on the model-construction process (i.e. "meta-tutoring") has been shown in several studies to produce larger learning gains than instruction that uses the same modeling tool without the meta-tutoring. Three other techniques from the tutoring system literature – the concrete articulation strategy, decomposition and reflective debriefings – have a few evaluations behind them, but have not been extensively tested. Finally, students ask too few questions to warrant tutoring based on answering student questions. On the whole, the evidence suggests that reliable benefits can be obtained with the basic approach of guiding students via prompting, hinting and feedback through the process of model construction including decomposition, reflection and perhaps concrete articulation.

Another form of scaffolding with good empirical support is the use of teachable agents and reciprocal teaching. These have mostly been used with younger students in classroom contexts, and the leading explanations for their benefits are motivational rather than cognitive. It is not clear whether they will continue to provide added value when they are used by older students or by students working alone.

As Sections 7.2 and 7.3 indicated, many notations have been tried for modeling languages, and many methods for helping students over the complexities and semantic issues of the languages have also been tried. Unfortunately, little experimental evidence exists about the effectiveness of notation-based scaffolding. It seems plausible that there are large benefits, but the experiments still need to be done.

Little evidence exists concerning the benefits of the remaining forms of scaffolding, which are covered in sections 7.4.2 onwards. Of these, generic schemas seem most plausibly helpful, but again, experimental evidence is lacking.

9. Conclusions

The study of model construction as a learning activity is fairly new, and the field has spent most of its energy just trying to find ways to get students to engage in model construction successfully. Many of the studies have been formative evaluations whose results are intended only to help redesign the instruction. Many papers allude to pilot studies but do not report their findings in detail. In only a few cases has the model-construction instruction reached a state of maturity that the developers thought it would be useful to do a summative evaluation. Although all these evaluations were mentioned earlier, they will be summarized here followed by suggestions for future work.

9.1 *Summative evaluations of model construction compared to other instruction*

Eight studies have compared instruction based on model construction to instruction that does not involve model construction. In four of these, concept mapping was the comparison instruction:

- Chin et al. (2010, study 1) found that Betty's Brain was significantly more effective than using commercial concept-mapping software. Sixth-grade students studied global warming for 11 classes over a 3-week period.

- Biswas et al. (2005, study 1) found that Betty's Brain was significantly more effective than concept mapping using Betty's modeling language but with execution turned off.[1] Fifth-grade students learned about river ecosystems during three one-hour sessions with the software. Students engaged in additional independent study between sessions in order to prepare themselves to use the software.
- Kurtz dos Santos and Ogborn (1994) found that IQON caused different behaviors than having students do concept mapping using its notation on paper. Qualitative observation of both learning and transfer behavior was presented, but no numerical measures of learning.
- Löhner (2005, chapter 5) compared students using CoLab to students doing concept mapping using CoLab's modeling language with execution turned off. Students in grades 11 and 12 constructed either executable models or concept maps of the greenhouse effect over 4 50-minute lessons that include some guided model exploration as well as model construction. For factual knowledge, the concept-mapping instruction was significantly more effective. For predicting the system's behavior, the model construction instruction was more effective. For explanation questions, there was a floor effect – both conditions answered poorly.

These four studies suggest that concept mapping is less effective than instruction based on model construction. However, this might be due to a difference in the availability of feedback. If students in the concept-mapping condition received no feedback on the correctness of their maps, that might explain their disadvantage. Although Chin et al. (2010) carefully controlled for feedback, it is not clear that the other studies did.

Three of the studies compared model construction to having students write summaries and explanations:

- van Borkulo et al. (2012) found that learning from CoLab was superior to writing summaries guided by a worksheet. Students in the 11th grade first learned about model construction for 2.5 hours then learned about global warming using either CoLab or writing. Perhaps due to the short duration of the manipulation, statistically reliable differences were observed for only two of eight assessments types.
- Lee et al. (2011) found that Model-It was significantly more effective than having students write while doing literature research. Fifth-grade students learned about the water cycle during a 10-week unit, whose second half involved either using Model-It or continuing with their directed study of the water cycle literature.
- Schwartz et al. (2008, study 1) compared Betty's Brain to having students write short explanations to questions. The small number of students (16 college students) and the short duration (40 minutes) prevented quantitative measures of results, but qualitative observations suggest the Betty's instruction was superior.

In all these studies, students were presented information about the system via access to multimedia resources, and they were asked to answer the same set of questions about the system. The experimental groups formulated their answers as models, and the control group formulated their answers as writing. It is not clear how much feedback on the writing was given to students in the control groups.

The last study compared model construction to three different types of model exploration (Hashem & Mioduser, 2011). This study had science majors in college work with two different NetLogo models. Prior to pre-testing, the model-construction group spent 48 hours learning how to program NetLogo, and the model-exploration groups attended a

two-hour introduction to NetLogo. The instructional manipulation occurred over two 90-minute sessions. It was preceded by a pre-test, and followed by a post-test and interviews. Model construction proved to be more effective than model exploration by a moderately large effect size: $d = 0.48$ (Hashem & Mioduser, 2011, Table 2).

