Learning How to Construct Models of Dynamic Systems: An Initial Evaluation of the Dragoon Intelligent Tutoring System

Kurt VanLehn, Jon Wetzel, Sachin Grover, and Brett van de Sande

Abstract—Constructing models of dynamic systems is an important skill in both mathematics and science instruction. However, it has proved difficult to teach. Dragoon is an intelligent tutoring system intended to quickly and effectively teach this important skill. This paper describes Dragoon and an evaluation of it. The evaluation randomly assigned students in a university class to either Dragoon or baseline instruction that used Dragoon as an editor only. Among students who did use their systems, the tutored students scored reliably higher ($p < .021$, $d = 1.06$) on the post-test than the students who used only the conventional editor-based instruction.

Index Terms—Intelligent tutoring systems, educational simulations

1 Introduction

The major issue addressed here is teaching students how to construct models of natural and man-made dynamic systems. Some terms that will be used here are: A *system* is part of the real world, and a *dynamic* system is one that changes over time. A *model* is an expression written in formal *modeling language*. The *behavior* of the system over time should match the *predictions* of the model. While some models can have their predictions generated by hand calculation, it is more common today to have computers do the calculations. Most models have a set of *parameters*, which are constant with respect to the time course of the system, but the values can be changed by users of the model. *Constructing* a model means developing a sufficient understanding of the system that one can write a model of it in the modeling language. *Exploring* a model means manipulating the parameter values of a given model and studying how these changes affect the model's predictions. *Modeling* refers to either constructing a model or exploring a given model. Our tutor system, Dragoon, supports both model construction and model exploration.

Modeling of dynamic systems is taught in the university at two levels. For students with strong mathematical backgrounds, it is taught using partial differential equations and MATLAB [1] or similar systems. It is typically a required topic in mechanical, industrial, electrical and other engineering degrees. For students with less advanced mathematical backgrounds, system dynamics modeling is taught with Stella [2], Vensim [3], Powersim [4] or similar systems, and it is typically a required course for some business and social science degrees.

In high school and earlier grades, modeling is prominent in recently developed standards. In the Next Generation Science Standards (http://www.nextgenscience.org), only eight scientific practices are threaded throughout the standards, and "developing and using models" is one of them. Of the 71 high school science performance expectations, 15 start with "develop a model of ..." or "Use a model to...." For example, HS-ESS2–6 is "Develop a quantitative model to describe the cycling of carbon among the hydrosphere, atmosphere, geosphere and biosphere." In the Common Core State Mathematics Standards [5], modeling is one of eight mathematical practices. The content topics taught at the high school level are marked with a star when modeling is involved (CCSSO, 2011). 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 percent are starred and thus involve modeling.

The problem addressed here is developing instruction that helps students learn how to construct models of dynamic systems. The focus is on both high school students and college students that have little more than a high school mathematics background (i.e., some algebra).

Fortunately, this research problem has enjoyed nearly 30 years of work [6], [7]. In a recent review of educational applications of system dynamics model construction, VanLehn [8] defined a multi-dimensional classification of research problems. Four of the major dimensions were:

- What modeling language?
- Model construction or model exploration?
- How were the systems presented to students?
- What were students intended to learn?

- *K. VanLehn, J. Wetzel, and S. Grover are with the School of Computing, Informatics and Decision Systems, Arizona State Unviersity, Tempe, AZ 85281. E-mail: {kurt.vanlehn, jon.wetzel, sgrover6}@asu.edu.*
- *B. van de Sande is with the Pearson Education, 3075 W Ray Rd #200, Chandler, AZ 85226. E-mail: brett.vandesande@pearson.com.*

Manuscript received 28 Jan. 2015; revised 2 Nov. 2015; accepted 16 Dec. 2015. Date of publication 7 Jan. 2016; date of current version 16 June 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TLT.2016.2514422

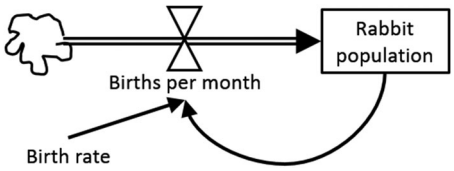

Fig. 1. Stock and flow model of a rabbit population.

This introduction begins by locating this research project along each dimension, then reviewing prior research that also falls into that cell of the multi-dimension classification.

1.1 Classification of the Research to be Presented

For modeling dynamic systems, the two most widely used modeling languages in K-12 education are graphical system dynamics modeling and agent-based modeling. A *system dynamics model* is essentially a set of coupled ordinary differential equations. However, Stella [2] and other model editors hide some of the mathematical details from the user by using a graphical "stock and flow" notation, which is described in the next paragraph. An *agent-based model* is a set of programs, one per agent, that control the agents' appearance, movement and other properties. NetLogo (https://ccl.northwestern.edu/netlogo/) is currently the most widely used language for agent-based modeling, although there are others [9], [10], [11], [12], [13], [14]. Interesting macro-level behavior often emerges from simple micro-level agent programs; this property is called emergence [15], [16]. Although emergence has excited both scientists and science educators, system dynamics models are still widely used professionally and are often simpler than agent based models. Thus, this project has focused on graphical system dynamics modeling languages.

As an introduction to the basic concepts, Fig. 1 shows a model in stock-and-flow notation for this system: "A rabbit population starts with 100 rabbits. Assuming there are no deaths and that the population increases by 10 percent per month, show the population over 12 months." The rectangular node is called a "stock" and represents a quantity that is the integral of its inputs. The inputs to a stock are shown with double-line arrows called "flows." The icon in the middle of the flow, which is intended to look like a valve, is actually a node representing the value of the flow. Clicking on it (or any other node) opens the node editor, which in this case would show that the number of "births per month" is the product of "rabbit population" and "birth rate". Inputs to nodes other than stocks are shown with thin arrows, which is why "rabbits born per month" has two arrows coming into it. Birth rate is an "auxiliary" node, which means it is neither a stock nor a flow. In general, auxiliary nodes can have any mathematical function inside them, but this particular node has only a constant (0.1) as its value.

Two common instructional activities are *model construction* and *model exploration* [6], [8], [17], [18], [19]. During a model construction task, students are given a *presentation* of the system and asked to construct a model of the system. During a model exploration task, students are given a model instead of constructing it. They manipulate the model's parameter values in order to see how its predictions change. With systems like Stella and NetLogo, the values of parameters can be manipulated by moving sliders, and the resulting predictions are displayed as graphs that change shape in real time as the user manipulates a slider. Often students are not shown the model itself, but only the sliders and graphs. The research presented here addresses only model construction and not model exploration.

Some common presentations of systems [8] are

- a short text [20]
- a set of documents [21], [22], [23], [24]
- a simulation of the system [11], [25], [26], [27], [28], [29]
- access to the real system and tools for measuring its properties [30].

For instance, students might be asked to construct a model of the population of Phoenix, Arizona given a short text such as "In 2010, the population of Phoenix was 1,445,632 and was increasing at 9.4 percent per year." Alternatively, students might be given as set of source documents, including census reports, population maps, etc. In the research presented here, the presentations are short texts.

There are many reasons for including model construction in the curriculum. Some of the major instructional objectives, discussed more thoroughly in [8], are:

- *Improved domain knowledge*. Sometimes students are expected to learn fundamental domain principles and concepts by constructing models of systems that involve them. For instance, physics students may be taught the principles of Newtonian dynamics by constructing models of falling blocks, cannonballs projected at angles and so on.
- *Improved understanding of a particular system*. Some systems are so important that students need to understand them deeply. For instance, earth science students might study global warming by constructing submodels then integrating them.
- *Understanding the role of models in science*. Model construction can help students appreciate the epistemology of models. Students can grapple with issues like accuracy, parsimony, under-determination, and so on [31], [32].
- *Improved model construction skill*. When model construction is treated as a cognitive skill, system presentations usually contain the information needed for constructing a model, and students are not expected to retain much information about the system and the domain principles underlying it. The focus is on becoming skilled in constructing models regardless of the content of the system.

This research focuses on just the last instructional objective: improving student's model construction skills. Such skill is arguably a pre-requisite for using model construction for the other purposes listed above [25], [33].

1.2 Prior Work on Teaching Model Construction

When Stella pioneered a graphical notation for models [34], many educators started to explore its instructional potential for K-14 instruction. Classroom studies of system dynamics model construction started with case studies then rapidly scaled up to large professional development projects. The STACIN project [19] and the CC-STADUS and CC-SUSTAIN projects [35] taught over 260 high school science and math teachers how to use Stella. One major finding was that the teachers constructed most models, which students

then used for model exploration [18]. On the few occasions when students were asked to construct models, the models were quite simple and the systems were presented by short, highly explicit texts. Teachers reported that it was difficult and time consuming to teach anything beyond basic model construction skills.

Perhaps in response to these difficulties, several research projects developed qualitative or partially qualitative modeling languages [30], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51]. Although there have been some promising results, there is as yet no experimental evidence that these qualitative languages are easier to learn than the quantitative graphical ones.

Seeking perhaps to find the underlying bottleneck in learning, several projects developed simple tests (often called concept inventories) for measuring system thinking, and used them to show that decision makers often did surprisingly poorly on them [52], [53], [54], [55], [56], [57]. Fortunately, even a small amount of instruction was sufficient to overcome at least some of the deficits, at least with some students [58].

Several projects have tried to understand how different methods for displaying models and predictions affect students' understanding and learning [27], [28], [59], [60]. The results uncovered no great flaws in stock-and-flow display or the display of model predictions as graphs.

