Computers & Education 75 (2014) 196–217

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Evaluation of a meta-tutor for constructing models of dynamic systems

Lishan Zhang*, Kurt VanLehn, Sylvie Girard, Winslow Burleson, Maria Elena Chavez-Echeagaray, Javier Gonzalez-Sanchez, Yoalli Hidalgo-Pontet

Arizona State University, Computing, Informatics, and Decision Systems Engineering, Tempe, AZ 85281, USA

ARTICLE INFO

Article history:
Received 5 September 2013
Received in revised form
17 February 2014
Accepted 25 February 2014
Available online 13 March 2014

Keywords:
Meta-tutor
Gaming the system
Intelligent tutoring systems
Modelling
Learning strategies

ABSTRACT

Modelling is an important skill to acquire, but it is not an easy one for students to learn. Existing instructional technology has had limited success in teaching modelling. We have applied a recently developed technology, meta-tutoring, to address the important problem of teaching model construction. More specifically, we have developed and evaluated a system that has two parts, a tutor and a meta-tutor. The tutor is a simple step-based tutoring system that can give correct/incorrect feedback on student's steps and can demonstrate steps for students when asked. Because deep modelling requires difficult analyses of the quantitative relationships in a given system, we expected, and found, that students tended to avoid deep modelling by abusing the tutor's help. In order to increase the frequency of deep modelling, we added a meta-tutor that coached students to follow a learning strategy that decomposed the overall modelling problem into a series of "atomic" modelling problems. We conducted three experiments to test the effectiveness of the meta-tutor. The results indicate that students who studied with meta-tutor did indeed engage in more deep modelling practices. However, when the meta-tutor and tutor were turned off, students tended to revert to shallow modelling. Thus, the next stage of the research is to add an affective agent that will try to persuade students to persist in using the taught strategies even when the meta-tutoring and tutoring have ceased.

1. Introduction

This paper reports progress on two research problems: (1) teaching students how to construct mathematical models of dynamic systems, and (2) teaching students to use effective learning strategies. Both research problems have long histories, which are covered in the next few sections.

1.1. A brief history of educational uses of system dynamics modelling

There are two distinct reasons why students should learn model construction. First, modelling is an important cognitive skill in itself. The Common Core State Standards for Mathematics (CCSSO, 2011) considers modelling to be one of 7 essential mathematical practices that should be taught at all grade levels. The Next Gen standards for science instruction (National, 2012) also have 7 strands that are threaded throughout the standards, and modelling is one of them.

Second, modelling is widely believed to be an important method for learning domain knowledge. For instance, modelling has been claimed to help in achieving a deep understanding of scientific systems, economic systems and other systems (Chin et al., 2010; Metcalf, Krajcik, & Soloway, 2000; Stratford, 1997), removing misconceptions and making of conceptual changes (Booth Sweeney & Sterman, 2000; Bredeweg & Forbus, 2003; Hestenes, 2007; Lee, Jonassen, & Teo, 2011; Mandinach & Cline, 1994b; Wilensky, 2003; Wilensky &

* Corresponding author.
E-mail addresses: lishan.zhang@asu.edu (L. Zhang), kurt.vanlehn@asu.edu (K. VanLehn), sylvie.girard@asu.edu (S. Girard), winslow.burleson@asu.edu (W. Burleson), mchaveze@asu.edu (M.E. Chavez-Echeagaray), javiergs@asu.edu (J. Gonzalez-Sanchez), yhidalgo@asu.edu (Y. Hidalgo-Pontet).

http://dx.doi.org/10.1016/j.compedu.2014.02.015
0360-1315/© 2014 Elsevier Ltd. All rights reserved.

Reisman, 2006), understanding the epistemology of models in science (Treagust, Chittleborough, & Mamiala, 2002), and developing intuitions, predilections and skills at understanding complex phenomena in general (Hogan & Thomas, 2001; Mandinach & Cline, 1994a; Schecker, 1993; Steed, 1992).

In short, modelling is both an important cognitive skill and a potentially powerful means of learning many topics. That is, it is both an end goal and a means to other end goals.

The modelling activity addressed here is traditionally called *system dynamics* modelling (see Collins and Ferguson (1993) for a particularly comprehensive taxonomy of modelling). The model is comprised of real-valued variables that are constrained by temporal differential equations. The variables denote quantities in the system whose values change over time. The job of the model is to predict those changing values as accurately as parsimony allows.

There is a long history of using system dynamics model construction as in instructional activity. According to the oral history of the System Dynamics Society (http://www.systemdynamics.org/oral-history/), system dynamics began to be used for university instruction around 1957 with Jay Forrester's formulation of system dynamics for teaching management. When a graphical language, Stella, became available (Richmond, 1985), instructional usage dramatically increased and extended to high school. Many early Stella projects trained teachers in modelling and let them invent activities (Mandinach & Cline, 1994a; Zaraza & Fisher, 1997).

After years of experience by hundreds of teachers, observers began to report that getting students to actually construct models took so much class time that most teachers used Stella only for model exploration activities, wherein students were given a model and were asked to observe how graphs of the variables values changed as the students manipulated parameters of the model (Alessi, 2000; Doerr, 1996; Mandinach & Cline, 1994b; Stratford, 1997).

Laboratory studies confirmed the observers' reports about both the length of time required for model construction and the importance of model construction. For example, Hashem and Mioduser (2011) found that students who *constructed* NetLogo models learned more about emergence, self-organization and other complex system concepts than students who *explored* NetLogo models that were given to them. The comparison took place during two 90-min lessons on complex systems, which were flanked by a pre-test and a post-test. However, prior to the pre-test, it took only 2 h to train the model exploration group whereas it took 48 h to train the model construction group. In a review of the modelling literature, VanLehn (2013) found that the only experiments that produced reliable positive results for model construction also devoted at least 5 h to training the students before the main lessons.

This history motivates the specific research problem addressed here: How can we speed up students' acquisition of skill in constructing system dynamics models?

Many methods for accelerating the acquisition of skill in model construction have been implemented, but only a few have been compared to baseline versions of the model construction activity in order to test their effectiveness (see VanLehn, 2013, for a review). Of those that have been evaluated, one form of scaffolding has shown considerable promise: The use of feedback and hints on student's steps in constructing the model. A whole model is usually composed of many parts (e.g., nodes, links, equations, labels, icons, numbers, etc.) which the student enters one at a time. Entering such a part is called a "step". Systems that give feedback and hints on steps are called *step-based tutoring systems* (VanLehn, 2006). Step-based tutoring systems have been used for a wide variety of tasks besides model construction, and appear to be almost as effective as human tutors (VanLehn et al., 2011). Thus, this project decided early on to build a step-based tutoring system for system dynamics modelling in the hope that it would accelerate students' acquisition of modelling skill.

1.2. *Learning strategies research*

A learning strategy is a process, procedure or method that meets two criteria (Donker, de Boer, Kostons, Dignath van Ewijk, & van der Werf, 2014): (1) Students can use the strategy when studying, but it is not required by the material that they are studying. (2) Using the learning strategy is believed to affect the student's learning. A good learning strategy is thought to improve students' learning, while a poor learning strategy is thought to harm the students' learning. When used without modification, "learning strategy" generally means a good learning strategy. Some examples are:

- When memorizing facts, a good learning strategy is to construct a mental image and associate each fact with a part of the image.
- When studying an example, a good learning strategy (called self-explanation) is to explain each step in the example to yourself, asking "Why is this true? Why did the author include this step?"
- When reading a text, a good learning strategy is to reflect afterwards on what you have learned.
- When reading a text, a poor learning strategy is to ignore words or passages that you don't understand.

Learning strategies have been studied for decades, and comprehensive meta-analytic reviews exist (Donker et al., 2014; Hattie, Biggs, & Purdie, 1996). Some of the main findings are:

A. Students often exhibit poor learning strategies.
B. Good learning strategies can be taught, often with little difficulty.
C. When students use the taught learning strategies, their domain learning often increases compared to students who are not taught to use the learning strategies.
D. When instruction in the learning strategy includes meta-cognitive and motivational components, students can often be prevented from reverting to poor learning strategies when the instruction ceases.
E. Specific learning strategies often have larger effect sizes than general purpose ones.

These findings mean that research on learning strategies is intimately linked to advances in instruction. Whenever a new instructional method or subject matter is developed, there are likely to be poor learning strategies that are specific to it (finding A) as well as specific good learning strategies that are likely to be effective (finding E) and easily taught (B). The key question is whether they are effective for domain learning (C) and students can be persuaded to continue using them after being taught (D). We have developed a step-based tutoring system for teaching students how to construct models, so it is likely that for this new form of instruction, students tend to exhibit poor learning strategies, but that good strategies can be easily taught; the main questions are whether the good strategies really are effective at increasing domain learning and whether students can be taught to persist in using them. These are the main research questions addressed here.

The next introductory section focuses on reviewing projects that are similar to the one described here. However, their relationship to learning strategies can be a bit hard to see, so a few prefatory remarks may be helpful.

Because this paper is concerned with teaching students how to construct models, and constructing a model is a kind of problem solving, it is worth clarifying how learning strategies differ from problem solving strategies. When students are taught an effective strategy for solving problems, the strategy qualifies as either a learning strategy, a problem solving strategy or both depending on the instructional objectives.

- If the objective is for students to solve problems well, then a strategy for solving problems counts as domain knowledge and is called a problem solving strategy. In the jargon of students, it's on the test.
- On the other hand, when problem solving is done only to give student practice in applying certain concepts and principles and the problem solving itself is not an instructional objective, then the strategy counts as a learning strategy because it is optional and yet it probably impacts the students' learning of the concepts and principles. In the jargon of students, it is not on the test.
- As pointed out earlier, the process of constructing models is both an instructional objective (especially in math classes) and a means for acquiring domain knowledge (especially in science classes). Thus, an effective strategy for constructing models counts as both a problem solving strategy in some cases (it's on the test) and a learning strategy in other cases (it's not on the test).

For example, suppose the instructional objective is to learn which botanical features go with which plants, and the instruction involves arranging cards labelled with plants and features. If this matching activity does not appear on the exams, then teaching students a methodical method for arranging the cards counts as a learning strategy even though it is also a problem solving strategy. From the students' point of view, what matters is whether the strategy is "on the test." That is, if a problem solving strategy is an instructional objective and is part of a test or other assessment, then students have a different attitude towards it than a strategy that is merely helpful for learning the material that will be on the test. As we discuss various methods for teaching model construction, it is important to note whether students believe a taught strategy is required or merely helpful.