As usual when reviewing studies of learning, the results are diverse and incomplete. Nonetheless, the results strongly suggest that model construction can be more effective than concept mapping and perhaps writing and concept exploration as well.

9.2 *Future work*

Perhaps, the major problem with using model construction for instruction is that it takes a long time for students to become skilled enough that they can actually start learning about the system and the domain. For instance, most of the studies of Betty's Brain take about 10 lessons spread over 2 or 3 weeks. It takes that long for the benefits of modeling to show up. Figure 12, from (Chin et al., 2010), illustrates this vividly. The *x*-axis divides show student performance on a pre-test, two mid-tests and a post-test. That is, about every 4 hours of instruction, there was a test. The students' scores on three different types of test questions are shown: those requiring short, medium and long chains of reasoning. The model-construction group begins to pull away from the concept-mapping group (who were using the commercial software Inspiration) only after 8 hours of instruction, and only on medium-length chains of reasoning. It is not until 11 lessons have been conducted that the benefits of model construction appear on all question types.

Of the experiments reviewed here, most of those that showed statistically significant differences used long time periods for the instruction. The experiments that produced null effects typically used short time periods (e.g. 2 hours) for the instruction. Clearly, experimenters should all be using multi-hour instruction in model construction despite the difficulty of running such experiments.

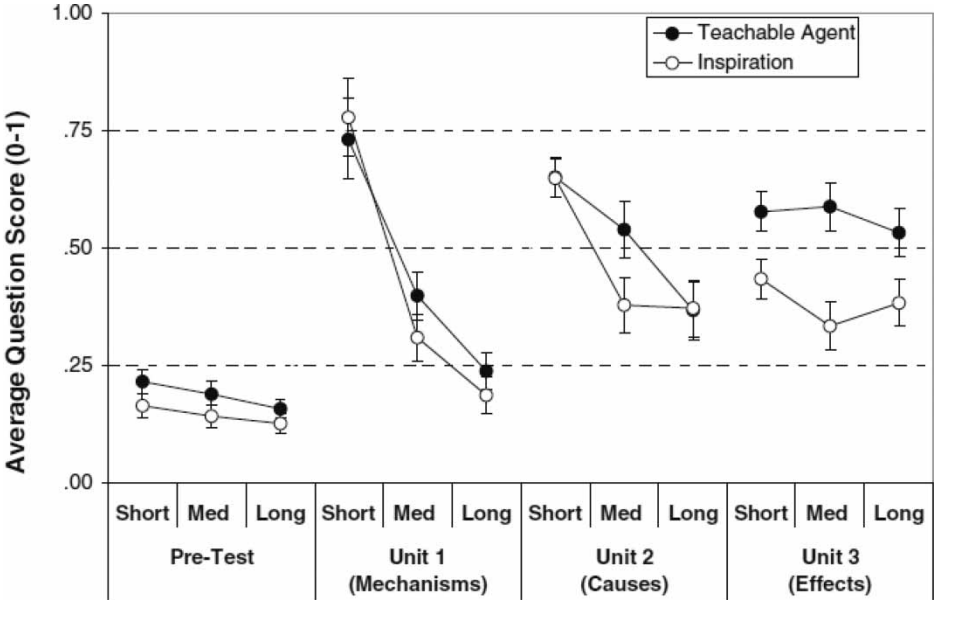

Figure 12. It takes many hours of instruction before model construction shows benefits over concept mapping. Reprinted from Chin et al. (2010, Figure 4), with the kind permission of Springer Science and Business Media.

Unless some method for dramatically speeding up students' learning of model construction is found, educators must face the prospect that meeting the new standards in modeling is going to require a substantial commitment of class time. On the other hand, once students have acquired the appropriate model-construction skills, it appears that they can use them to more easily acquire a deep understanding of systems and domains. Thus, a primary goal for future work should be to reduce the time required for students to become fluent in model construction.

Acknowledgements

This research was supported by the National Science Foundation under grants DRL-0910221, IIS-1123823 and DUE-1140901 and by the Office of Naval Research under contract N00014-13-C-0029.

Note

1. Although Biswas et al. (2005) do not present a statistical analysis of this comparison, the means and error bars on Figure 5(b) suggest that it would be reliable and large. However, these measures were taken over the models and concept maps constructed by students as they used the software, and thus do not comprise a typical post-test.

Notes on contributor

Kurt VanLehn is the Diane and Gary Tooker Chair for Effective Education in Science, Technology, Engineering and Math in the Ira A. Fulton Schools of Engineering at Arizona State University. He received a Ph.D. from MIT in 1983 in Computer Science, was a post-doc at BBN and Xerox PARC, joined the faculty of Carnegie-Mellon University in 1985, moved to the University of Pittsburgh in 1990 and joined ASU in 2008. He founded and co-directed two large NSF research centers (Circle; the Pittsburgh Science of Learning Center). He has published over 125 peer-reviewed publications, is a fellow in the Cognitive Science Society, and is on the editorial boards of *Cognition and Instruction and the International Journal of Artificial Intelligence in Education*. Dr VanLehn's research focuses on intelligent tutoring systems and other intelligent interactive instructional technology.