Some studies have undertaken detailed qualitative analyses of students' behavior as they tried to construct models [18], [61], [62], [63], [64], [65]. Many of the studies contrasted the behaviors of successful and unsuccessful modelers. Alessi [18] provided a sample of the observed issues:

- Students tend to confuse stocks with flows.
- Students try to incorporate the formulas of previous science and math classes (which they often do not fully understand) instead of doing true system analysis.
- When models do not work correctly, students include fudge factors. Fudge factors are formulas, constants, or logical conditions designed to artificially fix the problem, and not to realistically model the system.
- Students fail to test their models well, so the models tend to work only for common conditions.
- Students confuse flows with causality.
- Students create models that are unnecessarily complex and abstract.
- Students try to copy and adapt models from instructors or textbooks instead of thinking through the phenomenon and generating their own models from scratch.
- Students engage in trial-and-error modifications in the hope that the results will come out right.
- Students create initial models with too many components when they should start with a simple model and add complexity as needed.
- Students ignore the units of variables and, as a result, combine variables that have different units.

Bravo et al. [25] conducted a formative evaluation of a tutoring system for model construction. The system, CoLab, provided feedback and hints when students asked for help as they constructed a model. The evaluation uncovered several issues (e.g., students failed to make any changes in their model about 20 percent of the time after receiving advice) but did not measure learning gains nor compare the tutoring system to another form of instruction.

TABLE 1
Methods for Scaffolding Model Construction

Tutoring
* Feedback/hints on the model [20], [25], [66]
* Feedback/hints on the student's process (meta-tutoring) [20], [24], [67], [68]
Concrete articulation strategy [69], [70], [71]
Decomposition into subsystems [69], [72], [73]
* Reflective debriefings [74], [75], [76]
Answering student questions [24], [37], [77], [78], [79]
Clarifying the modeling language
* Notation for model [26], [27], [28], [30], [41], [44], [71], [80]
* Grounding the symbols [30], [41], [51]
* Facilitating comparing the model's predictions to the system's behavior [30], [49], [81], [82]
Students explaining their model [30], [69], [83]
Gradually increasing complexity
Hiding model features [30]
Qualitative-first model construction [38], [48], [65], [84]
* Model progressions [85], [86], [87], [88], [89], [90]
Other scaffolding
Teachable agents and reciprocal teaching [66], [91], [92], [93], [94]
Mental execution of models [48], [60]
Test suites [66]
* Generic models [44], [80], [95]
Gamification [21]

* *indicates ones used by Dragoon.*

In summary, there is ample evidence that students find it difficult to learn how to construct models of dynamic systems. However, only a few methods of scaffolding the learning have been studied so far, such as using qualitative math instead of ordinary math, and modifying the appearance of the modeling language.

Nonetheless, there are many potentially beneficial methods that can be drawn from the wider literature, which includes agent-based modeling and model exploration. VanLehn [8] identified several methods for helping students learn how to construct models. Table 1 lists them and indicates which are used by Dragoon and discussed in the next section.

2 Dragoon and its Instruction

Prior to the study reported here, we conducted six studies in summer school classes for high school students, three studies in a university sustainability class, and one study in a university informatics class [96], [97], [98], [99], [100], [101]. Although some of these studies were formal tests of hypotheses, all of them also served as formative evaluations. That is, from observations, interviews, log files and screen-capture videos, we inferred ways that the Dragoon system and our instruction could be improved. This section describes the major lessons learned, and how they relate to the experience of other researchers.

2.1 The Modeling Language

The initial version of Dragoon (which was called AMT) used the stock-and-flow notation. During focus groups with high school students, we found that they didn't understand the notation well even after using Dragoon for two hours. They

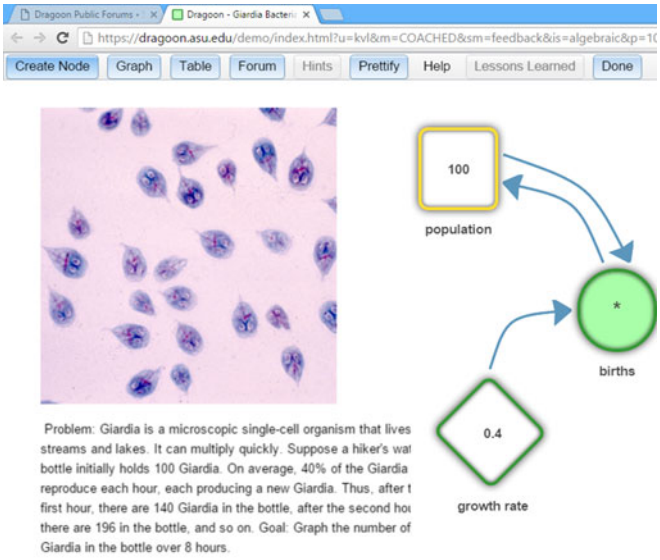

Fig. 2. Dragoon screen.

understood that the thin arrows indicate inputs, but they didn't understand why the double arrows, which were also inputs, were drawn differently. They thought the cloud-like icon was a node, but it isn't, so they thought it should be eliminated. They saw no point in having auxiliary nodes for constants, since they could put the constants inside expressions of other nodes. Several students suggested that we use function machine notation.

Simplifying the notation of system dynamics models is not popular, given how entrenched the stock and flow notation is. However, the Model-It project did use a novel notation [30]. The modeling language was similar to concept map notation in that it had just one kind of arrow and one kind of node. However, when the user clicked on a link to edit it, it turns out that there were two kinds of links: Immediate and Rate, which correspond to the thin arrows and the double arrows of the stock-and-flow notation, respectively [see the Appendix of 102]. However, the Rate links (and hence the implicit stock nodes) were not often used. In one of the final studies of Model-It, only 26 percent of the students used any Rate links at all [30, p. 112]. Our instructional goals placed more emphasis on dynamic systems, so we highlighted the necessary distinction between stocks and non-stocks, but dispensed with the superfluous distinction between flows (or Rate links) and regular links.

Dragoon's current notation, which is illustrated in Fig. 2, uses function machine notation in that an arrow always represent an input. However, Dragoon uses three three kinds of nodes:

- *Accumulators*: Stock nodes (rectangles) were renamed accumulators; the initial value of the accumulator is shown inside it. The value of an accumulator at time T + 1 is its value at time T plus the value of its inputs at time T.
- *Parameters*: A parameter (a diamond) is a constant whose value is controlled by a slider.
- *Functions:* Function nodes (circles) can in principle have any function inside; we have implemented arithmetic operations as well as some common functions.

The focus group students wanted to be able to see all the details of their models without having to open any of the nodes. We considered displaying the mathematical expressions inside the nodes, as is done with function machine notation. However, function machines tend to use short, one-character names for quantities, whereas a good practice for modeling is to use meaningful names such as "rabbit population." These long names thwart displaying the expression inside the node. In order to satisfy the students' wish for transparency on at least some simple models, the notation uses the following conventions.

- When an input to an accumulator should be subtracted from its value, then that input's arrowhead has a little circle with a negative sign inside it.
- If a function is a sum of its inputs, then a + appears inside it. As with accumulators, if an input is negated before being included in the sum, the input's arrowhead has a minus sign inside a small circle.
- If a function is a product of its inputs, then a * appears inside it. If any of the inputs is inverted before being included in the product, then a small circle with / inside it appears at the input's arrowhead.

These conventions allow most models to be written so that all their mathematical details can be inferred from their appearance. Users can write models that include more complex mathematical expressions inside the function nodes, but such nodes will display nothing inside their circle, so the students will need to click on the node to see what it has inside it.

2.2 Grounding: Choosing Names for Nodes

In commercial system dynamics editors, the user chooses names for the quantities. However, Dragoon needs to know the quantity that such a name refers to so that it can determine if the user's definition is correct. For instance, if Dragoon allowed students to type in their own names, and the students chose "R," then Dragoon would need to guess whether "R" refers to "rabbit population," "rate of rabbit births" or some other quantity. When Bravo et al. [25] let students type names for nodes, only 74 percent of the student names were correctly matched to system quantities. This is a general problem, called the *grounding* problem, that occurs when two participants in a dialogue struggle to arrive at the same meaning for a term [103].

Model editors have used several methods for reducing grounding problems, but none are perfect (see [8], Section 8.2.2 for discussion). For instance, Model-it [30], ModelingSpace [51] and Vmodel [41] had 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. However, when students were given a list of terms such as "water", "tanks", "water pressure" and "water level in a tank", they had trouble identifying which terms were variables [48].

The initial version of Dragoon attempted to avoid this by giving the student nodes that had appropriate names but were otherwise undefined. For our rabbit population example, it would give them nodes labelled "rabbit population," "births per month" and "birth rate." Unfortunately, despite all attempts to make the names crystal clear, students still misinterpreted them regularly. For instance, some students

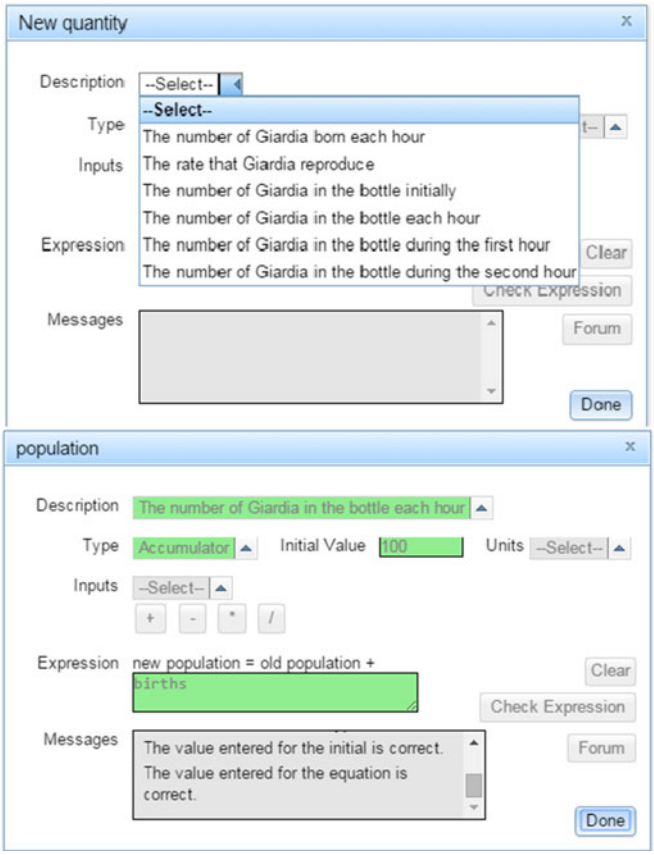

Fig. 3. Initial and final state of the node editor.

would build a model just like the one in Fig. 1, but the flow would be named "birth rate" and the constant-valued node would be named "births per month." Interviews with students uncovered the fundamental problem: They would only read the first word or two of the name, and if it seemed at all appropriate, they would use it without looking to see if there was a node with an even better name.