Interactive instructional systems, such as the tutoring system described herein, have their own unique learning strategies. Here are some examples:

1. Teachers often complain that their students would rather click than think. Actuating buttons, menus, etc. without thinking or even reading the rest of the screen is a poor learning strategy.
2. When a tutoring system gives immediate feedback on the correctness of a student's entry, then students often guess instead of think. That is, they rapidly revise and resubmit an incorrect entry until the tutoring system says that it is correct.
3. If the tutoring system gives a sequence of hints that get gradually mores specific until they tell the student exactly what to do, then students abuse the hint sequences by clicking rapidly on the Hint button until they get the final hint that tells them what to do.
4. A good learning strategy for tutoring systems is to ask for a hint only when you need one (Aleven, McLaren, Roll, & Koedinger, 2004).
5. After asking for hints from a tutoring system and finally making a correct entry, a good learning strategy is to reflect on why it is correct (self-explanation) and whether the hint makes sense (Shih, Koedinger, & Scheines, 2008).
6. Examples 2, 3 and 4 are help-seeking strategies (Aleven, Stahl, Schworm, Fischer, & Wallace, 2003). Examples 2 and 3 are often referred to as "gaming the system" (Baker, Corbett, Koedinger, & Wagner, 2004).

Feedback and hints are the signature methods of tutoring, but several projects have used feedback and hints for two distinct purposes, so let us distinguish them as follows:

- Let *domain knowledge* refer to what students are supposed to learn. It is typically measured with a post-test and sometimes a pre-test.
- When the hints address the domain and the feedback indicates whether the domain knowledge has been correctly applied, the system is said to be *tutoring* and the module responsible for it is called the *tutor*.
- A *learning strategy* is an optional method for using the system (i.e., it is not domain knowledge) that is believed to increase student's learning of domain knowledge.
- When the feedback and hints refer only to the learning strategy and whether it is being applied correctly, the system is said to be *meta-tutoring* and the module responsible for it is called the *meta-tutor*.

In this paper, meta-tutoring will only means a method for teaching students a learning strategy. However, in the tutoring literature, "meta-tutoring" is used more broadly to mean using feedback and hints to teach anything other than domain knowledge. For instance, meta-tutoring is sometimes used to increase motivation or change beliefs about self-efficacy. Du Boulay, Avramides, Luckin, Martinez-Miron, and Rebolledo-Mendez (2010) propose a framework that includes many types of meta-tutoring.

Now we can turn to reviewing prior work on using learning strategies to help students learn how to construct models.

1.3. Prior work on learning strategies for model construction

Betty's Brain (Leelawong & Biswas, 2008) was a step-based tutoring system for constructing models that also taught a learning strategy. It could give feedback and hints on the student's model (tutoring) or it could give feedback and hints on the way that the student was using the system to create the model (meta-tutoring).[1] For instance, sometimes it would not permit the student to evaluate the model with an instructor-provide test suite (called a "quiz") until the student had first examined specific predictions of their model (e.g., if air temperature goes down, what does body temperature do?). The system included some multimedia resources on the task domain, so another part of the learning strategy was encouraging students to read them. As these examples indicate, the learning strategy was specific to the particular instructional features provided by system. This is consistent with the findings in the learning strategies literature, which suggest that such specificity provides better results than general learning strategies.

Three evaluations of the effectiveness of Betty's meta-tutor were conducted (Biswas, Leelawong, Schwartz, & Vye, 2005, study 2; Leelawong & Biswas, 2008; Tan, Biswas, & Schwartz, 2006). Their methods will be described fully here, as the experiments to be reported later used similar methods. The Betty's Brain experiments all had two phases, called the *training* phase and the *transfer* phase here. During the training phase, fifth-grade students worked with Betty for approximately seven 45-min sessions on constructing a model of a river system. One group of students used Betty's Brain with the meta-tutor turned on, and another group used the same system with the meta-tutor turned off. Two months later the transfer phase occurred, where all the students used Betty's Brain with the meta-tutor turned off to create models for the nitrogen cycle. This transfer phase was used to assess their modelling skill.

The results were roughly the same in all three studies. Using the general results on learning strategy mentioned above as a framework, the results from the Betty's Brain studies were:

A. *Students often exhibit poor learning strategies*. True of the control conditions in all three studies.
B. *Good learning strategies can be taught, often with little difficulty*. In the training phase of all three studies, meta-tutored students' behaviour was consistent with the learning strategies.
C. *When students use the taught learning strategies, their domain learning increases compared to students who are not taught to use the learning strategies*. Unfortunately, using conventional science tests of ecology concepts, there were few significant differences between the meta-tutored students and the students who used Betty's Brain without meta-tutoring.[2] Using four measures of model quality, meta-tutored students' models were better than control students' models on only one measure.
D. *When instruction in the learning strategy includes meta-cognitive and motivational components, students can often be prevented from reverting to poor learning strategies when the instruction ceases*. Students who received meta-tutoring during the training phase tended to continue using the learning strategy in the transfer phase when the meta-tutoring was turned off.
E. *Specific learning strategies often have larger effect sizes than general purpose ones*. Not tested.

In short, although the meta-tutor in Betty's Brain was successful at teaching the learning strategy and persuading students to continue using it, the learning strategy had little impact on domain learning. More recent work has focused on finding out what behaviours distinguish good learners from poor learners (Segedy, Kinnebrew, & Biswas, 2012a, 2012b).

The Help Tutor (Aleven et al., 2004; Roll, Aleven, McLaren, & Koedinger, 2007a, 2007b; Roll, Aleven, McLaren, & Koedinger, 2011; Roll et al., 2006) was a meta-tutor that augmented an existing step-based tutoring system, the Cognitive Geometry Tutor (www.carnegielearning.com). Although the tutoring system did not teach students to construct models, it is included in this review of past work because it is an often-cited representative of using meta-tutoring to improve students' learning from step-based tutoring.

The Geometry Cognitive Tutor allowed students to ask for a hint. The first hint was rather general, but if the student kept asking for hints, then the last hint told the student exactly what to do. This was called the "bottom-out" hint. Students sometimes clicked rapidly on the Help button so that they could get to the bottom-out hint. As mentioned earlier, this abuse of the help system is a kind of "gaming the system" (Baker, Corbett, Koedinger, et al., 2004). In order to reduce the frequency of gaming the system, the Help Tutor gave students feedback and hints on their use of the help system.

Two studies evaluated the Help Tutor. As in the Betty's Brain studies, the Help Tutor studies had both a training phase where the meta-tutor was used by half the students and a transfer phase where none of the students used the meta-tutor. Using the general findings mentioned above as a framework, the results were:

A. *Students often exhibit poor learning strategies*. True of the control conditions in all both studies.
B. *Good learning strategies can be taught, often with little difficulty*. In the training phase of both studies, meta-tutored students' behaviour was consistent with the taught strategies.
C. *When students use the taught learning strategies, their domain learning increases compared to students who are not taught to use the learning strategies*. Unfortunately, students taught the learning strategy did not differ from the control group in their acquisition of geometry knowledge.
D. *When instruction in the learning strategy includes meta-cognitive and motivational components, students can often be prevented from reverting to poor learning strategies when the instruction ceases*. Students who received meta-tutoring during the training phase tended to continue using the learning strategy in the transfer phase when the meta-tutoring was turned off, albeit with less frequency.
E. *Specific learning strategies often have larger effect sizes than general purpose ones*. Not tested.

Betty's Brain and the Help Tutor provided most of their instruction via feedback and hints. A somewhat more didactic approach is to provide forms and phases that constrain students' modelling behaviour. For instance, the final version of Model-It (Metcalf et al., 2000) had

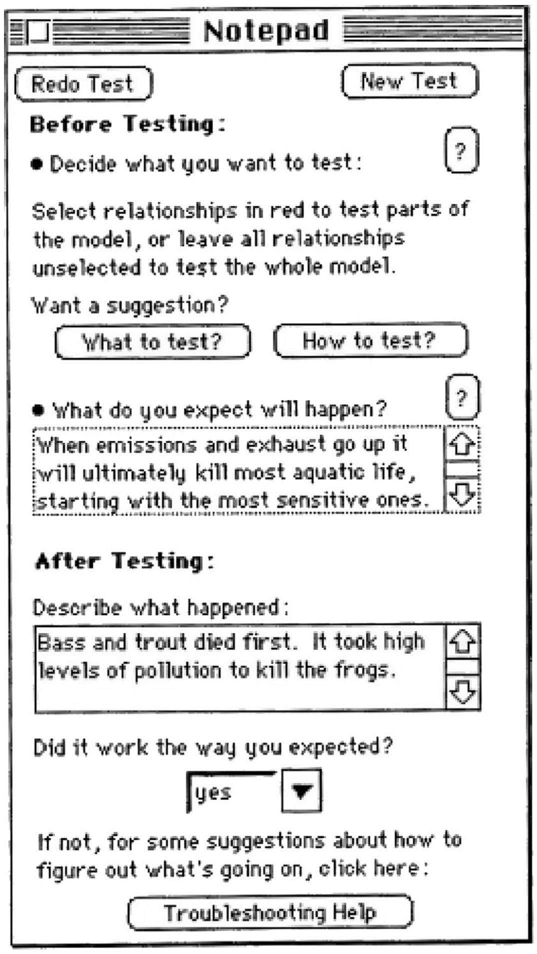
[1] Those familiar with Betty's Brain might be surprised to see it described as a tutoring system because the students see the system as two agents: Betty (a teachable agent) and Mr. Davis (a mentor). Students were told that the model they were constructing comprised the knowledge of Betty, so editing the model comprised "teaching Betty." When they ask Betty take a quiz, Mr. Davis gives the student feedback on the correctness of the model's predictions which would sometimes include unsolicited hints about fish, algae, carbon dioxide and other domain entities. Mr. Davis personifies the system's tutoring. On the other hand, meta-tutoring was done by both agents. For instance, Betty would sometimes refuse to take a quiz, and Mr. Davis would discourage students from using trial-and-error methods.

[2] In a fourth study, the meta-tutored students produced slightly better river system models than the control students (Tan, Wagster, Wu, & Biswas, 2007; Wagster, Tan, Biswas, & Schwartz, 2007; Wagster, Tan, Wu, Biswas, & Schwartz, 2007). However, results on the other measures were not reported.

4 phases, which were selected by clicking on one of four buttons labelled Plan, Build, Test and Evaluate. The Build mode was the actual model editor. The other phases presented forms to be filled in by the student. Fig. 1 shows the form for the Test phase. Students were given feedback and hints, which they could suppress if they desired, when they attempted to 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 phases and forms. None of these meta-tutors were evaluated separately from the rest of the system.