References

Alessi, S. M. (2000a, December). *The application of system dynamics modeling in elementary and secondary school curricula.* Paper presented at the RIBIE 2000 – The Fifth Iberoamerican Conference on Informatics in Education, Viña del Mar, Chile.

Alessi, S. M. (2000b). Building versus using simulations. In J. M. Spector & T. M. Anderson (Eds.), *Integrated and holistic perspectives on learning, instruction and technology* (pp. 175–196). Dordrecht, The Netherlands: Kluwer.

Anthony, L., Corbett, A. T., Wagner, A. Z., Stevens, S. M., & Koedinger, K. R. (2004). Student question-asking patterns in an intelligent algebra tutor. In J. C. Lester, R. M. Vicari, & F. Praguacu (Eds.), *Intelligent tutoring systems: 7th international conference, ITS 2004* (pp. 455–467). Berlin: Springer-Verlag.

Avouris, N., Margaritis, M., Komis, V., Saez, A., & Melendez, R. (2003). *ModellingSpace: Interaction design and architecture of a collaborative modelling environment*. Paper presented at the Sixth International Conference on Computer Based Learning in Sciences (CBLIS), Nicosia, Cyprus.

Baker, R. S. J. d., Corbett, A. T., Koedinger, K. R., & Wagner, A. Z. (2004). Off-task behavior in the cognitive tutor classroom: When students "Game the System". In E. Dykstra-Erickson & M. Tscheligi (Eds.), *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 383–390). New York, NY: ACM.

Basu, S., Kinnebrew, J. S., Dickes, A., Farris, A. V., Sengupta, P., Winger, J., & Biswas, G. (2012). *A science learning environment using a computational thinking approach*. Paper presented at the Proceedings of the 20th International Conference on Computers in Education, Singapore.

Beek, W., Bredeweg, B., & Lautour, S. (2011). Context-dependent help for the DynaLearn modelling and simulation workbench. In G. Biswas (Ed.), *Artificial intelligence in education* (pp. 4200–4422). Berlin: Springer-Verlag.

Biswas, G., Jeong, H., Kinnebrew, J. S., Sulcer, B., & Roscoe, R. D. (2010). Measuring self-regulated learning skills through social interactions in a teachable agent environment. *Research and Practice in Technology Enhanced Learning*, *5*(2), 123–152.

Biswas, G., Leelawong, K., Schwartz, D. L., & Vye, N. J. (2005). Learning by teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence*, *19*, 263–392.

Boohan, R. (1995). Children and computer modelling: Making worlds with WorldMaker. In J. D. Tinsley & T. J. van Weert (Eds.), *Proceedings of the sixth world conference on computers in education* (pp. 975–985). London: Chapman and Hall.

Booth, L. R. (1988). Children's difficulties in beginning algebra. In F. Coxford (Ed.), *The ideas of algebra, K-12 (1988 Yearbook*) (pp. 20–32). Reston, VA: NCTM.

Booth Sweeney, L., & Sterman, J. D. (2000). Bathtub dynamics: Initial results of a systems thinking inventory. *System Dynamics Review*, *16*(4), 249–286.

van Borkulo, S. P., van Joolingen, W. R., Savelsbergh, E. R., & de Jong, T. (2012). What can be learned from computer modeling? Comparing expository and modeling approaches to teaching dynamic systems behavior. *Journal of Science Education and Technology*, *21*, 267–275.

Bravo, C., van Joolingen, W. R., & de Jong, T. (2009). Using Co-Lab to build system dynamics models: Students' actions and on-line tutorial advice. *Computer and Education*, *53*, 243–251.

Bredeweg, B., & Forbus, K. D. (2003). Qualitative modeling in education. *AI Magazine*, *24*(4), 35–46.

Bredeweg, B., Liem, J., Beek, W., Salles, P., & Linnebank, F. (2010). Learning spaces as representational scaffolds for learning conceptual knowledge of system behavior. In M. Wolpers (Ed.), *EC-TEL* (pp. 46–61). Berlin: Springer-Verlag.

Bredeweg, B., Linnebank, F., Bouwer, A., & Liem, J. (2009). Garp3 – Workbench for qualitative modelling and simulation. *Ecological Informatics*, *4*, 263–281.

Bridewell, W., Sanchez, J. N., Langley, P., & Billman, D. (2006). An interactive environment for the modeling and discovery of scientific knowledge. *International Journal of Human-Computer Studies*, *64*, 1099–1114.

Buckley, B. C., Gobert, J. D., Kindfield, A. C. H., Horwitz, P., Tinker, R. F., Gerlits, B., … Willett, J. (2004). Model-based teaching and learning with BioLogica: What do they learn? How do they learn? How do we know? *Journal of Science Education and Technology*, *13*(1), 23–41.