Our next attempt at solving the grounding problem put all the names in a hierarchical menu, so that students would be forced to choose among similar names. For the rabbit problem, the menu was:

- Rabbit population
- Birth . . .
  - rate
  - per month

When the menu first appeared for naming a node, only "rabbit population" and "birth" would be visible. Students needed to click on "birth" which would then display the rest of the node names and allow them to pick one. This significantly reduced the grounding problems, but students complained about having to navigate the hierarchical menus, particularly when there were only a few possible node names, as in our rabbit example. Thus, we switched to listing the names alphabetically in a flat, non-hierarchical menu. This made the repetition of words at the beginning of the names quite salient. This appeared to work as well as the hierarchical menus, and it was faster to navigate.

In order to encourage students to choose names carefully, some problems have menus that contain names that are not necessary in the model. For instance, in a model to

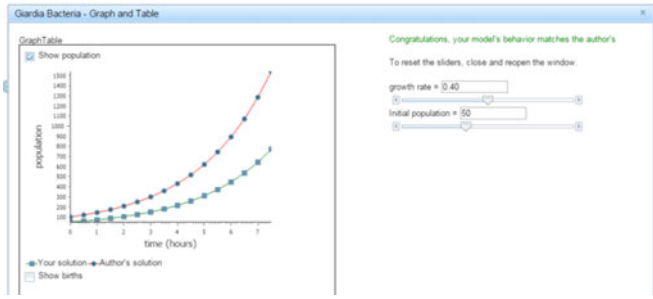

Fig. 4. Dragoon's graphs and sliders.

be discussed in a moment, only three nodes are necessary but six names appear in the menu (see top pane of Fig. 3).

2.3 Constraining Node Editor Actions

Like all system dynamics editors, Dragoon's users define a node by opening a pop-up node editor, which is essentially just a form that needs to be filled in. Although our initial node editor gave students considerable freedom in the order in which they filled out the form, this freedom often led to confusion. The current system can require that the blanks in the form be filled in a specific order. It enforces this ordering by enabling one blank at a time (see Fig. 3).

2.4 The Display of Predictions

Like all modern system dynamics editors, Dragoon can plot a graph of the values of a selected node over time. A common instructional problem is that students often do not notice when their model's predications fail to match the system's behavior [18]. Even if a graph of the value of a system quantity is given to the students as part of the system description, students often mistakenly think that their model's graph matches the given graph.

In order to get students to notice when their model's predictions do not match the given behavior, Dragoon puts the system's behavior and the model's predictions on the same graph. For instance, if the student mistakenly put the value of "birth rate" as 10 instead of 0.1, then Dragoon would display the graph of rabbit population shown in Fig. 4. The green line with square points indicates user's model's prediction, and the red line with circular points is the system's behavior. This instructional method has been used by other systems as well [25], [104].

2.5 Tutorial Feedback

Dragoon has two major modes, author mode and student mode. In author mode, the user enters not only a model but also a system description, time ranges, time units, and other details. In student mode, the screen opens showing a description of the system to be modelled. The student enters and tests a model with help from Dragoon. The amount of help given to the student is determined by the scaffolding mode:

- *Editor* mode: The student gets no help. Dragoon acts like Stella, Vensim or any other model editor, except that students select the name of a node from a list.
- *Test* mode: Whenever the student asks for a plot of a node's value, both its model values and the author's model's value are displayed in the same graph

(see Fig. 4). This is the only feedback that the student gets.

- *Immediate feedback* mode: Whenever the student enters something in the node editor, the entry turns green, red or yellow. It turns green if the entry matches the corresponding element of the author's model of the system (see Fig. 3), and red otherwise. However, if the student has already made several mistakes on this entry, then instead of turning red, the student's entry is replaced by the author's entry and colored yellow. This is sometimes called a "bottom-out hint" in the tutoring literature [105]. Such help prevents frustration and offers an opportunity to learn.
- *Coached* mode: The student receives not only immediate red, green and yellow feedback on each step, but also receives process advice about what steps to do next [20], [68], [106] and how to use the feedback appropriately [107].

Immediate feedback mode was included because it has been used successfully by other systems [20], [25], [105]. However, a common issue for immediate feedback is help abuse: Instead of trying hard to make an entry, students repeatedly either ask for help or make errors deliberately until the tutoring system tells them what to enter [108]. A variety of methods for reducing help abuse have been tried, but none have been a stunning success [107], [108], [109], [110]. In Dragoon, help abuse occurs when a student makes enough deliberate errors that Dragoon fills in the entry and colors it yellow. To discourage making errors deliberately, when even one entry in a node's definition is yellow, then the perimeter of the node's image is yellow. Students figure this out quickly, and it seems to reduce help abuse considerably. In order to further encourage students to think hard before making an entry, Dragoon will color a node solid green if the student defines it without making a single error (Fig. 2). Although these simple ideas for reducing help abuse seem to work, it would be simple and interesting to run an experiment comparing help abuse rates with and without the node coloring policies.

Whereas immediate feedback mode is feedback on the students' work *product*—the model being produced, coached mode is feedback on the students' *process*—what they choose to do next. The basic idea is backward chaining [111], but Dragoon explains it more simply, with these feedback messages:

1. Although the quantity you've picked is in the author's model, you should follow the Target Node Strategy, which says you should start by defining a node for a quantity that the problem asks you to graph or focus on, then define nodes for its inputs, and then define nodes for their inputs, etc. That way, every node you create is an input to some node.
2. Please follow the Target Node Strategy. That is, finish any incomplete node (triangle or dashed border) or, if there are no incomplete nodes, select a quantity the problem asks you to graph or focus on.
3. It is too soon to work on this node. Please follow the Target Node Strategy.

Dragoon keeps count of how many times a particular type of feedback has been presented, so the three messages above are presented for the first, second and subsequent times that this piece of feedback is needed. (All feedback messages have a similar three-fold presentation.) If the student chooses to ignore Dragoon's advice and create a premature node, then it turns blue but otherwise functions just like a green ("correct") node. In three studies that compared Dragoon with and without coached mode, students learned more with coached mode [20].

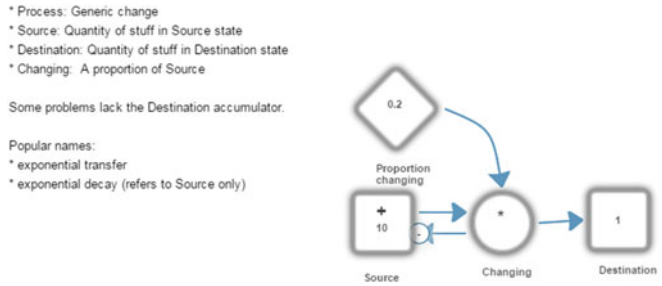

Fig. 5. Schema for exponential decay.

2.6 Schemas and a Theory of Skill Acquisition

Generic solutions to problems are often called *schemas* [112]. Mathematics textbooks often present many problems that are instances of the same generic schema. Mayer [113] analyzed 10 algebra textbooks, and found that they contained 1,097 word problems. However, there were only 93 schemas. An example of an algebra schema is the Overtake schema, wherein one vehicle starts later than another vehicle but overtakes it as they follow the same path.

A simple but incomplete theory of skill acquisition is that schemas are the unit of knowledge that students acquire when they practice problem solving. Thus, if students practice many Overtake problems and few Round Trip problems, then they will make fewer errors on Overtake problems than Round Trip problems.

Some system dynamics instruction includes general models of linear change, exponential change, logistic change and other common systems [80]. Dragoon includes such generic models as examples that students can refer to when they are trying to build a model of a specific system. Fig. 5 shows the example of exponential decay.

Schema theory is incomplete in several ways. First, some problems require more than one schema. For instance, one Dragoon system is a retirement account whose only income is interest (exponential growth schema) and whose only withdrawals are a constant amount per month (linear decay schema). Students need to develop skill in dividing complex problems into parts such that each part can be solved by applying a known schema. Second, when using systems like Dragoon that provide feedback on the results, students must learn how to debug solutions that are flagged as incorrect. Third, it may be that when students have mastered many schemas, then they develop an even more general skill in problem solving so that they can easily solve problems that don't match any schema that the student knows. For instance, a student who has mastered 50 of Mayer's schemas may be able to easily solve problems that match the other 43 schemas.

Although schema theory is incomplete, it does provide a workable solution for organizing problem solving instruction. That is why many math textbooks and some system

dynamics textbooks explicitly mention schemas (without calling them "schemas," which is Greek to most students). That is why Dragoon presents schemas and discusses them as solutions to problem, but it refers to them as "generic models" or "generic change processes."

2.7 Reflecting on a Solved Problem

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?" [30], [114], [115], [116]. One can also ask qualitative questions such as, "What would be different if the mass were increased substantially?" [74], [75], [76]. Katz et al. [75] found that the content of the reflection mattered more than the questioning. That is, they found that merely printing a succinct text with reflection points was marginally more effective than a guided dialogue that elicited the same information from the student.

Given these results, Dragoon uses only text for reflection, and does not use questions or dialogue. More specifically, whenever a student clicks on the Done button (see Fig. 2) and Dragoon agrees that the student is finished with the problem, then it presents a short text written by the author.