On the other hand, Mulder, Lazonder, de Jong, Anjewierden, and Bollen (2011) did assess the effects of a phase-based learning strategy. The experiments used Co-Lab, a system that taught system dynamics modelling. Co-Lab was not a step-based tutoring. Instead, students received feedback only on the accuracy of the model's predictions. The learning strategy was to encourage students to do their model construction in three stages. In Stage 1, they defined variables and drew undirected links between variables that directly affected each other somehow. In Stage 2, they added qualitative labels to the links indicating whether an increase in one variable caused an increase or decrease in the other variable. In Stage 3, they added equations to the model that further specified the relationships between variables. Three versions of the learning strategy were compared to a control version of Co-Lab that lacked phases. The three versions differed in how strictly they enforced the learning strategy. The restricted strategy required students to achieve a certain level of success in one stage before moving to the next. The semi-restricted version of the strategy allowed students to move from one stage to the next at will, but they were not allowed to move backwards. The unrestricted version allowed students to move at will between stages. Compared to the control version, all three versions of the learning strategy increased the number of correct model elements generated by students as they used the system. The three versions were not significantly different in productivity from each other, but there was a trend for the semi-restricted version to be better than the other two. However, this experiment included only a training phase and not a transfer phase, and it did not assess students' domain knowledge with pre- and post-testing. Thus, its consistency with the 5 general findings mentioned above (A through F) cannot be determined. Nonetheless, this work is interesting because the learning strategy was taught without using a meta-tutor, and the system was not a step-based tutoring system.

The meta-tutoring of Betty's Brain and the Help Tutor placed only weak constraints on students' behaviour. The phases and forms of Co-Lab, Model-It, the Cognitive Tutors and the Algebra Word Problem tutor placed somewhat stronger constraints on students' behaviour. At the far end of this progression is *procedural scaffolding*, which places very strong constraints on student behaviour. The basic idea of procedural scaffolding is to require 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 construct models, the procedure is used as a temporary scaffolding to guide students who might otherwise be quite lost.



Before Testing:

● Decide what you want to test:

Select relationships in red to test parts of the model, or leave all relationships unselected to test the whole model.

Want a suggestion?

● What do you expect will happen?

When emissions and exhaust go up it will ultimately kill most aquatic life, starting with the most sensitive ones.

After Testing:

Describe what happened:

Bass and trout died first. It took high levels of pollution to kill the frogs.

Did it work the way you expected?

yes

If not, for some suggestions about how to figure out what's going on, click here:

Fig. 1. Scaffolding for the Test mode of Model-It (Metcalf, 1999).

Although procedural scaffolding was first used by (Marshall, Barthuli, Brewer, & Rose, 1989) to scaffold arithmetic story problem solving, its benefits were not evaluated. Procedural scaffolding was first evaluated with Pyrenees (Chi & VanLehn, 2010; Vanlehn & Chi, 2012), which required students to construct models using a version of goal reduction, which is a well-known general purpose reasoning strategy used in artificial intelligence applications (Russell & Norvig, 2009).

Pyrenees' procedural scaffolding was evaluated in a two-phase experiment. In the training phase, students learned to construct model of probabilistic systems. In the transfer phase, student learned to construct models of mechanical energy systems. Using the framework mentioned earlier, the findings were:

A. *Students often exhibit poor learning strategies.* True of the control conditions in both phases.
B. *Good learning strategies can be taught, often with little difficulty*. This could not be determined, because Pyrenees required students in the experimental condition to follow the learning strategy during the training phase.
C. *When students use the taught learning strategies, their domain learning increases compared to students who are not taught to use the learning strategies*. In both the training phase and the transfer phase, the experimental group acquired more domain knowledge than control group. The effect sizes were large ($d \approx 1.0$).
D. *When instruction in the learning strategy includes meta-cognitive and motivational components, students can often be prevented from reverting to poor learning strategies when the instruction ceases*. The experimental group tended to use the learning strategy during the transfer phase on difficult problems, but not on simple problems. However, even on simple problems, they did not use poor learning strategies.
E. *Specific learning strategies often have larger effect sizes than general purpose ones*. Not tested.

In summary, Betty's Brain and the Help Tutor used the standard methods of hints and feedback and their meta-tutoring succeeded in improving students' behaviour, but there were only weak improvements at best in their learning of the domain. Co-Lab's learning strategy increased performance, but the effect on learning was not measured. Pyrenees used procedural scaffolding, and its meta-tutoring caused large improvements in both behaviour and domain learning.

1.4. Our research questions

Our research questions are the same 4 questions that our predecessors have focused one:

A. Without instruction in good learning strategies, do students often exhibit poor learning strategies?
B. Can good learning strategies be easily taught?
C. When students use the taught learning strategies, does their domain learning increase compared to students who are not taught to use the learning strategies?
D. When instruction in the learning strategy ceases, do students revert to poor learning strategies?

Although these 4 questions about learning strategies are the central focus of the research, we are also interested in seeing if the time required to achieve adequate competence in model construction can be reduced from 5 or more hours to 2 for fewer hours.

Because Pyrenees was arguably more successful than the other methods, we chose to use procedural scaffolding as the basic method of teaching learning strategies. However, there is a major difference between our research problem and the one addressed by Pyrenees. Pyrenees assumed that students had already been taught all the relevant domain principles. Indeed, the students repeatedly selected principles from a menu. In contrast, when students construct system dynamics models, they are seldom taught domain principles in advance. They are instead asked to infer domain relationships from multimedia resources, common sense and/or experimentation. They do this while constructing the model. This allows modelling to be used as part of enquiry-based instruction where students construct the domain knowledge themselves. Having to infer domain relationships while learning to model may be one of the reasons why it takes students so long to master system dynamics model construction. Nonetheless, inferring quantities and their relationships is exactly the skill we want students to learn.

This suggests that if we require students to follow a procedure, as Pyrenees did, then they will learn that the key to modelling is a single non-mechanical step: figuring out the mathematical relationship among a set of directly related quantities. Once they have mastered that step, we expect that they will be able to construct models well, even if they don't follow the procedure. Thus, our research question is whether applying the Pyrenees approach of heavy-handed procedural scaffolding to the cognitive skill of model construction will cause deeper, more effective modelling skills to develop during the training phase. If the tutor does work in the training phase, our next question is whether the benefits will persist in the transfer phase. To answer these two questions, we developed an instructional system, called AMT, and conducted experiments to evaluate it. The following sections described the system, how it was implemented, and the evaluation at last.

The name "AMT" is an acronym for the overall project, which is called the Affective Meta-Tutoring project. The first objective of the project is to develop and test a meta-tutor that improves student's learning of modelling by using procedural scaffolding as well as the traditional feedback and hints. The results of that phase of the project are reported here. The second phase of the project will be to add an affective learning companion to the AMT system; hence the term "affective" in the project name. The second phase will be described further in the discussion section.

2. The AMT system's design and behaviour

This section has three parts. The first describes the student's task, and in particular, the graphical language in which they write models. The second section describes the tutoring system. The third section describes the meta-tutor and the strategy that it teaches students to use.

2.1. Constructing models in the AMT modelling language

In order to decrease the difficulty of learning how to construct system dynamics models, most prior work has used graphical modelling languages where a model consists of several types of nodes and links. These graphical languages are easier to learn than text-based languages (Löhner, Van Joolingen, & Savelsbergh, 2003). The traditional "stock and flow" language has two types of links, and one of them (the flow links) acts somewhat like nodes. In pilot studies, our high school students found this confusing, so we removed the confusing type of link.

In our modelling language, a model is a directed graph with one type of link. Each node represents both a variable and the computation that determines the variable's value. The inputs of that computation, which are themselves variables, are indicated by incoming links. In Fig. 2 for example, the computation for "births" requires the values of "growth rate" and "population." The value of a variable is a real number that can change over time, where time is represented discretely.

There are three types of nodes:

- A *fixed value* node represents a constant value that is directly specified in the problem. A fixed value node has a diamond shape. It never has incoming links.
- An *accumulator* node accumulates the values of its inputs. That is, its current value is the sum of its previous value plus its inputs. An accumulator node has a rectangular shape and always has at least one incoming link.
- A *function* node's value is an algebraic function of its inputs. A function node has a circular shape and at least one incoming link.

The students' task is to draw a model that represents a situation that is completely described by a relatively short text. For instance, Fig. 2 is a correct model for the following problem:

Rust destroys steel and can spread quickly. Suppose that you take a large sheet of steel, such as one that might be used as the roof of the boxcar on a train, and you put it outside in the weather. Suppose it starts with a spot of rust that is 10 square inches in area. However, each week the rust spot gets bigger, as it grows by 30%. Therefore at the end of the first week, the rust spot is 13 square inches in area. Graph the size of the rust spot over 10 weeks.

Such a text contains all the information that students need. However, it sometimes contains extra quantities (e.g., the rust spot at the end of the first week) that are not needed in the model. Students are asked to draw only the necessary nodes, so drawing a node for the rust spot at the end of the first week is an indication of shallow modelling.

2.2. The tutoring system

Many tutoring systems have a sequence of feedback and hints (VanLehn, 2006). When the student makes an entry, the tutoring system first just tells the student whether the entry is correct or incorrect. If the student asks for help again, the system gives a general hint. Subsequent requests for help on the same entry generate increasingly specific hints. Eventually, the final hint (called the "bottom-out hint") tells the student what the correct entry is. There is some evidence that the mid-level hints are not pedagogically useful (Muldner, Burleson, van de Sande, & VanLehn, 2011; Timms, 2007), so our tutoring system does not use them. Instead, it generates just the first and last members of the typical hint sequence. More specifically, it has a Check and a Give-up button. Clicking on the Check button causes the tutoring system to give minimal feedback: It colours entries red if they are incorrect and green if they are correct. Clicking on the Give-up button causes the tutoring system to fill in entries correctly, that is, to give a bottom-out hint. This section describes the AMT model editor, pointing out where the Check and Give-up buttons appear.

The system presents itself to the student as a large tabbed window (Fig. 3). The Model tab (shown in the figure) is for drawing models. The Situation tab shows a textual representation of the problem, and is illustrated by a static picture. The Instructions tab contains a slide deck that teaches students the basics of the modelling language, the user interface, the modelling process and the learning strategy. Students can access the Introduction and Situation tab at any moment during the construction of the model (cf. Section 4.4).

The Model tab has three buttons: Create Node, Run Model and Done. The Done button is disabled (grey) until the student has completed a problem successfully (i.e. created an accurate model of the system), at which time clicking on the Done button advances the system to the next problem.