Chan, T.-W., & Chou, C.-Y. (1997). Exploring the design of computer supports for reciprocal tutoring. *International Journal of Artificial Intelligence and Education*, *8*, 1–29.

Chase, C. C., Chin, D. B., Oppenzzo, M., & Schwartz, D. L. (2009). Teachable agents and the Protégé effect: Increasing the effort towards learning. *Journal of Science Education and Technology*, *18*(4), 334–352.

Chi, M., & VanLehn, K. (2008). Eliminating the gap between the high and low students through meta-cognitive strategy instruction. In B. P. Woolf, E. Aimeur, R. Nkambou, & S. P. Lajoie (Eds.), *Intelligent tutoring systems: 9th international conference: ITS2008* (pp. 603–613). Berlin: Springer.

Chi, M., & VanLehn, K. (2010). Meta-cognitive strategy instruction in intelligent tutoring systems: How, when and why. *Journal of Educational Technology and Society*, *13*(1), 25–39.

Chin, D., Dohmen, I. M., Cheng, B. H., Oppezzo, M., Chase, C. C., & Schwartz, D. L. (2010). Preparing students for future learning with teachable agents. *Educational Technology Research and Development*, *58*, 649–669.

Clariana, R. B., & Strobel, J. (2007). Modeling technologies. In J. M. Spector (Ed.), *Handbook of research on educational communications and technology* (pp. 329–344). New York: Taylor & Francis.

Clark, D. B., Nelson, B. C., Sengupta, P., & D'Angelo, C. (2009). Rethinking science learning through digital games and simulations: Genres, examples and evidence. An NAS commissioned paper.

Clark, H. H., & Brennan, S. E. (1991). Grounding in communication. In L. B. Resnick, J. M. Levine, & S. D. Teasley (Eds.), *Perspectives on socially shared cognition* (pp. 127–149). Washington, DC: American Psychological Association.

Clement, J. (1982). Algebra word problem solutions: Thought processes underlying a common misconception. *Journal of Research in Mathematics Education*, *13*(1), 16–30.
Collins, A., & Ferguson, W. (1993). Epistemic forms and epistemic games: Structures and strategies to guide inquiry. *Educational Psychologist*, *28*(1), 25–42.
CCSSO. (2011). The common core state standards for mathematics. Retrieved October 31, 2011, from www.corestandards.org
Connelly, J., & Katz, S. (2009). Toward more robust learning of physics via reflective dialogue extensions. In G. Siemens & C. Fulford (Eds.), *Proceedings of the world conference on educational multimedia, hypermedia and telecommunications 2009* (pp. 1946–1951). Chesapeake, VA: AACE.
Corbett, A., Wagner, A. Z., Chao, C.-y., Lesgold, S., Stevens, S. M., & Ulrich, H. (2005). Student questions in a classroom evaluation of the ALPS learning environment. In C.-K. Looi & G. McCalla (Eds.), *Artificial intelligence in education* (pp. 780–782). Amsterdam: IOS Press.
Core, M. G., Moore, J. D., & Zinn, C. (2003). The role of initiative in tutorial dialogue. In P. Paroubek (Ed.), *Proceedings of the 11th conference of the European chapter of the association for computational linguistics (EACL)* (pp. 67–74). Morristown, NJ: Association of Computational Linguistics.
Crawford, B. A., & Cullin, M. (2004). Supporting prospective teachers' conceptions of modelling in science. *International Journal of Science Education*, *26*(11), 1370–1401.
Cronin, M., Gonzalez, C., & Sterman, J. D. (2009). Why don't well-educated adults understand accumulation? A challenge to researchers, educators and citizens. *Organizational Behavior and Human Decision Processes*, *108*, 116–130.
Doerr, H. M. (1996). Stella ten-years later: A review of the literature. *International Journal of Computers for Mathematical Learning*, *1*, 201–224.
Erden, M., Komoto, H., van Beek, T. J., D'Amelio, V., Echavarria, E., & Tomiyama, T. (2008). A review of function modeling: Approaches and applications. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *22*(2), 147–169.
Forbus, K. D., Carney, K., Sherin, B. L., & Ureel Il, L. C. (2005). VModel: A visual qualitative modeling environment for middle-school students. *AI Magazine*, *26*(3), 63–72.
Gentner, D., & Stevens, A. L. (1983). *Mental models*. Mahah, NJ: Erlbaum.
Goel, A. K., Rugaber, S., & Vattam, S. (2009). Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *23*, 23–35.
Graesser, A. C., Person, N., & Magliano, J. (1995). Collaborative dialog patterns in naturalistic one-on-one tutoring. *Applied Cognitive Psychology*, *9*, 359–387.
Gulz, A., Haake, M., & Silvervarg, A. (2011). Extending a teachable agent with a social conversation module – Effects on student experiences and learning. In G. Biswas, S. Bull, J. Kay, & A. Mitrovic (Eds.), *Artificial intelligence in education* (pp. 106–114). Berlin: Springer.
Harrison, A. G., & Treagust, D. F. (2000). A typology of school science models. *International Journal of Science Education*, *22*(9), 1011–1026.
Hashem, K., & Mioduser, D. (2010). *Learning by modeling (LbM) – The contribution of computer modeling to students' evolving understanding of complexity.* Paper presented at the Second International Conference on Educational Technology and Computer (ICETC), Shanghai, China.
Hashem, K., & Mioduser, D. (2011). The contribution of learning by modeling (LbM) to students' understanding of complexity concepts. *International Journal of e-Education, e-Business, e-Management and e-Learning*, *1*(2), 151–157.
Heffernan, N. T. (2001). *Intelligent tutoring systems have forgotten the tutor: Adding a cognitive model of human tutors* (PhD dissertation). Carnegie Mellon University, Pittsburgh, PA. Retrieved from http://gs260.sp.cs.cmu.edu/diss/diss.pdf
Heffernan, N. T., & Koedinger, K. R. (1997). The composition effect in symbolizing: The role of symbol production vs. text comprehension. In M. G. Shafto & P. Langley (Eds.), *Proceedings of the nineteenth annual meeting of the cognitive science society* (pp. 307–312). Mahwah, NJ: Erlbaum.
Heffernan, N. T., Koedinger, K. R., & Razzaq, L. (2008). Expanding the model-tracing architecture: A 3rd generation intelligent tutor for algebra symbolization. *International Journal of Artificial Intelligence in Education*, *18*, 153–178.
Hestenes, D. (2007). *Modeling theory for math and science education*. Paper presented at the ICTMA-13: The International Community of Teachers of Mathematical Modelling and Applications, Indiana, IL.