2.8 Model Progressions

A model progression is a carefully designed sequence of model construction problems that is intended to optimize students' learning by moving from simple problems to complex ones [115]. Three dimensions of increasing complexity are common:

- The models start simple and become increasingly complex [85], [86], [87], [88], [89], [90].
- The presentations start simple and become increasingly complex.
- The model editor starts with a simple set of features, and more sophisticated features are enabled gradually [30], [38].

Because Dragoon's modeling language is so simple, only the first two ordering principles are used.

Our current definition of "model complexity" is based on the set of schemas required to solve the model construction problem. In particular, each new problem either (1) repeats a problem structure used earlier, (2) introduces a new schema, or (3) is a novel combination of familiar schemas. The progression never introduces two new schemas, nor introduces a schema and a novel combination at the same time. Toward the end of the sequence, problems that cannot be solved with familiar schemas are presented. We believe this ordering principle is more favorable to learning than our old principle, which ordered problems based on the number of nodes in the model.

Next, we describe the five problem sets used in the experiment, detailing how both the models and the presentation increase in complexity through each problem set.

In problem set 1, all the problems involved applying just one schema. Within that set of problems, the following ordering conventions were used. The first time a schema was used for solving a problem, the problem showed a picture of the schema and printed the bulleted analysis that goes along with it (see Fig. 6). The second time a schema was used, the bulleted analysis was still present, but the name of the schema and its image were absent. The third and subsequent times a schema was used in a problem, even the bulleted analysis was absent. Thus, the presentations of problems began with scaffolding that was rapidly faded out.

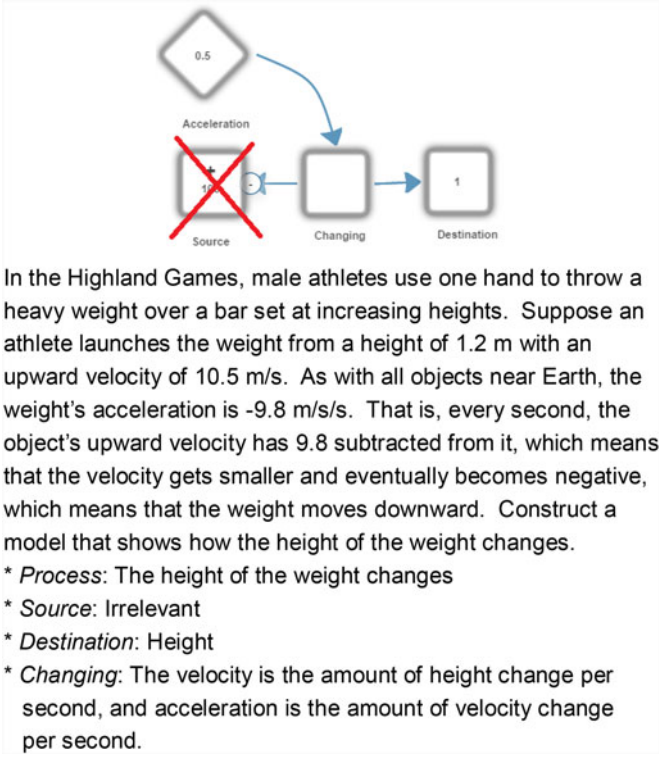

Fig. 6. First problem to use the acceleration schema.

In problem set 2, all the problems involved combining two or more schemas. The first time a combination was required, the bulleted analysis was part of the presentation, as in Fig. 6 but for both schemas. On the second and subsequent presentations of problems involving that particular schema combination, the bulleted analysis was absent.

In problem sets 3 and 4, all the problems involved system dynamics schemas combined with algebra schemas and irrelevant quantities. Although algebra schemas could have been taught in the same explicit way as system dynamics schemas, we did not do so because we assumed that most students would have mastered such schemas during high school. This assumption now seems false, as discussed later.

In problem set 5, the problems could not be solved by combining familiar schemas. They mostly involved classic system dynamics models, such as the Lotka-Volterra model of predator-prey ecosystems, the logistic model of capacity-limited growth, and Richardson's model of the arms race. These are complicated problems that sometimes appear late in college-level courses.

3 Evaluation

An evaluation of Dragoon was conducted in order to determine whether it was more effective than baseline instruction. The evaluation was conducted in a university class on modeling at a large southwestern university. The system dynamics module of the course lasted three weeks; the rest of the course addressed modeling with other languages. The course had taught system dynamics modeling construction for years

without the benefit of a tutoring system, and its instruction served as the baseline against which Dragoon was compared. Although the students in this course were required to have taken brief calculus, discrete math, probability and statistics before taking this course, their performance during the course suggested that only some of them had mastered the expected mathematics skills. Thus, although this research project is aimed at high school applications, only some of the students in this study had a high school-level math background; others were more advanced.

3.1 Design and Sample

The study was a two-condition, between-subjects experiment with a post-test but no pre-test. The Institutional Review Board approved the study as non-exempt, and 34 students volunteered to participate. Students knew that if one condition turned out to have a higher mean score than the other, the difference in the means would be added to the scores of the students in the lower-mean condition. They also received three extra credit points for participation.

The students were assigned to condition by stratified assignment. That is, after they were numbered by their score in the initial three modules of the course, even numbered students were assigned to one condition and odd number students were assigned to the other. Unfortunately, one student was accidently assigned to the wrong condition, so there were 16 students in the Tutor (experimental) condition and 18 students in the Editor (control) condition.

3.2 Design of the Experiment

Because the class emphasized developing skill in modeling, all modules in the course emphasized problem solving over lecture. A typical 75 minute class period had three activities: a quiz, a short lecture, and supervised problem solving. During the quiz, students solved a single problem similar to the ones they did on the preceding homework assignment. The homework assignments were not graded, so the main motivation for doing them was to score well on the quiz. During the supervised problem solving period, students began working on the next homework assignment while the instructor circulated among them providing help. Students were also encouraged to help each other during this time.

The system dynamics module was composed of six consecutive class meetings, with a homework assignment between each meeting. The first class meeting had no quiz, and a 30 minute lecture that introduced the basic concepts of system dynamics and Dragoon. The last class meeting was an exam. The remaining four class meetings had the format described above: quiz, short lecture and guided homework. The quizzes were worth five points each, and the exam was worth 50 points.

Students in both conditions of the experiment attended class together. Their homework assignments were identical, and everyone used Dragoon to do them. Section 2.8 describes the problem set sequence. The only difference was that students in the Tutor condition solved problems in Dragoon's coached, immediate feedback and test modes, whereas students in the Editor condition used Dragoon is its editor mode.

As had been done in previous years, students in the Editor condition had access to a PowerPoint slide deck learning aid. For each problem in a homework assignment, there were three slides in the deck that gave gradually more specific hints on solving the problem. The first slide indicated how many nodes of each type were needed in the model. The second slide presented the basic model structure without the mathematical details. The last slide presented the complete model. This three-slide learning aid was familiar to the students, as similar learning aids had been used during the first three modules of the course.

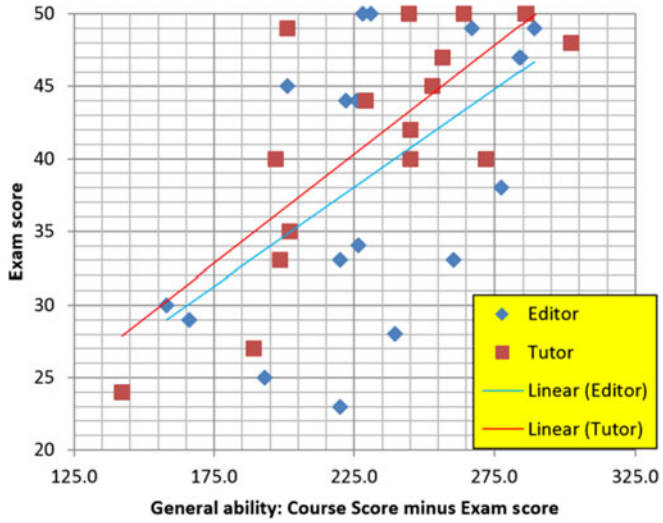

Fig. 7. Post-test versus general ability.

The only difference between the Editor condition and the instruction in earlier years was that the earlier years used a commercial system dynamics editor, Vensim, instead of Dragoon. Thus, the Editor condition was very similar to baseline instruction.

3.3 Measures

The main measure of performance was the students' score on the module exam, which served as an immediate post-test. No pretest was necessary because students were, as always in this class, unfamiliar with system dynamics model construction. The exam was scored by the course instructor, who was blind to the condition of the students. The exam consisted of three Dragoon problems (see Supplemental Materials, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TLT.2016.2514422). All were solved in editor mode, which was familiar to all students, as the quizzes were in editor mode. Supplementary measures included log data collected automatically by Dragoon and quiz scores.

Because students in this class typically had a variety of preparations and the class was required for some and elective for others, several analyses described below used course performance as a covariate in order to partially compensate for the varying abilities of the students. The course performance was calculated as the number of points scored in the whole course minus the points scored during the system dynamics module.