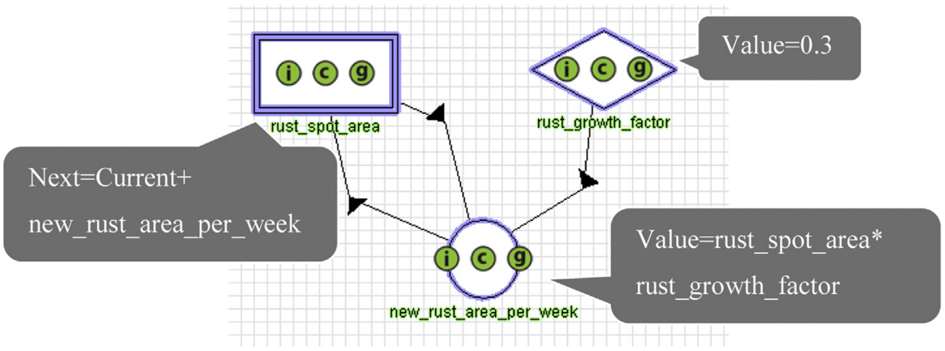
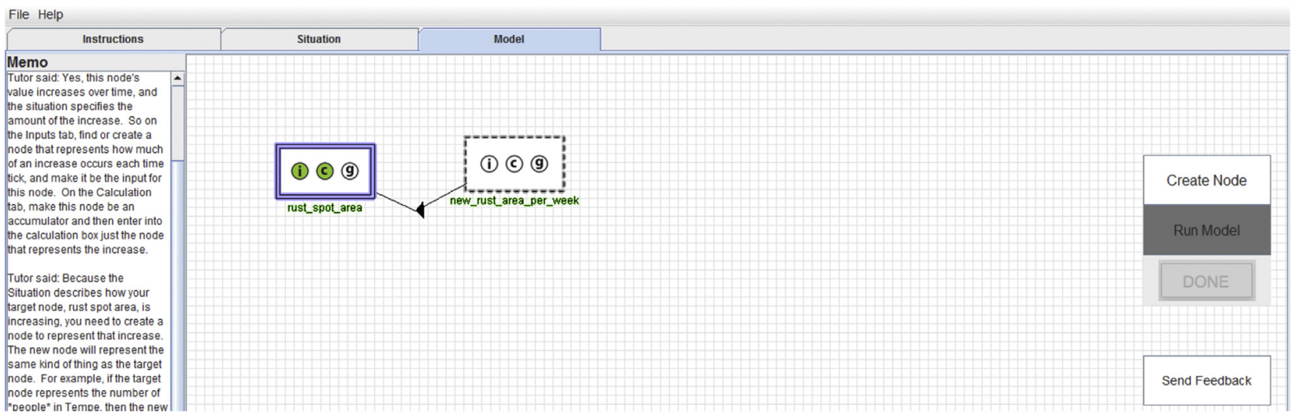

Value=0.3
rust_spot_area
rust_growth_factor
Next=Current+ new_rust_area_per_week
Value=rust_spot_area* rust_growth_factor
new_rust_area_per_week
i c g

Fig. 2. A model. The grey bubbles have been added to this figure in order to show how the value of each variable is calculated.



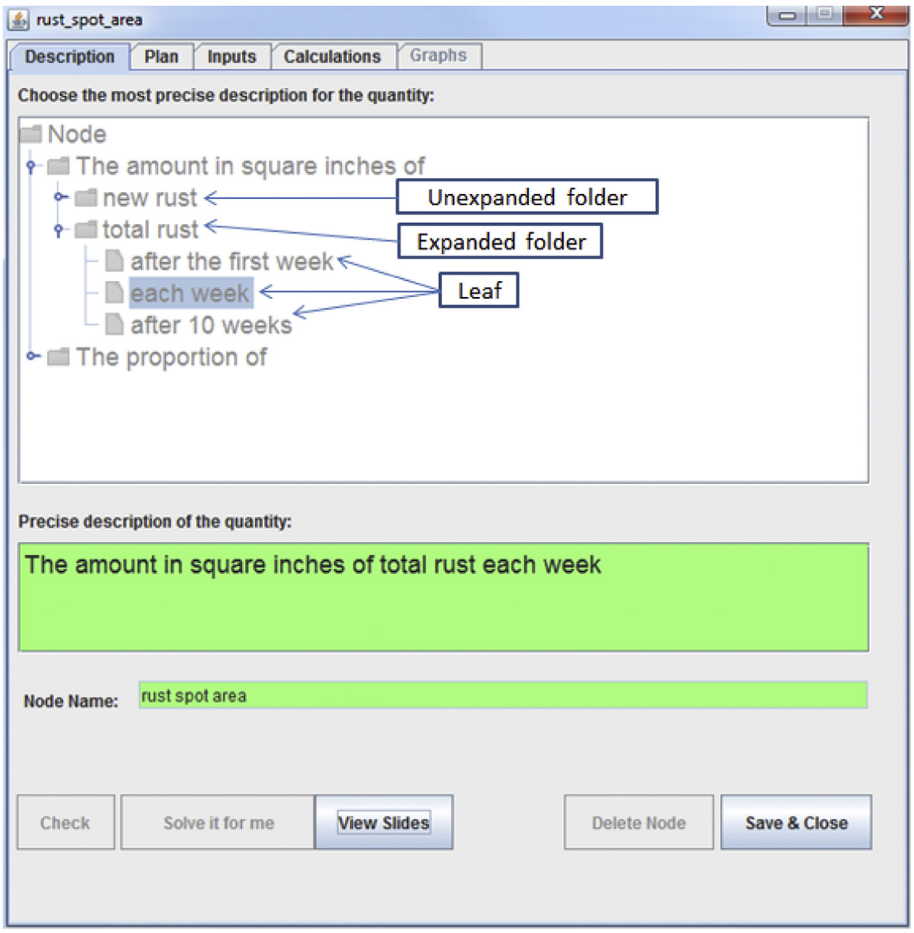
Tutor said: Yes, this node's value increases over time, and the situation specifies the amount of the increase. So on the Inputs tab, find or create a node that represents how much of an increase occurs each time tick, and make it be the input for this node. On the Calculation tab, make this node be an accumulator and then enter into the calculation box just the node that represents the increase.

Tutor said: Because the Situation describes how your target node, rust spot area, is increasing, you need to create a node to represent that increase. The new node will represent the same kind of thing as the target node. For example, if the target node represents the number of *people* in Tempe, then the new

Fig. 3. The tutoring system's screen.

The Create Node button creates a new node symbol in the model area and opens the Node Editor on it. The Node Editor (Fig. 4) is divided into five tabs: Description, Plan, Inputs, Calculations and Graph. Students can edit all tabs with the exception of the Graph tab, where a graph showing the evolution of the node's value over time is generated by the tutoring system once the model is run. To create a node, the student fills out the four tabs in order, left to right. Students can also change the information within a node by double clicking on the node shape within the model area, which opens the Node Editor, and then selecting the tab they wish to modify.

The Description tab was engineered to facilitate grounding, where "grounding" is the process of negotiating mutual understanding of the meaning of a term between two participants in a conversation (Clark & Brennan, 1991). In system dynamics model editors that are not used for tutoring, such as Stella, Vensim and Powersim, users merely type in the name of the node that they want. This allows them to use names such as "x" that do not match anything in the problem. While this is unproblematic for such editors, it makes it difficult or impossible for a



Choose the most precise description for the quantity:

Precise description of the quantity:

The amount in square inches of total rust each week

Node Name: rust spot area

Fig. 4. The node editor, showing the Description tab.

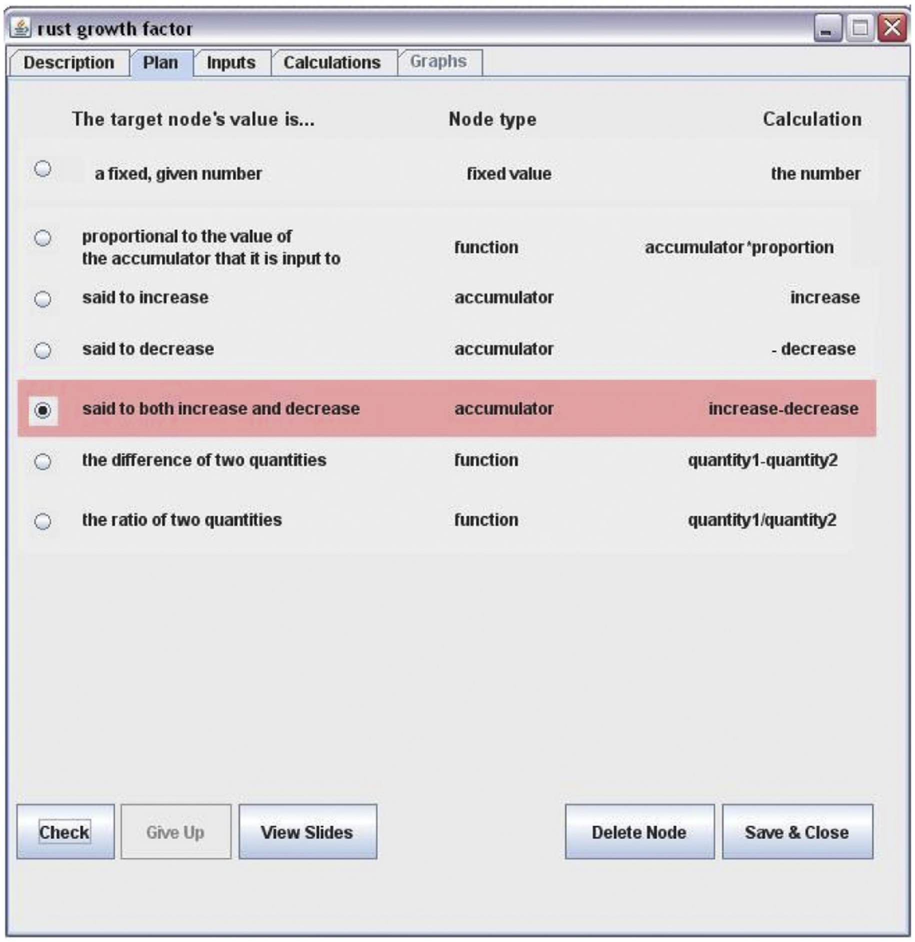
tutoring system to determine what quantity the student's node denotes, and hence the tutoring system cannot determine whether the node is defined correctly. Urging the students to choose adequately precise names is only partially successful. In one study, only 78% of the node names could be identified by the tutoring system (Bravo, van Joolingen, & de Jong, 2009). This is a case of bad grounding: the tutoring system did not understand the meaning of 22% of the students' terms.