Hestenes, D., Wells, M., & Swackhamer, G. (1992). Force concept inventory. *The Physics Teacher*, *30*, 141–158.

Hmelo-Silver, C., & Pfeffer, M. G. (2004). Comparing expert and novice understanding of a complex system from the perspective of structures, behaviors and functions. *Cognitive Science*, *28*, 127–138.

Hogan, K., & Thomas, D. (2001). Cognitive comparisons of students' systems modeling in ecology. *Journal of Science Education and Technology*, *10*(4), 319–345.

Hopper, M., & Stave, K. (2008). *Assessing the effectiveness of systems thinking interventions in the classroom*. Paper presented at the International Conference of the System Dynamics Society, Athens, Greece. Retrieved from http://www.systemdynamics.org/conferences/2008/proceed/index.htm

Jacobson, M. J., & Wilensky, U. (2006). Complex systems in education: Scientific and educational importance and implications for the learning sciences. *Journal of the Learning Sciences*, *15*(1), 11–34.

Jonassen, D., & Strobel, J. (2006). Modeling for meaningful learning. In D. Hung & M. S. Khine (Eds.), *Engaged learning with emerging technologies* (pp. 1–27). Amsterdam: Springer.

de Jong, T., Martin, E., Zamarro, J.-M., Esquembre, F., Swaak, J., & van Joolingen, W. R. (1999). The integration of computer simulation and learning support: An example from the physics domain of collisions. *Journal of Research in Science Teaching*, *36*(5), 597–615.

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.

van Joolingen, W. R., De Jong, T., Lazonder, A., Savelsbergh, E. R., & Manlove, S. (2005). Co-Lab: Research and development of an online learning environment for collaborative scientific discovery learning. *Computers in Human Behavior*, *21*, 671–688.

Kainz, D., & Ossimitz, G. (2002). *Can students learn stock-flow-thinking? An empirical investigation*. Paper presented at the System Dynamics Conference, Palermo, Italy.

Katehi, L., Pearson, G., & Feder, M. (2008). Engineering in K-12 Education: Understanding the status and improving the prospects. Retrieved October 31, 2011, from http://www.nap.edu/catalog.php?record_id=12635

Katz, S., Allbritton, D., & Connelly, J. (2003). Going beyond the problem given: How human tutors use post-solution discussions to support transfer. *International Journal of Artificial Intelligence in Education*, *13*, 79–116.

Katz, S., Connelly, J., & Wilson, C. (2007). Out of the lab and into the classroom: An evaluation of reflective dialogue in Andes. In R. Luckin & K. R. Koedinger (Eds.), *Proceedings of AI in education, 2007* (pp. 425–432). Amsterdam, The Netherlands: IOS Press.

Kessler, C. (1988). *Transfer of programming skills in novice Lisp learners* (PhD dissertation). Carnegie Mellon University, Pittsburgh, PA.

de Kleer, J., & Brown, J. S. (1981). Mental models of physical mechanisms and their acquisition. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 285–310). Mahwah, NJ: Erlbaum.

Koedinger, K. R., Alibali, M. W., & Nathan, M. J. (2008). Trade-offs between grounded and abstract representations: Evidence from algebra problem solving. *Cognitive Science*, *32*, 366–397.