3.4 Results

Fig. 7 shows a scatterplot of the 34 participants. As the scatterplot suggests, the stratified assignment to conditions worked in that there was no difference between conditions

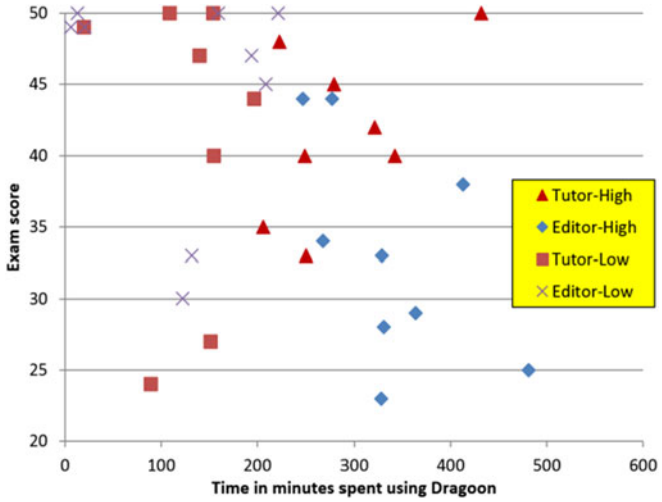

Fig. 8. Post-test versus time on Dragoon.

in mean *course* performance, which is our measure of general ability (two-tailed T-test, $p = 0.94$, $d = 0.03$). More importantly, and unfortunately, the mean *exam* score of the Tutor students (41.5) was not reliability different (two-tailed T-test, $p = 0.41$, $d = 0.29$) from the mean exam score of the Editor students (38.9). In an ANCOVA with exam score as the dependent variable and course performance as a covariate, the means of the conditions were still not reliably different ($p = 0.31$).

However, students varied in their usage of the system from 6 minutes to 491 minutes (mean: 219 minutes). In order to separate out those who actually used Dragoon, as either an editor or a tutoring system, from those who did not use it much, a median split was done. That is, each group was split into high and low duration users so that there were the same number of high duration users as low duration users. As one would expect, the two low duration groups, Tutor-Low ($N = 8$) and Editor-Low ($N = 9$), had nearly the same post-test score (41.4 versus 44.8, $p > .46$), as their scores reflected mostly their general abilities rather than their use of Dragoon. However, the mean of the Tutor-High group's ($N = 8$) exam scores (41.6) was reliably higher (two-tailed T-test, $p < .021$) than the mean of the Editor-High group's ($N = 9$) exam score (33.1), and the effect size was large ($d = 1.06$). Moreover, this difference was still observed when course performance was factored out in an ANCOVA ($p < .047$). This suggests that when students actually do their homework, then Dragoon's tutoring causes more learning than using Dragoon as an editor with hints from PowerPoint slides.

Fig. 8 shows a scatterplot of the four groups of students. The vertical axis is the exam score, but now the horizontal axis is the time spent using Dragoon, as extracted from the log files. Although the Tutor-High group spent less time using Dragoon (mean: 288 minutes) than the Editor-High group (337 min.), the trend was large but not reliable ($p = 0.19$, $d = 0.65$). Because the Tutor-High group was earning higher scores in less time than the Editor-High group, one would expect that their efficiencies were dramatically different. When efficiency is measured as the exam score divided by the time spent on Dragoon, the mean of the Tutor-High group (0.15) was significantly higher ($p < .025$) than the mean of the Editor-High group (0.10) with a large effect size ($d = 1.04$). More specifically the Tutor-High group earned 50 percent more exam points per unit time than the Editor-High group. These results suggest that the tutoring features of Dragoon were effective.

4 Discussion

The good news is that most of the students who used Dragoon appear to have mastered the cognitive skill of constructing system dynamics models. Of the eight students in the Tutor-High group, seven scored 80 percent or higher on the final exam. This was probably due to the tutorial features of Dragoon, because only two of the nine students who used the Editor version of Dragoon scored higher than 80 percent.

The scatterplot of Fig. 8 reveals a curious phenomenon. The data points in the upper left corner show that four students aced the exam even though they spent only 20 minutes or less using Dragoon. This phenomenon also occurred with earlier versions of Dragoon. During every trial with high school students in the summer school classes, one or two students would learn dramatically faster than the other ~35 students in the class. We are not sure why, but it seems that, for about 5 to 10 percent of the target population, learning how to construct system dynamics models is extremely easy. It appears that these students have already mastered some skill that transfers to the Dragoon context. Let us first consider some other studies using Dragoon, then return to the question of what this general and highly desirable skill might be. The next few sections are quite speculative, so some readers may wish to skip ahead to Section 4.3 "A Final Word."

4.1 Analytic Mastery versus Notational Mastery

Dragoon has been used in two instructional contexts that are quite different than the one discussed here. It has been used in six high school science classes, where the focus was primarily on learning about specific systems, such as the digestive system's regulation of energy intake, expenditure and storage [117]. Each class worked for several class periods on a worksheet that was specific to the system they were studying. Although the worksheets had them construct models with Dragoon, the models were described explicitly in text. For instance, if the model of Fig. 2 were described this way, the description would exclude the discussion of what Giardia is, as that would be covered earlier in the worksheet. However, it would describe each of the nodes required for the model:

Suppose a hiker's water bottle starts with 100 Giardia in it, and that this strain of Giardia grows at 40 percent per hour. Construct a model in Dragoon with:

- A parameter node that represents the growth rate, 0.40
- A function node that represents the number of Giardia born each hour. Its value is the growth rate times the current number of Giardia in the bottle.
- An accumulator node that represents the number of Giardia in the bottle each hour. Its initial value is 100. It has one input, which is the number of baby Giardia born each hour. Thus, its value grows by that amount each hour.

Such descriptions allow students to construct models of the appropriate systems, but they do not require the analytic skill that was taught during the study described here.

However, even given such explicit descriptions of models, students still have some initial difficulty operating Dragoon and building the models. It takes about 30 to 60 minutes for these students to become fluent with the Dragoon user interface and the three-node notation for models. Let us call this level of competence *notational mastery*, and use *analytic mastery* for the competence acquired by the best students in the study described in Section 3.

It was initially unclear whether the high school biology students could acquire a deep understanding of the systems they were studying given that they only had notational mastery. We feared that analytic mastery would be required, and thus science classes would have to include a six-class module on model construction as a pre-requisite for learning about their target system. However, our studies suggest that notational mastery suffices [117].

A second context for using Dragoon was a college class on sustainability where students used Dragoon for an end-of-semester project. Over four weeks, students worked in small groups to build models of the urban ecosystem of the city of Phoenix that would allow them to propose policies that were demonstrably sustainable. Students were given no information about Phoenix; they had to find all the information that they needed. As an example of the kinds of challenges that students' faced, consider creating a node for the average number of miles driven per person per day. One can find a value (27.30 miles, according to http://www.smartgrowthamerica.org/documents/phoenixsprawl.pdf), but it is difficult to find out what effects that value, and even more difficult to get numerical estimates of the size of those effects. We were initially worried that students would require analytic mastery in order to complete the project, so several hours of instruction on Dragoon were required before students started on their projects. As it turned out, most groups picked one person to do all the editing of the model. As mentioned earlier, about 5 to 10 percent of the target students seem to acquire analytic mastery rapidly, in less than 20 minutes. Moreover, the scatterplot of Fig. 8 suggests that a few hours with Dragoon suffice for many others to attain analytic mastery. Perhaps because every group had at least one person with analytic mastery, all groups developed impressively large, sophisticated models [118].

This sheds new light on the success of Dragoon's predecessor, Model-It [30], [119]. In several studies, students used Model-It in projects similar to the sustainability one. The Model-It students had to do literature research or gather data in order to determine how the target systems worked. Metcalf et al. [30, p. 109] remark that, "Tutorial materials introducing students to the software were kept to a minimum (less than 1 hour of class time)." This seems a remarkably short period of time for learning a system dynamics model construction tool, given the earlier studies showing how difficult it was for students to learn model construction [18], [35], [120]. However, the Model-It students always worked in small groups. Even if only some of the Model-It students achieved analytic mastery in one hour, every group may have had at least one student capable for constructing models easily.

Here is a speculative account that ties these facts together. Suppose that there is a general skill, labelled *analytic mastery* of model construction, which some students acquire quickly and others take at least several hours of practice to acquire. While analytic mastery is an important instructional objective in its own right, and clearly called for by the standards, it is not a pre-requisite for using model construction to learn about specific systems. Students can use model construction to learn about, say, stream ecology, by engaging in inquiry activities in small groups so that at least one member per group has analytic mastery, or by using explicit descriptions of the models to be constructed by individual students so that only notational mastery is required.

4.2 What is the Relationship of Well-Defined Model Construction to Systems Thinking?

Systems thinking is a valued construct that is not easily defined. Booth-Sweeney and Sterman [53] developed an assessment of system thinking that has students draw graphs of a system's behavior given graphs depicting inputs to the system. Although the assessment has been widely used [52], [54], [55], [56], [57], [58], [121], Hopper and Stave [122] questioned whether it reflected common practice in defining systems thinking. From a comprehensive review of 200 articles that studied instruction in systems thinking, they located only 17 studies that used formal assessments. They characterized the assessments in terms of seven levels:

1. Recognizing interconnections: Listing system parts connections among parts, and emergent properties.
2. Identifying feedback: What parts are involve; negative or positive feedback.
3. Understanding dynamic behavior: Recognition and explanation of important behaviors.
4. Differentiating stocks, flows and other types.
5. Using conceptual models to explain the effect of parameter manipulation on behavior.
6. Constructing a model in a modeling language or editor.
7. Testing policies by using a model.

They concluded that that the Booth-Sweeney and Sterman assessment taps mostly the third and fourth levels, and thus are at best an incomplete measure of systems thinking.

Our model construction tasks involve levels 1, 2, 4 and 6. During the instruction presented here, students were never asked to draw or state the model's predictions, which is what tasks 3 and 5 require, because generating model predictions was the job of Dragoon. Thus, it seems likely that students who have aced our exam would do poorly on the Booth-Sweeney and Sterman assessment of systems thinking, but would do well on the Hopper and Stave assessment. In short, whether well-defined model construction is a part of systems thinking depends crucially on how systems thinking is assessed.

4.3 A Final Word

Because this discussion has focused mostly on speculations and future work, it is easy to lose track of the main result. In an average of about 5 hours of problem solving and 1 hour of lecture, six of the seven Tutor-High students were able to construct some fairly sophisticated models on the post-test. Moreover, this success can be attributed to the tutoring given by Dragoon, as the Editor-High students did much worse (effect size = 1.06) when using Dragoon as an editor with PowerPoint slides as a learning aid.