On the other hand, some tutoring systems for system dynamics modelling, including the first version of this system, provide students with nodes that already have names but are otherwise undefined (VanLehn et al., 2011). Unfortunately, students often do not pay enough attention to the names, which can be rather similar. Students sometimes create models that are correct except that the names of two nodes, such as "rust growth factor" and "new rust area per week", have been switched. This is also a case of bad grounding: the student did not understand the meaning of the system's terms.

The Description tab of AMT, illustrated in Fig. 4, is intended to prevent bad grounding. First, the student walks through the tree in the Description tab located in the large box at the top of the window. Clicking on a leaf in the displayed tree selects both a description and a name for the node. Each problem has a different tree, and the tree's contents are engineered to make the student select among subtly different descriptions. After selecting a leaf, the student clicks on the Check button. If the selected description denotes a quantity that the system understands, and that quantity does not already have a node defined for it, then clicking on the Check button turns some boxes green, as shown in Fig. 4. Otherwise, the boxes turn red. Students may also click on the Give-up button that fills out the tab correctly but colours the boxes yellow. When the student exits the node editor, the node's name, which is shown beneath the node, is highlighted by the colour of the Description tab's boxes. Thus, a student who had given up on the Description tab would forever see yellow highlighting on the node's name. This feature is intended to discourage giving up.

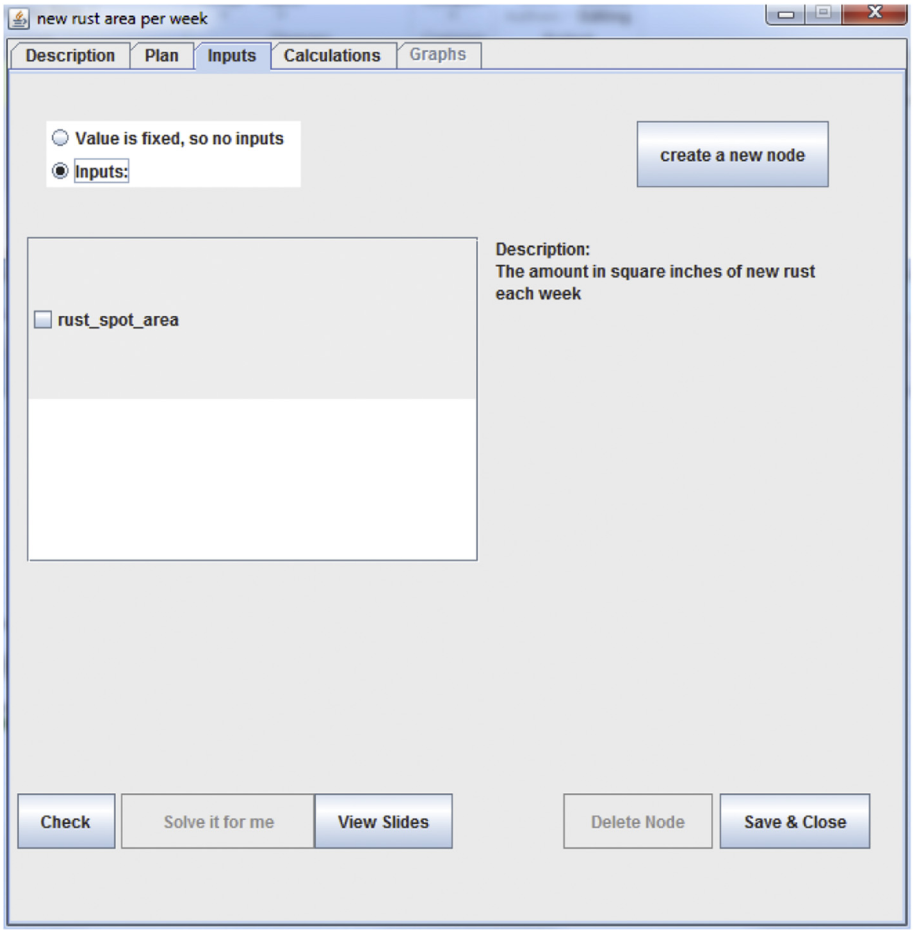
When the Description tab is completed correctly and its boxes are either green or yellow, then students can go on to the Plan tab. The Plan tab lets students choose among 7 different plans (Fig. 5). The Instruction tab describes the meaning of these plans and gives an example of each plan. After a selection is made, students may click on the Check button and see their selection coloured green (correct) or red (incorrect). Clicking on the Give-up button causes the correct plan to be selected and coloured yellow. Unlike the other tabs, filling out the Plan tab correctly is optional. Students can go to the Inputs tab regardless of how their plan selection is coloured, and they can skip the Plan tab entirely if they want.

The Inputs tab (Fig. 6) allows students to indicate whether the node's value is a fixed, given constant or if it is computed from other nodes' values. In the latter case, they click on "Inputs:" and choose some of the existing nodes as inputs, which causes links are drawn between those nodes and the current node. When students see that the input they want is not in the list because they have not yet created a node for it, they can click on the convenient "Create a new node" button, define the desired node using a pop-up version of the Description tab, then



	The target node's value is...	Node type	Calculation
○	a fixed, given number	fixed value	the number
○	proportional to the value of the accumulator that it is input to	function	accumulator*proportion
○	said to increase	accumulator	increase
○	said to decrease	accumulator	- decrease
●	said to both increase and decrease	accumulator	increase-decrease
○	the difference of two quantities	function	quantity1-quantity2
○	the ratio of two quantities	function	quantity1/quantity2

Fig. 5. The plan tab.



Description:
The amount in square inches of new rust each week

Fig. 6. The Inputs tab.

return to this Input tab by closing the pop-up. As always, the Check and Give-up buttons colour the student's choices in either red (incorrect), green (correct) or yellow (gave up). While red choices can be revised later on, green or yellow choices cannot be changed.

When the Inputs tab is filled out correctly, students can go to the Calculation tab (Fig. 7). If they had selected "Fixed Value" on the Inputs tab, then "has a fixed value" is checked here on the Calculation tab, and so all they need to do here is enter the number that is the value of the Fixed Value node. On the other hand, if they had selected "Inputs:" on the Inputs tab, then they must click either the "accumulates the values of its inputs" button or the "is a function of its inputs" button. This selection determines which form appears in the lower part of the tab. Fig. 7 shows the tab to fill for the Accumulator node type on the left, and the Function node type on the right. For both the Function and Accumulator nodes, the inputs appear initially in the "Available inputs:" box, and then move rightward into the calculation box as they are clicked. In this fashion, students enter a calculation.

As a model is being constructed, the state of the node's definition is indicated by little circular indicators ("i" for input, "c" for calculation and "g" for graph) and the node's outline (see Fig. 8). A dotted border indicates that the type of the node hasn't been defined yet. A blue border means that the node's definition is complete.

When all the nodes have blue borders, the student can click on the Run Model button. If there are syntactic errors in the model that prevent running it, a pop-up window describes them. Otherwise, Run Model colours the "g" indicators on every node either green or red, depending on whether the node's graph is correct or not. Opening the Graph tab of a node displays both the user's model's graph and the expected graph (see Fig. 9). Students can use the difference in the graphs as a clue to where the bug in the model could be found.

A model is considered completely correct when all the graphs match, in which case all the "g" indicators are green. At this point, students can click on the Done button and go to the next problem.

This is a step-based tutoring system (VanLehn, 2006) because it can give feedback on every step taken by the student. It only gives such feedback when the student clicks on the Check button. The feedback is minimal: just correct vs. incorrect. Absent are the usual sequences of hints that many step-based tutoring systems have. However, the "Give-up" button implements the bottom-out hint in that it gives away exactly what step the student should do at this point.

The AMT system has a mode, called *test mode*, which is used to assess students' skill at modelling. Although students can still debug their model by running it and seeing which graphs are correct, they cannot use the Check and Give-up buttons on any of the tabs that they fill out, with one exception. The Check button is always enabled on the Description tab. This is because the system and the student must agree on the meaning of nodes. If the system doesn't know which quantity is denoted by the student's node, then it can't know which graph is correct and can't colour the "g" indicators. In test mode, the Check button is enabled only on the Description tab, whereas in training mode, it is enabled everywhere. In test mode, the Give-up button is never enabled.

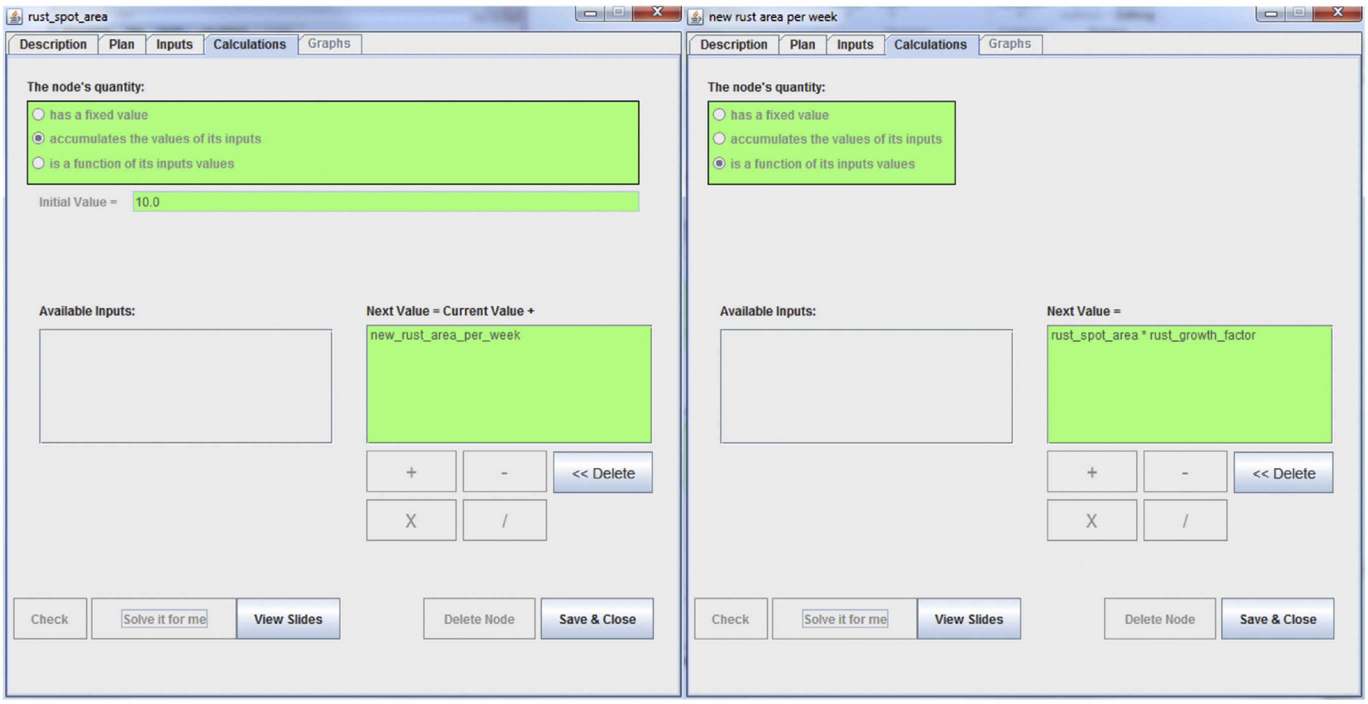


Fig. 7. The Calculations tab.