Koedinger, K. R., & Anderson, J. R. (1998). Illustrating principled design: The early evolution of a cognitive tutor for algebra symbolization. *Interactive Learning Environments*, *5*, 161–180.

Koedinger, K. R., & Nathan, M. J. (2004). The real story behind story problems: Effects of representations on quantitative reasoning. *Journal of the Learning Sciences*, *13*(2), 129–164.

Kurtz dos Santos, A. d. C., & Ogborn, J. (1994). Sixth form students' ability to engage in computational modelling. *Journal of Computer Assisted Learning*, *10*(3), 182–200.

Lazonder, A., Hagemans, M. G., & de Jong, T. (2010). Offering and discovering domain information in simulation-based inquiry learning. *Learning and Instruction*, *20*, 511–520.

Lee, C. B., Jonassen, D., & Teo, T. (2011). The role of model building in problem solving and conceptual change. *Interactive Learning Environments*, *19*(3), 247–265.

Leelawong, K., & Biswas, G. (2008). Designing learning by teaching agents: The Betty's Brain system. *International Journal of Artificial Intelligence and Education*, *18*(3), 181–208.

Löhner, S. (2005). *Computer based modeling tasks: The role of external representation* (PhD dissertation). University of Amsterdam, Amsterdam, The Netherlands.

Löhner, S., Van Joolingen, W. R., & Savelsbergh, E. R. (2003). The effect of external representation on constructing computer models of complex phenomena. *Instructional Science*, *31*, 395–418.

Löhner, S., Van Joolingen, W. R., Savelsbergh, E. R., & Van Hout-Wolters, B. (2005). Students' reasoning during modeling in an inquiry learning environment. *Computers in Human Behavior*, *21*, 441–461.

Mandinach, E. B., & Cline, H. F. (1994). *Classroom dynamics: Implementing a technology-based learning environment*. Mahwah, NJ: Erlbaum.

Marshall, S. P., Barthuli, K. E., Brewer, M. A., & Rose, F. E. (1989). *Story problem solver: A schema-based system of instruction*. San Diego, CA: Center for Research in Mathematics and Science Education, San Diego State University.

Matsuda, N., Yarzebinski, E., Keiser, V., Raizada, R., Stylianides, G. J., Cohen, W. W., & Koedinger, K. R. (2011). Learning by teaching SimStudent – An initial classroom baseline study comparing with Cognitive Tutor. In G. Biswas & S. Bull (Eds.), *Proceedings of the international conference on artificial intelligence in education* (pp. 213–221). Berlin: Springer.

Mayer, R. E. (1981). Frequency norms and structural analysis of algebra story problems into families, categories and templates. *Instructional Science*, *10*(2), 135–175.

McArthur, D., Lewis, M., Ormseth, T., Robyn, A., Stasz, C., & Voreck, D. (1989). *Algebraic thinking tools: Support for modeling situations and solving problems in Kids' World*. Santa Monica, CA: RAND Corporation, p. 22.

Metcalf, S. J. (1999). *The design of guided learning-adaptable scaffolding in interactive learning environments* (PhD dissertation). University of Michigan, Ann Arbor, MI.

Metcalf, S. J., Krajcik, J., & Soloway, E. (2000). Model-It: A design retrospective. In M. J. Jacobson & R. B. Kozma (Eds.), *Innovations in science and mathematics education: Advanced designs for technologies of learning* (pp. 77–115). Mahwah, NJ: Lawrence Erlbaum Associates.

Miller, R., Ogborn, J., Briggs, J., Borough, D., Bliss, J., Boohan, R., … Sakonidis, B. (1993). Educational tools for computational modelling. *Computers and Education*, *21*(3), 205–261.

Moxnes, E. (2000). Not only the tragedy of the commons: Misperceptions of feedback and policies for sustainable development. *System Dynamics Review*, *16*(4), 325–348.

Mulder, Y. G., Lazonder, A., & de Jong, T. (2010). Finding out how they find it out: An empirical analysis of inquiry learners' need for support. *International Journal of Science Learning*, *32*(15), 2033–2053.

Mulder, Y. G., Lazonder, A. W., de Jong, T., Anjewierden, A., & Bollen, L. (2011). Validating and optimzing the effects of model progression in simulation-based inquiry learning. *Journal of Science Education and Technology*, *21*, 722–729.

Muldner, K., Burleson, W., van de Sande, B., & VanLehn, K. (2011). An analysis of students' gaming behaviors in an intelligent tutoring system: Predictors and impacts. *User Modeling and User-Adapted Interaction*, *21*(1–2), 99–135.

Nathan, M. J. (1998). Knowledge and situational feedback in a learning environment for algebra story problem solving. *Interactive Learning Environments*, *5*, 135–159.

Nathan, M. J., Kintsch, W., & Young, E. (1992). A theory of algebra-word-problem comprehension and its implications for the design of learning environments. *Cognition and Instruction*, *9*(4), 329–389.

National Research Council. (2012). *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas*. Washington, DC: National Academies Press.