ACKNOWLEDGMENTS

This research was supported by the Office of Naval Research contract N00014-13-C-0029 and by US National Science Foundation (NSF) award 1123823.

REFERENCES

[1] Mathworks. (2015, Jan. 15) [Online]. Available: http://www.mathworks.com/products/matlab/

[2] ISEE. (2011). *Stella: Systems Thinking for Education and Research* [Online]. Available: http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx

[3] VentanaSystems. (2015, Jan. 15). [Online]. Available: http://vensim.com/

[4] PowerSim. (2015, Jan. 15). [Online]. Available: http://www.powersim.com/

[5] CCSSO, "The Common Core State Standards for Mathematics," ed: Downloaded from www.corestandards.org on Oct. 31, 2011, 2011.

[6] S. J. Stratford, "A review of computer-based model research in precollege science classroom," *J. Comput. Mathe. Sci. Teaching*, vol. 16, pp. 3–23, 1997.

[7] H. M. Doerr, "Stella ten-years later: A review of the literature," *Int. J. Comput. Mathe. Learn.*, vol. 1, pp. 201–224, 1996.

[8] K. VanLehn, "Model construction as a learning activity: A design space and review," *Interactive Learn. Environ.*, vol. 21, pp. 371–413, 2013.

[9] L. Bollen and W. R. Van Joolingen, "SimSketch: Multiagent simulations based on learner-created sketches for early science education," *IEEE Transactions on Learning Technologies*, vol. 6, no. 3, pp. 208–216, Jul. 2013.

[10] A. Repenning, A. Ioannidou, and J. Zola, "AgentSheets: End-user programmable simulations," *J. Artif. Societies Soc. Simul.*, vol. 3, 2000.

[11] S. Basu, J. S. Kinnebrew, A. Dickes, A. V. Farris, P. Sengupta, J. Winger, et al., "A science learning environment using a computational thinking approach," presented at the Proceedings of the 20th International Conference on Computers in Education, Singapore, 2012.

[12] R. Boohan, "Children and computer modelling: Making worlds with WorldMaker," in *Proc. 6th World Conf. Comput. Edu.*, 1995, pp. 975–985.

[13] E. K. Neumann, W. Feurzeig, and P. Garik, "An object-based modelling tool for science inquiry," in *Modelling and Simulation in Science and Mathematics Education*, W. Feurzeig and N. Roberts, Eds. New York, NY, USA: Springer, 1999, pp. 138–148.

[14] S. Fortmann-Roe and G. Bellinger. (2015). *Insight Maker* [Online]. Available: https://insightmaker.com/

[15] M. J. Jacobson and U. Wilensky, "Complex systems in education: Scientific and educational importance and implications for the learning sciences," *J. Learn. Sci.*, vol. 15, pp. 11–34, 2006.

[16] U. Wilensky and M. Resnick, "Thinking in levels: A dynamic systems approach to making sense of the world," *J. Sci. Edu. Technol.*, vol. 8, pp. 3–19, 1999.

[17] S. M. Alessi, "Building versus using simulations," in *Integrated and Holistic Perspectives on Learning, Instruction and Technology*, J. M. Spector and T. M. Anderson, Eds. Dordrecht, The Netherlands: Kluwer, 2000, pp. 175–196.

[18] S. M. Alessi, "The application of system dynamics modeling in elementary and secondary school curricula," in *–Proc. 5th Iberoamerican Conf. Informat. Edu.*, Viña del Mar, Chile, 2000.

[19] E. B. Mandinach and H. F. Cline, *Classroom Dynamics: Implementing a Technology-Based Learning Environment*. Mahwah, NJ, USA: Erlbaum, 1994.

[20] L. Zhang, K. Vanlehn, S. Girard, W. Burleson, M.-E. Chavez-Echeagaray, J. Gonzalez-Sanchez, et al., "Evaluation of a meta-tutor for constructing models of dynamic systems," *Comput. Edu.*, vol. 75, pp. 196–217, 2014.

[21] D. L. Schwartz, C. Chase, D. B. Chin, M. Oppezzo, H. Kwong, S. Y. Okita, et al., "Interactive metacognition: Monitoring and regulating a teachable agent," in *Handbook of Metacognition in Education*, D. J. Hacker, J. Dunlosky, and A. C. Graesser, Eds. New York, NY, USA: Taylor & Francis, 2009, pp. 340–358.

[22] D. L. Schwartz, K. P. Blair, G. Biswas, K. Leelawong, and J. Davis, "Animations of thought: Interactivity in the teachable agent paradigm," in *Learning with Animations: Research and Implications for Design*, R. Lowe and W. Schnotz, Eds. Cambridge, UK: Cambridge Univ. Press, 2008, pp. 114–141.

[23] G. Biswas, H. Jeong, J. S. Kinnebrew, B. Sulcer, and R. D. Roscoe, "Measuring self-regulated learning skills through social interactions in a teachable agent environment," *Res. Practice Technol. Enhanced Learn.*, vol. 5, pp. 123–152, 2010.

[24] K. Leelawong and G. Biswas, "Designing learning by teaching agents: The Betty's brain system," *Int. J. Artif. Intell. Edu.*, vol. 18, pp. 181–208, 2008.

[25] C. Bravo, W. R. van Joolingen, and T. de Jong, "Using co-lab to build system dynamics models: Students' actions and on-line tutorial advice," *Comput. Edu.*, vol. 53, pp. 243–251, 2009.

[26] W. R. van Joolingen, T. De Jong, A. Lazonder, E. R. Savelsbergh, and S. Manlove, "Co-Lab: Research and development of an online learning environment for collaborative scientific discovery learning," *Comput. Human Behavior*, vol. 21, pp. 671–688, 2005.

[27] S. Löhner, W. R. Van Joolingen, E. R. Savelsbergh, and B. Van Hout-Wolters, "Students' reasoning during modeling in an inquiry learning environment," *Comput. Human Behavior*, vol. 21, pp. 441–461, 2005.

[28] S. Löhner, W. R. Van Joolingen, and E. R. Savelsbergh, "The effect of external representation on constructing computer models of complex phenomena," *Instructional Sci.*, vol. 31, pp. 395–418, 2003.

[29] P. Sengupta, J. S. Kinnebrew, S. Basu, G. Biswas, and D. B. Clark, "Integrating computational thinking with K12 science education using agent-based computation: A theoretical framework," *Edu. Inf. Technol.*, vol. 18, pp. 351–380, 2013.

[30] S. J. Metcalf, J. Krajcik, and E. Soloway, "Model-It: A design retrospective," in *Innovations in Science and Mathematics Education: Advanced Designs for Technologies of Learning*, M. J. Jacobson and R. B. Kozma, Eds. Evanston, IL, USA: Routledge, 2000, pp. 77–115.

[31] B. A. Crawford and M. Cullin, "Supporting prospective teachers' conceptions of modelling in science," *Int. J. Sci. Edu.*, vol. 26, pp. 1370–1401, 2004.

[32] D. F. Treagust, G. Chittleborough, and T. Mamiala, "Students' understanding of the role of scientific models in learning science," *Int. J. Sci. Edu.*, vol. 24, pp. 357–368, 2002.

[33] S. P. van Borkulo, W. R. van Joolingen, E. R. Savelsbergh, and T. de Jong, "What can be learned from computer modeling? Comparing expository and modeling approaches to teaching dynamic systems behavior," *J. Sci. Edu. Technol.*, vol. 21, pp. 267–275, 2012.

[34] B. M. Richmond, "STELLA: Software for bringing system dynamics modeling to the other 98%," presented at the Proceedings of the 1985 International Conference of the System Dynamics Society: 1985 International System Dynamics Conference, 1985.

[35] R. Zaraza and D. Fisher, "Training system modelers: The NSF CC-STADUS and CC-SUSTAIN projects," in *Modeling and Simulation in Science and Mathematics Education*, vol. 1, W. Feurzeig and N. Roberts, Eds. New York, NY, USA: Springer, 1999, pp. 38–69.

[36] A. d. C. Kurtz dos Santos, M. R. Thielo, and A. A. Kleer, "Students modelling environmental issues," *J. Comput. Assisted Learn.*, vol. 13, pp. 35–47, 1997.

[37] W. Beek, B. Bredeweg, and S. Lautour, "Context-dependent help for the DynaLearn modelling and simulation workbench," in *Artificial Intelligence in Education*, G. Biswas, Ed. Berlin, Germany: Springer-Verlag, 2011, pp. 4200–422.

[38] B. Bredeweg, J. Liem, W. Beek, P. Salles, and F. Linnebank, "Learning spaces as representational scaffolds for learning conceptual knowledge of system behavior," in *Proc. 5th Eur. Conf. Technol. Enhanced Learn. Conf. Sustaining TEL: From Innovation Learn. Practice*, 2010, pp. 46–61.

[39] B. Bredeweg, F. Linnebank, A. Bouwer, and J. Liem, "Garp3 –Workbench for qualitative modelling and simulation," *Ecol. Informat.*, vol. 4, pp. 263–281, 2009.

[40] B. Bredeweg, A. Gomez-Perez, E. Andre, and P. Salles, "DynaLearn – Engaging and informed tools for learning conceptual system knowledge," presented at the AAAI, 2009.

[41] K. D. Forbus, K. Carney, B. L. Sherin, and L. C. Ureel II, "VModel: A visual qualitative modeling environment for middle-school students," *AI Mag.*, vol. 26, pp. 63–72, 2005.

[42] C. Rose, C. Torrey, V. Aleven, A. Robinson, C. Wu, and K. Forbus, "CycleTalk: Toward a dialogue agent that guides design with an articulate simulator," in *Proc. 7th Int. Conf. Intell. Tut. Syst.*, 2004, pp. 401–411.

[43] K. D. Forbus, L. C. Ureel Il, K. Carney, and B. L. Sherin, "Qualitative modeling for middle-school students," presented at the Qualitative Reasoning, 2004.