2.3. The meta-tutor

So far, only the tutoring system has been described. It is essentially just a model editor with Check and Give-up buttons. This section described the meta-tutor, which is the remaining module of the AMT system. This section begins by describing and motivating the learning strategy that the meta-tutor teaches, then describes how it teaches that strategy.

Like Pyrenees (Chi & VanLehn, 2010), the meta-tutor teaches students a simple, goal reduction procedure for constructing models called the *Target Node Strategy*. The basic idea is to focus on one node at a time (the *target node*) and complete all the tabs for this node before working on other nodes. As students complete the target node, they may create new nodes as a side effect, but the new nodes will only be named and not fully defined. These nodes are displayed with dotted borders. Thus, when students have finished the target node, they can pick any node that has a dotted border as the next target node, and begin working on defining it. When there are no more nodes with dotted borders, the model is complete.

There are sound reasons to think that the Target Node Strategy might help students learn more effectively, but it will take several paragraphs to describe them. These paragraphs also illustrate the distinction between deep and shallow modelling practices.

The key steps in the Target Node Strategy are filling out the Inputs tab and the Calculation tab. These steps correspond to the moment where students must analyze the given system information and determine the quantitative relationships between the target node's quantity and other quantities in the problem. That is, given a particular target quantity, students must find a set of quantities such that the

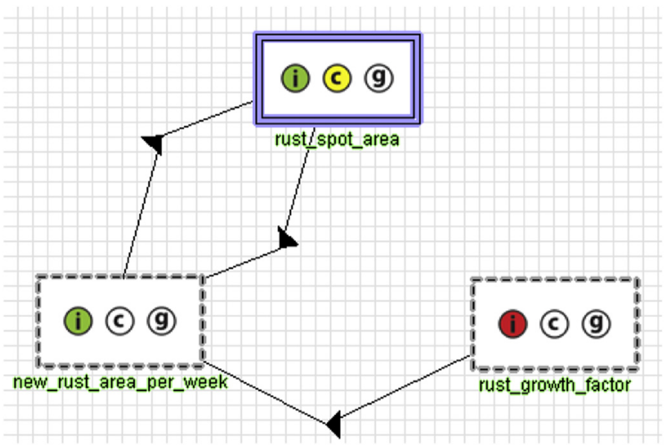
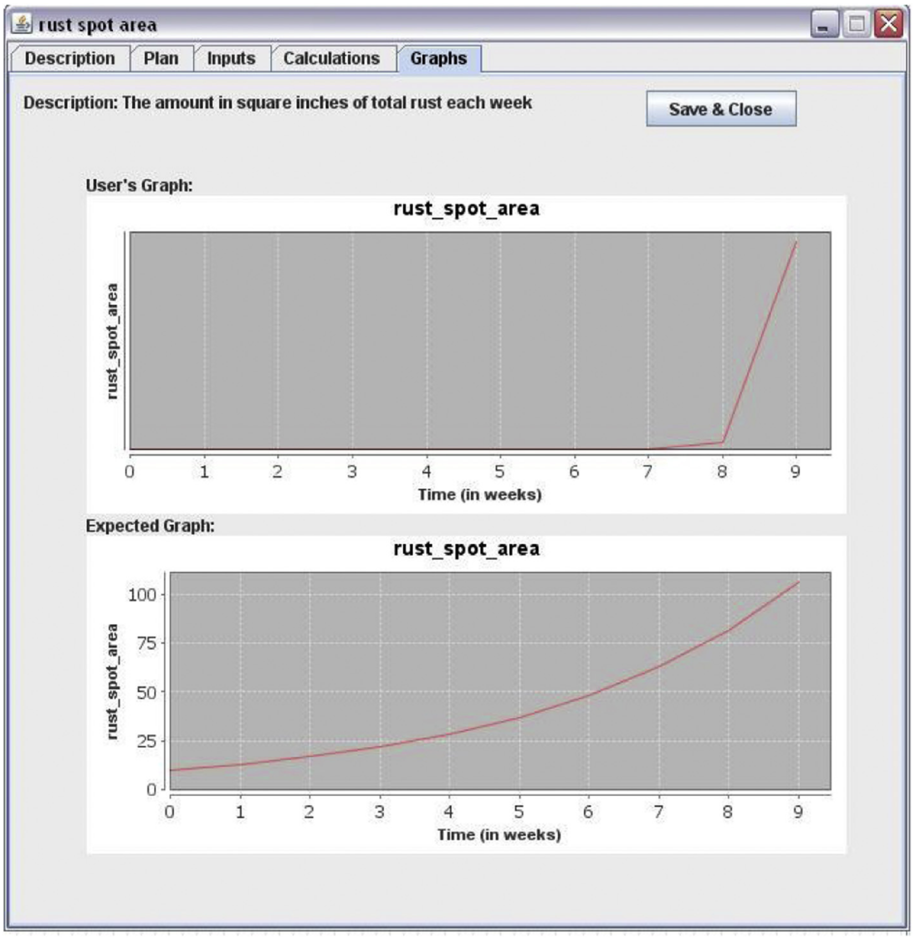


Fig. 8. An incomplete model.



Description: The amount in square inches of total rust each week

Fig. 9. The Graphs tab.

target quantity's value can be computed as a simple function of their values. The whole problem of constructing a model of a system can be decomposed into making a series of decisions like this one. One might call these decisions "atomic modelling problems", as they cannot be easily decomposed further.

For example, consider the system described earlier:

"*Rust destroys steel and can spread quickly. Suppose that you take a large sheet of steel, such as one that might be used as the roof of the boxcar on a train, and you put it outside in the weather. Suppose it starts with a spot of rust that is 10 square inches in area. However, each week the rust spot gets bigger, as it grows by 30%. Therefore at the end of the first week, the rust spot is 13 square inches in area. Graph the size of the rust spot over 10 weeks*."

Suppose students are following the Target Node Strategy and have decided that the first target node is the size of the rust spot. They create a node and select "rust spot area" as its description. Now they face an atomic modelling problem. How to determine the value of *rust spot area*? What we want students to think is something like, "I know the rust spot area is changing, and the value increases every week". This reasoning suffices for filling out the Plan tab correctly. Next the students need to fill out the Input tab, so they should think, "The rust spot area is increased by the rust produced during the week". There are no other nodes at the moment, so the student must click on "Create a new node." The student browses the available descriptions, and chooses the one that comes closest to the student's idea of "rust produced during the week." Clicking on this description defines the node *new rust area per week*. The student can then select this node as an input for the node *rust spot area.* Next comes the Calculation tab, which requires for the student to turn his/her qualitative understanding of the relationship into a formal, mathematical one: the student should decide that the next value of *rust spot area* is its current value plus *new rust per week*. This illustrates how the user interface decomposes the key reasoning into three major steps, corresponding to the Plan, Input and Calculation tabs. These correspond roughly to the phases of Co-Lab (Mulder et al., 2011) and Model-It (Metcalf et al., 2000). This paragraph also illustrates how the key reasoning should be done: by thinking hard about the system, its quantities and their interrelationships, and then abstracting the mathematical relationships from them. This is the "deep" way to solve an atomic modelling problem.

However, there are shallow ways to solve the atomic modelling problem as well. When students need to create nodes in order to fill out the Inputs tab, there are only a finite number of choices: the leaves in the tree of descriptions on the Description tab. Thus, they can work through all combinations until they have found the appropriate inputs. When the Check button is available and the problem is simple, this can be done rapidly. For instance, the rust spot area problem's Description tab has 7 possible descriptions, of which only 4 are legal quantities in the problem (1 is an extra node; 3 are necessary for the model). It will not take the student long to create all the legal nodes, and then test each one to see if it turns the Inputs tab green. Such extensive guessing is one “shallow” method for solving atomic modelling problems. The Give-up button is another. Other methods involve looking for keywords (e.g., “initially”) or patterns.

Many of our students’ shallow modelling methods involve abusing the Check button or the Run Model button, so their behaviour is a form of *gaming the system* (Baker et al., 2006; Baker, Corbett, & Koedinger, 2004), which is what the Help Tutor (Roll et al., 2011), Scooter the Tutor (Baker et al., 2006), and other tutoring systems have tried to address. However, studies of modelling that did not use tutoring systems have also noted this propensity of students to do shallow modelling (Alessi, 2000; Booth Sweeney & Sterman, 2000; Doerr, 1996; Mandinach & Cline, 1994b; Nathan, 1998; Zaraza & Fisher, 1999). Thus, we prefer to refer to the phenomenon as “shallow modelling” rather than “gaming the system.”

Teaching the Target Node Strategy does not *require* that student do deep modelling, so one might wonder why we think it could help students model better. However, it does decompose the whole problem of modelling a system into a series of atomic modelling problems, and even decomposes an atomic modelling problem into three major steps. Like Pyrenees, it teaches students that if they just master this one difficult but small skill (deep modelling applied to atomic modelling problems), then the rest of the problem solving is purely mechanical.

Having described and motivated the Target Node Strategy, it is finally time to describe how the meta-tutor teaches it. First, the Introduction slides describe the strategy briefly. Even students who use AMT with the meta-tutor turned off see this presentation of the strategy. Also when the meta-tutor is turned off, students can edit any node at any time. Moreover, they can fill out some of a node’s tab, then quit editing the node and come back to it later. This freedom is removed when the meta-tutor is turned on. Students are required to correctly fill out each tab before moving on to the next, and they must fill out all tabs before closing the node editor. When they are on the Description tab and choosing which quantity to create a node for, if they choose one that would not be selected by Target Node Strategy and yet it will eventually be include in the model, then the selection turns blue and a pop-up window says, “That quantity is in the correct model, but it is too early to define it now.” This message is typical of others that pop up when students stray from the Target Node Strategy. In short, the main job of the meta-tutor is simply to keep students on the one of the paths that are consistent with the Target Node Strategy.