Neumann, E. K., Feurzeig, W., & Garik, P. (1999). An object-based modelling tool for science inquiry. In W. Feurzeig & N. Roberts (Eds.), *Modelling and simulation in science and mathematics education* (pp. 138–148). New York: Springer.

Novak, J. D., & Canas, A. J. (2008). *The theory underlying concept maps and how to construct and use them*. Pensacola, FL: Florida Institute for Human and Machine Cognition.

Obayashi, F., Shimoda, H., & Yoshikawa, H. (2000). Construction and evaluation of a CAI system based on "Learning by teaching" to Virtual Student. *Transactions of Information Processing Society of Japan*, *41*(12), 3386–3393.

Paige, G., & Simon, H. A. (1966). Cognitive processes in solving algebra word problems. In B. Kleinmuntz (Ed.), *Problem solving: Research, method and theory* (pp. 51–119). New York: Wiley.

Palinscar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction*, *1*, 117–175.

Papert, S. (1980). *Mindstorms*. New York: Basic Books.

Pareto, L., Arvemo, T., Dahl, Y., Haake, M., & Gulz, A. (2011). A teachable-agent arithmetic game's effects on mathematics understanding, attitude and self-efficacy. In G. Biswas & S. Bull (Eds.), *Proceedings of artificial intelligence in education* (pp. 247–255). Berlin: Springer.

Penner, D. E. (2001). Cognition, computers and synthetic science: Building knowledge and meaning through modelling. *Review of Research in Education*, *25*, 1–37.
Pfundt, H., & Duit, R. (1998). *Bibliography: Students' alternative frameworks and science education* (5th ed.). Kiel, Germany: Institute for Science Education.
Ploetzner, R., & VanLehn, K. (1997). The acquisition of informal physics knowledge during formal physics training. *Cognition and Instruction*, *15*(2), 169–206.
Quinn, J., & Alessi, S. M. (1994). The effects of simulation complexity and hypothesis-generation strategy on learning. *Journal of Research in Computing in Education*, *27*(1), 75–92.
Quintana, C., Reiser, B. J., Davis, E. A., Krajcik, J., Fretz, E., Duncan, R. G., … Soloway, E. (2004). A scaffolding design framework for software to support scientific inquiry. *Journal of the Learning Sciences*, *13*(3), 337–386.
Ramachandran, S., & Stottler, R. (2003). A meta-cognitive computer-based tutor for high-school algebra. In D. Lassner & C. McNaught (Eds.), *Proceedings of world conference on educational multimedia, hypermedia and telecommunications 2003* (pp. 911–914). Chesapeake, VA: AACE.
Razzaq, L., Mendicino, M., & Heffernan, N. T. (2008). Comparing classroom problem-solving with no feedback to Web-based homework assistance. In B. P. Woolf, E. Aimeur, R. Nkambou, & S. P. Lajoie (Eds.), *Intelligent tutoring systems: 9th international conference, ITS2008* (pp. 426–437). Berlin: Springer.
Reif, F., & Scott, L. A. (1999). Teaching scientific thinking skills: Students and computers coaching each other. *American Journal of Physics*, *67*(9), 819–831.
Repenning, A., Ioannidou, A., & Zola, J. (2000). AgentSheets: End-user programmable simulations. *Journal of Artificial Societies and Social Simulations*, *3*(3). Retrieved from http://jasss.soc.surrey.ac.uk/3/3/forum/1.html
Schecker, H. (1993). Learning physics by making models. *Physics Education*, *28*, 102–106.
Schwartz, D. L., Blair, K. P., Biswas, G., Leelawong, K., & Davis, J. (2008). Animations of thought: Interactivity in the teachable agent paradigm. In R. Lowe & W. Schnotz (Eds.), *Learning with animations: Research and implications for design* (pp. 114–141). Cambridge: Cambridge University Press.
Schwartz, D. L., Chase, C., Chin, D. B., Oppezzo, M., Kwong, H., Okita, S. Y., … Wagster, J. (2009). Interactive metacognition: Monitoring and regulating a teachable agent. In D. J. Hacker, J. Dunlosky, & A. C. Graesser (Eds.), *Handbook of metacognition in education* (pp. 340–358). New York: Taylor & Francis.
Schwarz, C. V., Reiser, B. J., Davis, E. A., Kenyon, L., Acher, A., Fortus, D., … 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.
Schwarz, C. V., & White, B. Y. (2005). Metamodeling knowledge: Developing students' understanding of scientific modeling. *Cognition and Instruction*, *23*(2), 165–205.
Segedy, J. R., Kinnebrew, J. S., & Biswas, G. (2012). *Supporting student learning using conversational agents in a teachable agent environment*. Paper presented at the Proceedings of the 10th International Conference of the Learning Sciences, Sydney, Australia.
Sherin, B. L. (2006). Common sense clarified: The role of intuitive knowledge in physics problem solving. *Journal of Research in Science Teaching*, *43*(6), 535–555.
Singley, M. K., & Anderson, J. R. (1989). *The transfer of cognitive skill*. Cambridge, MA: Harvard University Press.
Steed, M. (1992). Stella, a simulation construction kit: Cognitive process and educational implications. *Journal of Computers in Mathematics and Science Teaching*, *11*, 39–52.
Sterman, J. D., & Booth Sweeney, L. (2002). Cloudy skies: Assessing public understanding of global warming. *System Dynamics Review*, *18*(2), 207–240.
Stratford, S. J. (1997). A review of computer-based model research in precollege science classroom. *Journal of Computers in Mathematics and Science Teaching*, *16*(1), 3–23.
Swaak, J., van Joolingen, W. R., & de Jong, T. (1998). Supporting simulation-based learning: The effects of model progression and assignments on definition and intuitive knowledge. *Learning and Instruction*, *8*(3), 235–252.
Tan, J., & Biswas, G. (2006). The role of feedback in preparation for future learning: A case study in learning by teaching environments. In M. Ikeda, K. Ashley, & T.-W. Chan (Eds.), *Intelligent tutoring systems: 8th international conference, ITS 2006* (pp. 370–381). Berlin: Springer-Verlag.