[44] B. Bredeweg and K. D. Forbus, "Qualitative modeling in education," *AI Mag.*, vol. 24, pp. 35–46, 2003.

[45] K. D. Forbus, J. O. Everett, L. Ureel, M. Brokowski, J. Baher, and S. E. Kuehne, "Distributed coaching for an intelligent learning enviornment," in *Proc. Qualititave Reseasoning*, Sea Crest Resort, Cape Cod, MA, 1998.

[46] K. D. Forbus and P. Walley, "Using qualitative physics to build articulate software for thermodynamics education," in *Proc. AAAI*, 1994, pp. 1175–1182.

[47] K. D. Forbus, "The role of qualitative dynamics in naive physics," in *Formal Theories of the Commonsense World*, J. R. Hobbes and R. C. Moore, Eds. Norwood, NJ, USA: Ablex, 1985, pp. 185–226.

[48] A. d. C. Kurtz dos Santos and J. Ogborn, "Sixth form students' ability to engage in computational modelling," *J. Comput. Assisted Learn.*, vol. 10, pp. 182–200, 1994.

[49] R. Miller, J. Ogborn, J. Briggs, D. Borough, J. Bliss, R. Boohan, et al., "Educational tools for computational modelling," *Comput. Edu.*, vol. 21, pp. 205–261, 1993.

[50] S. L. Jackson, S. J. Stratford, J. Krajcik, and E. Soloway, "A learning-centered tool for students building models," *Commun. ACM*, vol. 39, pp. 48–49, 1996.

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

[52] J. D. Sterman and L. Booth Sweeney, "Cloudy skies: Assessing public understanding of global warming," *Syst. Dyn. Rev.*, vol. 18, pp. 207–240, 2002.

[53] L. Booth Sweeney and J. D. Sterman, "Bathtub dynamics: Initial results of a systems thinking inventory," *Syst. Dyn. Rev.*, vol. 16, pp. 249–286, 2000.

[54] E. Moxnes, "Not only the tragedy of the commons: Misperceptions of feedback and policies for sustainable development," *Syst. Dyn. Rev.*, vol. 16, pp. 325–348, 2000.

[55] G. Ossimitz, "Stock-flow-thinking and reading stock-flow-related graphs: An empirical investigation in dynamic thinking abilities," presented at the International System Dynamics Conference, 2002.

[56] M. Cronin, C. Gonzalez, and J. D. Sterman, "Why don't well-educated adults understand accumulation? A challenge to researchers, educators and citizens," *Org. Behavior Human Decision Processes*, vol. 108, pp. 116–130, 2009.

[57] M. Cronin and C. Gonzalez, "Understanding the building blocks of dynamic systems," *Syst. Dyn. Rev.*, vol. 23, pp. 1–17, 2007.

[58] D. Kainz and G. Ossimitz, "Can students learn stock-flow-thinking? An empirical investigation," presented at the System Dynamics Conference, Palermo, Italy, 2002.

[59] P. W. B. Aikins, R. E. Wood, and P. J. Rutgers, "The effects of feedback format on dynamic decision making," *Org. Behavior Human Decision Processes*, vol. 88, pp. 587–604, 2002.

[60] S. Löhner, "Computer based modeling tasks: The role of external representation," Ph.D., Faculty of Social and Behavioural Sciences, Univ. of Amsterdam, Amsterdam, NL, 2005.

[61] S. J. Stratford, J. Krajcik, and E. Soloway, "Secondary students' dynamic modeling processes: Analyzing, reasoning about, synthesizing, and testing models of stream ecosystems," *J. Sci. Edu. Technol.*, vol. 7, pp. 215–234, 1998.

[62] K. Hogan and D. Thomas, "Cognitive comparisons of students' systems modeling in ecology," *J. Sci. Edu. Technol.*, vol. 10, pp. 319–345, 2001.

[63] P. H. M. Sins, E. R. Savelsbergh, and W. R. van Joolingen, "The difficult process of scientific modelling: An analysis of novices' reasoning during computer-based modelling," *Int. J. Sci. Edu.*, vol. 27, pp. 1695–1721, 2005.

[64] K. Thompson and P. Reimann, "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," *Comput. Edu.*, vol. 54, pp. 392–403, 2010.

[65] Y. G. Mulder, A. Lazonder, and T. de Jong, "Finding out how they find it out: An empirical analysis of inquiry learners' need for support," *Int. J. Sci. Learn.*, vol. 32, pp. 2033–2053, 2010.

[66] G. Biswas, K. Leelawong, D. L. Schwartz, and N. J. Vye, "Learning by teaching: A new agent paradigm for educational software," *Appl. Artif. Intell.*, vol. 19, pp. 263–392, 2005.

[67] M. Chi and K. VanLehn, "Eliminating the gap between the high and low students through meta-cognitive strategy instruction," in *Proc. 9th Int. Conf. Intell. Tut. Syst.*, 2008, pp. 603–613.

[68] M. Chi and K. VanLehn, "Meta-cognitive strategy instruction in intelligent tutoring systems: How, when and why," *J. Edu. Technol. Soc.*, vol. 13, pp. 25–39, 2010.

[69] N. T. Heffernan, K. R. Koedinger, and L. Razzaq, "Expanding the model-tracing architecture: A 3rd generation intelligent tutor for algebra symbolization," *Int. J. Artif. Intell. Edu.*, vol. 18, pp. 153–178, 2008.

[70] K. R. Koedinger and J. R. Anderson, "Illustrating principled design: The early evolution of a cognitive tutor for algebra symbolization," *Interactive Learn. Environ.*, vol. 5, pp. 161–180, 1998.

[71] D. McArthur, M. Lewis, T. Ormseth, A. Robyn, C. Stasz, and D. Voreck, *Algebraic Thinking Tools: Support for Modeling Situations and Solving Problems in Kids' World*. Santa Monica, CA, USA: *RAND Corporation*, 1989.

[72] S. Ramachandran and R. Stottler, "A meta-cognitive computer-based tutor for high-school algebra," in *Proc. World Conf. Edu. Multimedia, Hypermedia Telecommun.*, 2003, pp. 911–914.

[73] N. T. Heffernan, "Intelligent tutoring systems have forgotten the tutor: Adding a cognitive model of human tutors," Ph.D. dissertation, School of Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2001.

[74] S. Katz, D. Allbritton, and J. Connelly, "Going beyond the problem given: How human tutors use post-solution discussions to support transfer," *Int. J. Artif. Intell. Edu.*, vol. 13, pp. 79–116, 2003.

[75] S. Katz, J. Connelly, and C. Wilson, "Out of the lab and into the classroom: An evaluation of reflective dialogue in Andes," in *Proc. AI Edu.*, pp. 425–432.

[76] J. Connelly and S. Katz, "Toward more robust learning of physics via reflective dialogue extensions," presented at the Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications, 2009, pp. 1946–1951.

[77] L. Anthony, A. T. Corbett, A. Z. Wagner, S. M. Stevens, and K. R. Koedinger, "Student question-asking patterns in an intelligent algebra tutor," in *Proc. 7th Int. Conf. Intell. Tut. Syst.*, 2004, pp. 455–467.

[78] A. Corbett, A. Z. Wagner, C.-Y. Chao, S. Lesgold, S. M. Stevens, and H. Ulrich, "Student questions in a classroom evaluation of the ALPS learning environment," in *Artificial Intelligence in Education*, C.-K. Looi and G. McCalla, Eds. Amsterdam, The Netherlands: IOS Press, 2005, pp. 780–782.

[79] J. R. Segedy, J. S. Kinnebrew, and G. Biswas, "Supporting student learning using conversational agents in a teachable agent environment," presented at the Proceedings of the 10th International Conference of the Learning Sciences, Sydney, Australia, 2012.

[80] W. Bridewell, J. N. Sanchez, P. Langley, and D. Billman, "An interactive environment for the modeling and discovery of scientific knowledge," *Int. J. Human-Comput. Studies*, vol. 64, pp. 1099–1114, 2006.

[81] M. J. Nathan, W. Kintsch, and E. Young, "A theory of algebra-word-problem comprehension and its implications for the design of learning enviroments," *Cognition Instruction*, vol. 9, pp. 329–389, 1992.

[82] M. J. Nathan, "Knowledge and situational feedback in a learning environment for algebra story problem solving," *Interactive Learn. Environ.*, vol. 5, pp. 135–159, 1998.

[83] S. J. Metcalf, "The design of guided learning-adaptable scaffolding in interactive learning environments," Ph.D. dissertation, Comput. Science and Eng. Dept., Univ. of Michigan, Ann Arbor, MI, USA, 1999.

[84] Y. G. Mulder, A. W. Lazonder, T. de Jong, A. Anjewierden, and L. Bollen, "Validating and optimizing the effects of model progression in simulation-based inquiry learning," *J. Sci. Edu. Technol.*, vol. 21, pp. 722–729, 2011.

[85] B. Y. White, "ThinkerTools: Causal models, conceptual change and science education," *Cognition Instruction*, vol. 10, pp. 1–100, 1993.