In addition to requiring students to follow the Target Node Strategy, the meta-tutor enacts a bit more scaffolding. For completeness, this section describes the remaining forms of help.

When the meta-tutor is turned on, it complains if students appear to be guessing too much or giving up too early, just as the Help Tutor did. When a student clicks on the Check button on the same tab twice within 3 s and gets red both times, the meta-tutor complains that the student is guessing. If the student clicks on the Give-up button without filling out anything and checking it, the meta-tutor complains that the student is giving up too early.

Some of the training problems let students practice debugging an incorrect model. They present a model that will run but generates incorrect graphs. This kind of situation occurs frequently when the system is in test model (the tutor and meta-tutor are turned off). When the model has been run, students see that some nodes have red “g” indicators and some have green ones. They face a decision of where to start looking for an error in the model. The Introduction slides teach all students two heuristics:

- When possible, start by examining a node that has a red “g” indicator and no incoming links from nodes with red “g” indicators. Such a node is guaranteed to have an error inside it.
- Avoid editing nodes that have green “g” indicators, because if the graph is correct, the node’s definition is probably correct.

When the meta-tutor is on and students are working on a debugging problem, it constrains them to obey these two heuristics.

In summary, the meta-tutor actually uses three types of scaffolding: (1) it requires students to follow the Target Variable Strategy; (2) it complains when they abuse the Check or Give-up buttons; and (3) it teaches some heuristics for locating errors in buggy models. Although (1) is procedural scaffolding, for convenience, we refer to this collection as “the meta-strategy.”

3. System architecture and implementation

The AMT system can be divided into two parts: the tutor and the meta-tutor. The tutor’s main functionalities included drawing a model, checking correctness and running the model. It is an example-tracing tutor (VanLehn, 2006) in that an author must provide a correct model with each problem; the students’ work is checked against that model. On the other hand, the meta-tutor is driven by algorithms (e.g., the Target Node Strategy) that work for any problem. The student’s actions are check against the actions selected by the meta-strategy.

Although the meta-tutor and the tutor belong to the same Java project, they don’t share objects in the memory. This made it possible to develop the tutor and meta-tutor programme independently. It also facilitates data analysis as will be explained later. More details about the tutor’s architecture and implementation are presented in (Gonzalez-Sanchez, Chavez-Echeagaray, VanLehn, & Burleson, 2011). This section describes the meta-tutor only.

Between the tutor and meta-tutor, there are three kinds of communication, each with its own channel as shown in Fig. 10:

(1) Every action made by student is sent to the meta-tutor via the *activity* channel.
(2) Before executing certain user actions, such as closing a node, the tutor sends a message to the meta-tutor via the *block* channel and meta-tutor responds to the tutor, indicating whether the action should be blocked or not.
(3) Whenever the meta-tutor detects that the student needs an unsolicited hint, it sends a command to the tutor via the *message* channel to tell it to initialize the tutorial dialogue. The tutor sends back the student’s response. Based on the response, the meta-tutor tells the tutor either to display another dialogue box or to close the dialogue.

The meta-tutor is a production system, implemented in Java using Drools Expert (http://www.jboss.org/drools/drools-expert.html). The production system implements two major functionalities: requiring the student to follow the Target Node Strategy and discouraging gaming. The following sections describe the implementations of each function.

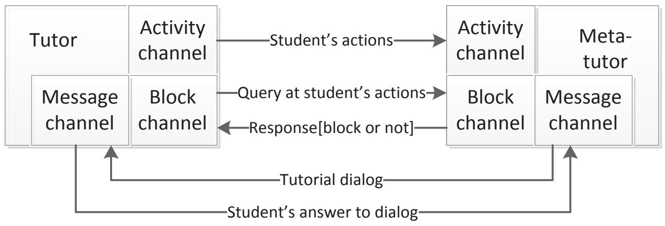


Fig. 10. How the tutor and meta-tutor communicate with each other.

3.1. *Constraining students to follow the Target Node Strategy*

One main function of the meta-tutor is to constrain the student to follow the Target Node Strategy. In order to do so, it implements the algorithm shown in Fig. 11. Aside from a little bookkeeping, the procedure's actions are either "Prompt and constrain the student to do <step>," or "Constrain the student to do <step>".

Prompting means that as soon as the problem state allows a step to be done, the meta-tutor gives the student an unsolicited hint about the step. Here are examples of production rules that implement prompting:

- IF the activity channel reports that the student's action is "closed a node"
  AND there are at least two nodes that can become the next target node
  THEN
  send a command via the message channel to show a dialogue box that lists the nodes and ask which one the student wants to choose as the target node.
- IF the context is that the node editor is open and the current tab is the plan tab,
  AND the activity channel reports that the student's action is "clicked on the Check"
  AND the correct plan of this node is "a fixed, given number"
  AND the student has selected this plan
  THEN
  send a command via the message channel to show a message that says that the next step is to fill in the Inputs tab by clicking on "Value is fixed, so no inputs."

Constraining the student to do a step means that if the problem state is such that the student should do a certain action, and the student tries to do another action instead, then the meta-tutor blocks the student's action from occurring and pops up a message indicating the right action to do. Some examples of production rules that implement constraining are:

- IF the context is that the student is in the model tab and the node editor is not open
  AND the student's action is clicking on a node in order to open it in the node editor
  AND the node that student is trying to open is not the target node
  THEN
  send via the block channel the message "Please focus on your target node, <target node name>" and block opening the clicked-on node.
- IF the context is that the node editor is open
  AND the student's action is closing the node
  AND the node's calculation tab is not finished yet
  THEN
  Send via the block channel the message "Please finish the target node, <target node name>, before leaving the editor" and block closing of the node editor.

In order for the meta-tutor to track the student's progress and respond appropriately, its rules need to have up-to-date information about the state of the problem. Using the activity channel, the tutoring system sends all student actions to the meta-tutor, which converts some of them (e.g., not node movements) into elements in the production system's working memory. The working memory is also used to store the identity of the target node, the contents of the sought set and other bookkeeping information that is kept updated via production rules.

3.2. *Discouraging gaming*

As mentioned earlier, the Give-up and Check buttons can be easily misused by students to construct models without thinking deeply or learning, e.g., "gaming the system" (Baker et al., 2006; Baker, Corbett, Koedinger, et al., 2004). Although constraining students to follow the Target Node Strategy may discourage gaming, the meta-tutor also looks for patterns of student actions that indicate gaming is occurring. When it sees such a pattern, it pops up a warning message but does not block the student's actions.

Detecting and responding to gaming of the Check button is implemented by the finite state machine shown in Fig. 12. The students are always in one of 5 states. When they start working on a problem, they start in state S1 (the node editor is not open). Clicking on a node opens the node editor and moves to state S2 (no check yet). Edits to the contents of the open tab can occur in any of the states except S1, and are not shown in Fig. 12, as they merely add a loop from a state back to itself. The arc label "Wrong check" means that the student clicked on the Check button and got red, indicating the tab was not filled out correctly. The arc label "Got the answer right" means the student either

1. Initialize the target node to the top level goal (i.e., the quantity the problem wants graphed). Initialize the sought set to null.
2. Constrain the student to create a node and fill out its Description tab to correspond to the top level goal quantity.
3. Constrain the student to fill out the Plan tab correctly.
4. Prompt and constrain the student to fill out the Input tab correctly. Add any newly created nodes to the sought set.
5. Prompt and constrain the student to fill out the Calculation tab correctly.
6. Constrain the student to close the node editor.
7. If there are no nodes in the sought set, then prompt and constrain student to run the model. If there is just one node in the sought set, then set it as the target node and tell the student. If there are two or more nodes in the sought set, ask the student which one should be the target node and set it to be the target node.
8. Remove the target node from the sought set.
9. Prompt and constrain the student to create a node and fill out its Description tab to correspond to the target node.
10. Go to step 3 above.

Fig. 11. The Target Node Strategy implemented by the meta-tutor.

clicked on the Check button and got green, or clicked on the Give-up button. Whenever the student enters state S4 (gaming detected), the meta-tutor sends a randomly chosen message such as, “Guessing so quickly wastes the opportunity to learn.”

In short, because the main job of the meta-tutor was to teach students to stay on a certain path defined by the meta-strategy by blocking off-path actions, and the meta-strategy was simple and easily defined, the only significant technical challenge was integrating the meta-tutor with the tutor. The tutor was implemented with a standard model-view-controller architecture, and the meta-tutor essentially was given a chance to intervene after the user’s actions had actuated the control and before the model was updated.

At this point, the design and implementation of the AMT system have been defined, so it is time to consider whether it works. That is, does meta-tutoring improve students’ learning compared to tutoring without meta-tutoring?

4. Evaluating the meta-tutor

So far, we have conducted 5 studies of the AMT system. In the first two studies, which were conducted in the summer of 2010, students used the tutoring system without the meta-tutor. This led to significant changes in the design of the tutoring system (VanLehn et al., 2011), the materials and the experimental procedure (VanLehn et al., 2011). The two studies also produced data revealing students’ deep and shallow modelling practices, and thus allowed us to design the meta-tutor. This article presents the results of the next three studies. There were slight changes to the tutor and the experimental procedure in between the studies 3 and 4, which were conducted in summer 2011. Analysis of the data from these two studies over the next year led to significant changes to the AMT system, such as adding the Plan tab, which led to study 5, which was conducted in summer 2012.

All the three studies had two phases, training and transfer, as did most of the studies reviewed earlier. During the training phase, a meta-tutor taught students both to use a good learning strategy (adapted from Pyrenee’s) and to avoid using poor learning strategies. During the transfer phase, the meta-tutor and the tutor were turned off, thus allowing us to measure both domain learning and spontaneous use of the learning strategy.

4.1. Research hypotheses

Recall that our framework, adopted from the literature on learning strategies, poses four research questions:

A. Without instruction in good learning strategies, do students often exhibit poor learning strategies?
B. Can good learning strategies be easily taught?
C. When students use the taught learning strategies, does their domain learning increase compared to students who are not taught to use the learning strategies?