Tan, J., Biswas, G., & Schwartz, D. L. (2006). Feedback for metacognitive support in learning by teaching environments. In R. Sun & N. Miyake (Eds.), *Proceedings of the 28th annual meeting of the cognitive science society* (pp. 828–833). Mahwah, NJ: Erlbaum.

Tan, J., Skirvin, N., Biswas, G., & Catley, K. (2007). Providing guidance and opportunities for self-assessment and transfer in a simulation environment for discovery learning. In D. S. McNamara & J. G. Trafton (Eds.), *Proceedings of the 29th annual meeting of the cognitive science society* (pp. 1539–1544). Austin, TX: Cognitive Science Society.

Tan, J., Wagster, J., Wu, Y., & Biswas, G. (2007). Effect of metacognitive support on student behaviors in learning by teaching environments. In R. Luckin, K. R. Koedinger, & J. Greer (Eds.), *Proceedings of the 13th international conference on artificial intelligence in education* (pp. 650–652). Amsterdam: IOS Press.

Teodoro, V. D., & Neves, R. G. (2011). Mathematical modeling in science and mathematics education. *Computer Physics Communications*, *182*, 8–10.

Thompson, K., & Reimann, P. (2010). Patterns of use of an agent-based model and a system dynamics model: The application of patterns of use and the impacts on learning outcomes. *Computers and Education*, *54*, 392–403.

Treagust, D. F., Chittleborough, G., & Mamiala, T. (2002). Students' understanding of the role of scientific models in learning science. *International Journal of Science Education*, *24*(4), 357–368.

VanLehn, K., Burleson, W., Chavez Echeagaray, M.-E., Christopherson, R., Gonzalez Sanchez, J., Hastings, J., … Zhang, L. (2011). *The affective meta-tutoring project: How to motivate students to use effective meta-cognitive strategies*. Paper presented at the 19th International Conference on Computers in Education, Chiang Mai, Thailand.

VanLehn, K., & van de Sande, B. (2009). Acquiring conceptual expertise from modeling: The case of elementary physics. In K. A. Ericsson (Ed.), *Development of professional expertise: Toward measurement of expert performance and design of optimal learning environments* (pp. 356–378). Cambridge: Cambridge University Press.

Wagster, J., Tan, J., Biswas, G., & Schwartz, D. L. (2007). *How metacognitive feedback affects behavior in learning and transfer.* Paper presented at the 13th International conference on Artificial Intelligence in Education: Workshop on Metacognition and Self-Regulated Learning in ITSs, Marina del Rey, CA.

Wagster, J., Tan, J., Wu, Y., Biswas, G., & Schwartz, D. L. (2007). Do learning by teaching environments with metacognitive support help students develop better learning behaviors? In D. S. McNamara & J. G. Trafton (Eds.), *Proceedings of the 29th annual meeting of the cognitive science society* (pp. 695–700). Austin, TX: Cognitive Science Society.

Weld, D. S. (1983). *Explaining complex engineered devices*. Cambridge, MA: Bolt, Beranek and Newman, p. 50.

Wheeler, J. L., & Regian, J. W. (1999). The use of a cognitive tutoring system in the improvement of the abstract reasoning component of word problem solving. *Computers in Human Behavior*, *15*, 243–254.

White, B. Y. (1984). Designing computer games to help physics students understand Newton's Laws of Motion. *Cognition and Instruction*, *1*(1), 69–108.

White, B. Y. (1993). ThinkerTools: Causal models, conceptual change and science education. *Cognition and Instruction*, *10*(1), 1–100.

White, B. Y., & Frederiksen, J. R. (1990). Causal model progressions as a foundation for intelligent learning environments. *Artificial Intelligence*, *42*, 99–157.

Yaron, D., Karabinos, M., Lange, D., Greeno, J. G., & Leinhardt, G. (2010). The chemcollective–virtual labs for introductory chemistry courses. *Science*, *328*(5978), 584–585.