[86] B. Y. White and J. R. Frederiksen, "Causal model progressions as a foundation for intelligent learning environments," *Artif. Intell.*, vol. 42, pp. 99–157, 1990.
[87] B. Y. White, "Designing computer games to help physics students understand Newton's Laws of Motion," *Cognition Instruction*, vol. 1, pp. 69–108, 1984.
[88] J. Swaak, W. R. van Joolingen, and T. de Jong, "Supporting simulation-based learning; The effects of model progression and assignments on definition and intuitive knowledge," *Learn. Instruction*, vol. 8, pp. 235–252, 1998.
[89] T. de Jong, E. Martin, J.-M. Zamarro, F. Esquembre, J. Swaak, and W. R. van Joolingen, "The integration of computer simulation and learning support: An example from the physics domain of collisions," *J. Res. Sci. Teaching*, vol. 36, pp. 597–615, 1999.
[90] J. Quinn and S. M. Alessi, "The effects of simulation complexity and hypothesis-generation strategy on learning," *J. Res. Comput. Edu.*, vol. 27, pp. 75–92, 1994.
[91] F. Reif and L. A. Scott, "Teaching scientific thinking skills: Students and computers coaching each other," *Am. J. Phys.*, vol. 67, pp. 819–831, 1999.
[92] T.-W. Chan and C.-Y. Chou, "Exploring the design of computer supports for reciprocal tutoring," *Int. J. Artif. Intell. Edu.*, vol. 8, pp. 1–29, 1997.
[93] C. C. Chase, D. B. Chin, M. Oppenzzo, and D. L. Schwartz, "Teachable agents and the Protégé effect: Increasing the effort towards learning," *J. Sci. Edu. Technol.*, vol. 18, pp. 334–352, 2009.
[94] L. Pareto, T. Arvemo, Y. Dahl, M. Haake, and A. Gulz, "A teachable-agent arithmetic game's effects on mathematics understanding, attitude and self-efficacy," in *Proc. Artif. Intell. Edu.*, 2011, pp. 247–255.
[95] S. P. Marshall, K. E. Barthuli, M. A. Brewer, and F. E. Rose, "Story Problem Solver: A schema-based system of instruction," *Center for Research in Mathematics and Science Education*. San Diego, CA, USA: San Diego State Univ., 1989.
[96] J. Gonzalez-Sanchez, M.-E. Chavez-Echeagaray, K. VanLehn, and W. Burleson, "From behavioral description to a pattern-based model for intelligent tutoring systems," presented at the SPLASH: Software, Programming, Languages and Applications: Software for Humanity, Portland, OR, USA, 2011.
[97] L. Zhang, W. Burleson, M.-E. Chavez-Echeagaray, S. Girard, J. Gonzalez-Sanchez, Y. Hidalgo Pontet, et al., "Evaluation of a meta-tutor for constructing models of dynamic systems," in *Proc. 16th Int. Conf. Artif. Intell. Edu.*, 2013, pp. 666–669.
[98] S. Girard, L. Zhang, Y. Hidalgo Pontet, K. VanLehn, W. Burleson, M. E. Chavez Echeagary, and J. Gonzalez-Sanchez, "Using HCI task modeling techniques to define how deeply students model," in *Proc. 16th Int. Conf. Artif. Intell. Edu.*, 2013, pp. 766–769.
[99] S. Girard, M.-E. Chavez-Echeagaray, J. Gonzalez-Sanchez, Y. Hidalgo Pontet, L. Zhang, W. Burleson, et al., "Defining the behavior of an affective learning companion in the affective meta-tutor project," in *Proc. 16th Int. Conf. Artif. Intell. Edu.*, 2013, pp. 21–30.
[100] K. VanLehn, W. Burleson, M.-E. Chavez-Echeagaray, R. Christopherson, J. Gonzalez-Sanchez, J. Hastings, et al., "The affective meta-tutoring project: How to motivate students to use effective meta-cognitive strategies," presented at the 19th International Conference on Computers in Education, Chiang Mai, Thailand, 2011.
[101] K. VanLehn, W. Burleson, M.-E. Chavez-Echeagaray, R. Christopherson, J. Gonzalez-Sanchez, J. Hastings, et al., "The level up procedure: How to measure learning gains without pre- and post-testing," in *Proc. 19th Int. Conf. Comput. Edu.*, 2011, pp. 96–100.
[102] S. L. Jackson, S. J. Stratford, J. Krajcik, and E. Soloway, "Making dynamic modeling accessible to precollege science students," *Interactive Learn. Environ.*, vol. 4, pp. 233–257, 1994.
[103] H. H. Clark and S. E. Brennan, "Grounding in communication," in *Perspectives on Socially Shared Cognition*, L. B. Resnick, J. M. Levine, and S. D. Teasley, Eds. Washington, DC, USA: American Psychological Assoc., 1991, pp. 127–149.
[104] Y. G. Mulder, L. Bollen, and T. De Jong, "Learning from erroneous models using SCYDynamics," presented at the Proceedings of the 32nd International Conference of the System Dynamics Society, Delft, Netherlands, 2014.
[105] K. VanLehn, "The behavior of tutoring systems," *Int. J. Artif. Intell. Edu.*, vol. 16, pp. 227–265., 2006.
[106] K. Vanlehn and M. Chi, "Adaptive expertise as acceleration of future learning: A case study," in *Adaptive Technologies for Training and Education*, P. J. Durlach and A. Lesgold, Eds. Cambridge, U.K.: Cambridge Univ. Press, 2012.
[107] I. Roll, V. Aleven, B. McLaren, and K. R. Koedinger, "Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system," *Learn. Instruction*, vol. 21, pp. 267–280, 2011.
[108] V. Aleven, E. Stahl, S. Schworm, F. Fischer, and R. M. Wallace, "Help seeking and help design in interactive learning environments," *Rev. Edu. Res.*, vol. 73, pp. 277–320, 2003.
[109] R. S. Baker, A. Corbett, K. R. Koedinger, S. Evenson, I. Roll, A. Z. Wagner, et al., "Adapting to when students game an intelligent tutoring system," in *Intelligent Tutoring Systems*. Berlin, Germany: Springer, 2006, pp. 392–401.
[110] R. C. Murray and K. VanLehn, "Effects of dissuading unnecessary help requests while providing proactive help," in *Proc. Artif. Intell. Edu.*, 2005, pp. 887–889.
[111] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2009.
[112] K. VanLehn, "Cognitive skill acquisition," in *Annual Review of Psychology*, vol. 47, J. Spence, J. Darly, and D. J. Foss, Eds. Palo Alto, CA, USA: Annual Reviews, 1996, pp. 513–539.
[113] R. E. Mayer, "Frequency norms and structural analysis of algebra story problems into families, categories and templates," *Instructional Sci.*, vol. 10, pp. 135–175, 1981.
[114] B. C. Buckley, J. D. Gobert, A. C. H. Kindfield, P. Horwitz, R. F. Tinker, B. Gerlits, et al., "Model-based teaching and learning with BioLogica: What do they learn? How do they learn? How do we know?," *J. Sci. Edu. Technol.*, vol. 13, pp. 23–41, 2004.
[115] T. de Jong and W. R. van Joolingen, "Scientific discovery learning with computer simulations of conceptual domains," *Rev. Edu. Res.*, vol. 68, pp. 179–201, 1998.
[116] B. K. A. Weusijana, C. Riesbeck, and J. T. Walsh, "Fostering reflection with Socratic tutoring software: Results of using inquiry teaching strategies with web-based HCI techniques.," in *Proc. 6th Int. Conf. Learn. Sci.*, 2004, pp. 535–541.
[117] K. VanLehn, G. Chung, S. Grover, A. Madni, and J. Wetzel, "Learning about dynamic systems and domain principles: The effectiveness of the Dragoon intelligent tutoring system," *Int. J. Artif. Intell. Edu.*, in press.
[118] D. M. Iwaniec, D. L. Childers, K. VanLehn, and A. Wiek, "Studying, teaching and applying sustainabilty visions using systems modeling," *Sustainability*, vol. 6, pp. 4452–4469, 2014.
[119] C. B. Lee, D. Jonassen, and T. Teo, "The role of model building in problem solving and conceptual change," *Interactive Learn. Environ.*, vol. 19, pp. 247–265, 2011.
[120] E. B. Mandinach and H. F. Cline, "Modeling and simulation in the secondary school curriculum: The impact on teachers," *Interactive Learn. Environ.*, vol. 4, pp. 271–289, 1994.
[121] O. Pala and J. A. M. Vennix, "Effect of system dynamics education on systems thinking inventory task performance," *Syst. Dyn. Rev.*, vol. 21, pp. 147–172, 2005.
[122] M. Hopper and K. Stave, "Assessing the effectiveness of systems thinking interventions in the classroom," presented at the International Conference of the System Dynamics Society, Athens, Greece, 2008.

Kurt VanLehn received the BS degree in mathematics from Stanford in 1974, and the MS and PhD degrees in computer science from M.I.T. in 1983 and 1978, respectively. He has been a professor at C.M.U, the University of Pittsburgh, and now at Arizona State University, where he is the the Diane and Gary Tooker Chair for effective education in science, technology, engineering and math. He is on the editorial boards of the *International Journal of A.I. in Education* and *Cognition and Instruction*. He has published more than 165 refereed journal articles and conference papers, including 11 that have received best paper awards. His main research area is intelligent interactive instructional systems. He is a fellow in the Cognitive Science Society.

Jon Wetzel received the BS and MEng degrees in computer science & engineering from M.I.T. in 2006 and 2007, respectively, and the PhD degree in computer science from Northwestern University in 2014. He now has a postdoctoral scholar position at Arizona State University, where he leads the Dragoon project. He has received a US National Science Foundation Graduate Resarch Fellowship, and has published more than 15 refereed journal, conference, and workshop publications. He also has one patent. His main research area is artificial intelligence for intelligent interactive tutoring systems.

Sachin Grover received the BTech degree from the National Institute of Technology, Rourkela, India in 2010 and is currently working toward the MS degree in computer science from Arizona State University. He has worked as an analyst for web technologies at Sapient Inc and now works as a graduate research assistant advised by Prof. Kurt VanLehn. He has published one IEEE conference paper in the area of image processing. His main areas of interest are planning in autonomous agents and intelligent interactive tutoring systems.

Brett van de Sande received the BS degree in physics from Caltech in 1988 and the PhD degree in physics from Ohio State in 1995. He was a professor at Geneva College, a research professional at the University of Pittsburgh and Arizona State University, and is currently a research/data scientist at Pearson Education. He received a Von Humboldt Fellowship and has a patent. He has published 27 refereed journal articles and conference papers, including one that has received a best paper award.