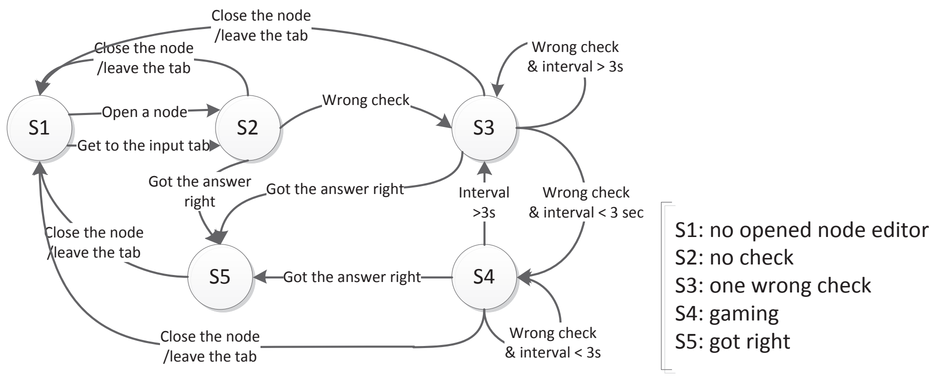

Close the node /leave the tab
Close the node /leave the tab
Wrong check & interval > 3s
Open a node
Get to the input tab
Wrong check
S1
S2
S3
Got the answer right
Got the answer right
Interval >3s
Wrong check & interval < 3 sec
Close the node /leave the tab
S5
Got the answer right
S4
Close the node /leave the tab
Wrong check & interval < 3s
S1: no opened node editor
S2: no check
S3: one wrong check
S4: gaming
S5: got right

Fig. 12. Finite state machine for gaming the Check button.

D. After instruction in the learning strategy ceases, do students revert to poor learning strategies?

Our observations during studies 1 and 2 answered question A by demonstrating that our students often use poor learning strategies when they are not taught good ones. Moreover, because the meta-tutor requires students to follow the Target Node Strategy during the training phase, question B cannot be addressed by our studies. This leaves us to focus on C and D.

Addressing question C requires that we clearly define the expected domain learning. As mentioned earlier, two entirely different instructional objectives are associated with model construction: Using model construction to learn domain concepts and principles (usually in a science classes) and learning how to construct models (usually in math classes). In these studies, we focus only on the second objective, so "domain learning" in question C means learning how to do deep model construction. Moreover, question C can be asked of both the training phase, where the learning strategy is taught, and the transfer phase, when the meta-tutor and tutor are turned off. Thus, the specific hypotheses we tested in studies 3, 4 and 5 are:

1. In the *transfer* phase, do meta-tutored students display deep modelling more frequently than students who were not meta-tutored?
2. In the *training* phase, do meta-tutored students display deep modelling more frequently than students who do not receive meta-tutoring?
3. In the *transfer phase*, do meta-tutored students follow the Target Node Strategy more frequently than students who were not meta-tutored?

Although we predict positive answers for all three questions, there is a caveat concerning the third hypothesis. The instruction used in studies 3, 4 and 5 did not provide much meta-cognitive and motivational encouragement to use the learning strategy, because we are developing that part of the instruction for use in later studies. Thus, we have low expectations for hypothesis 3.

4.2. Method

Students were randomly assigned to one of two conditions: with and without meta-tutoring. The difference between the conditions occurred only during a training phase where students learned how to solve model construction problems. The meta-tutor group solved problems with the meta-tutor turned on, while the control group solved the same problems with the meta-tutor turned off. During the training phase, all students could use the Check and Give-up buttons at any time.

In order to assess how much students learned, a transfer phase followed the training phase. During the transfer phase, all students solved model construction problems with almost no help. That is, the meta-tutor and the Give-up button were turned off, and the Check button was turned off everywhere except on the Description tab where it remained enabled in order to facilitate grounding, as mentioned earlier.

Because system dynamics is rarely taught in high school, the procedure did not include a pre-test in modelling dynamic systems.

In order to provide a motivation similar to that which occurs in school, we told students that prizes would be awarded to students who solved the most problems during the transfer phase. In particular, we repeatedly told them to use the Check and Give-up buttons judiciously during the training, trying always to learn as much as possible, so that they could rapidly solve problems during the transfer phase when the Check and Give-up buttons would not be available.

4.3. Participants

There were 34 student participants in the first experiment, 44 students participated in the second experiment, and 34 students in the third experiment. No person participated in more than one experiment. All were high school students who were participating in summer camps at our university.

4.4. Procedure

The three experiments followed the same procedure. Each lasted two and a half hours and had two main phases, training and transfer.

Sensors recorded students' physiological states throughout both the training and transfer phases. The sensors were: wireless skin conductance bracelets, facial expression cameras, posture-sensing chairs, and pressure sensitive mice. Periodically, a window would pop up and ask students to report their affective state. These data are being used to develop algorithms for affect detection, and will not be discussed further here. Also in the first two experiments, but not the third, students were asked to speak their thoughts aloud into a headset microphone, and their verbal protocols were recorded by screen-capture software.

The summer camp students were available for only a fixed period of time and all needed to be kept occupied productively during that period. Thus, each phase of the experiment lasted a fixed period of time, and students solved as many problems as they could during that time.

We obtained parental consent for minors prior to the experiment. During the experiment, student gave informed consent, filled out a background questionnaire, donned sensors and started the training phase. The training phase lasted a total of 75 min. It consisted of first studying a sequence of PowerPoint slides that introduced them to the user interface, model construction and the Target Node Strategy, and then solving a sequence of model construction problems. The introduction slides remained available while students were solving problems. During pilot testing, we noticed that some students spent a long time studying the introduction then rarely referred back to it, whereas other studied the introduction briefly and referred back to it frequently as they solved problems. Thus, we let students decide how to allocate their 75 min between the studying introduction and solving the training problems. The training phase was followed by a 15-min break where students went to another room and had a snack. After the break, all the students began the transfer phase, where both conditions were identical. The transfer phase lasted for 30 min, and a debriefing of the students followed.

Students in both conditions solved the same problems in the same order. Except for a few debugging problems during the training phase, where student were asked to correct a given faulty model, both training and transfer problems required students to construct a correct model, which is defined as a model whose graphs match graphs of correct values. When the model was correct, students were allowed to go on to the next problem. This procedure was followed for both the training and transfer phases.

The only procedural difference between the training and transfer phases was the amount of scaffolding available. During the training phase, the Check and Give-up buttons were enabled, and the meta-tutor was active for students in the meta-tutor condition. During the transfer phase, the Check button was enabled only on the Description tab because it was necessary for grounding, as discussed earlier. None of the other scaffolding was available during the transfer phase.

4.5. Measures

To test the hypotheses and answer the research questions, 7 measures were defined by extracting information about student's interaction with software from students' log files. The measures are described in this section.

Hypothesis 1 is that the meta-tutored students will use deep modelling more frequently than the control students during the transfer phase. Deep modelling is not easy to measure directly, so we used three indirect indicators:

- *The number of the "Run model" button presses* for completing one problem in the transfer phase indicates how much help the student needs from the system. Deep modellers should be able to complete problems using the Run model button only a couple of times per model.
- Another measure was *the number of extra nodes* created during the transfer phase, where extra nodes are defined as the nodes that can be legally created for the problem but are not required for solving the problem. Deep modellers should realize that these quantities are irrelevant and therefore avoid modelling them.
- *The number of problems completed* during the 30 min transfer period was considered to be an indicator of deep modelling with the assumption that it would be faster for students to solve atomic modelling problems deeply than shallowly.

Hypothesis 2 is that meta-tutored students should use deep modelling more frequently than the control group students during the training phase. The three dependent measures used to evaluate this hypothesis are described next.

- *Help button usage*: Ideally, a deep modeller would fill out a tab, click on the Check button and the tab's fields would turn green (correct). A student who is trying to do deep modelling but hasn't mastered the skill might have to click the Check button twice, thinking hard before each attempt, in order to get them right. On the other hand, shallow modellers might click on the Check button repeatedly as they guess or use the Give-up button. To measure usage of the help buttons, the following measure was calculated:

$$\textit{Help usage} = (n_{wc} + 3n_{gu})/n_{rn}$$

where n_{wc} is the number of Check button presses that yielded red, n_{gu} is the number of Give-up buttons the student clicked, and n_{rn} is the number of nodes required by the problem. The "3" here is a weight to make n_{gu} be roughly comparable to n_{wc}. The denominator represents the number of nodes required for a correct, minimal solution rather than the actual number of nodes the student completed. Should the prediction for Hypothesis 2 hold true, meta-tutored students' usage of the help button should be lower than the control students' one.

- *Correct on first Check*: This measure for Hypothesis 2 is a traditional one in tutoring system research. It represents how frequently students succeed on their first attempt at filling out a tab. That is, what proportion of the tabs turned green when students first clicked on the Check button? Deep modellers should get almost everything right the first time, whereas a student who is confused or guessing might rarely get elements right on the first try. Unfortunately, this measure could only be applied to experiment 5, as insufficient log data was kept during the 2011 experiments.
- *Training efficiency*: This measure is based on the (now dubious) assumption that deep modelling is faster than shallow modelling. Speed is often measured by counting the number of problems solved during a fixed training period. However, our problems' solutions varied in their complexity. Some had many nodes and some had few nodes. Moreover, students could always use the Give-up button, which does part of the problem solving for them. There was one Give-up button per tab, and completing a node requires filling out three tabs, so clicking on three Give-up buttons per node would solve the problem. Thus, a better measure of the amount of problem solving accomplished than "problems completed" is the number of tabs the student completed without using the Give-up button, which is given by:

$$\textit{Training efficiency} = 3n_{cn} - n_{gu}$$

where n_{cn} is the number of nodes that the student completed correctly (so $3n_{\mathrm{cn}}$ is the number of tabs), and n_{gu} is the number of Give-up buttons the student clicked. The prediction for Hypothesis 2 is that the training efficiency of meta-tutored students should be higher than the training efficiency of control students.

Hypothesis 3 is that the experimental group, which was required to follow the Target Node Strategy during training, would continue to use it during the transfer phase. To evaluate this hypothesis, we calculated the proportion of student steps consistent with the target node strategy. The algorithm of Fig. 12 describes how it is calculated.

5. Results

Experiments 1 and 2 found, as expected, that students often exhibit shallow learning strategies (VanLehn et al., 2011a). They also led to many revisions in the software and the experimental procedure (VanLehn et al., 2011b). However, both experiments had only one condition, and it did not include the meta-tutor. This section reports on comparisons of the system with the meta-tutor turned on and turned off.

5.1. Experiment 3 results

Experiment 3 was conducted in June 2011. Of the 34 participants, some students' data needed to be omitted. Probably because there were too many introduction slides (101 in total), 11 out of 34 students were still in the introduction phase at the break, and thus had no time in the training phase where the manipulation occurred. These 11 students were omitted from the analyses. Two students, one in each condition, performed much better than others. Both students' training measures and test measures were greater than three standard deviations from the means. These two students were also excluded from the analyses. The final number of students left for each group was 11 (control) and 12 (meta-tutor). All the analyses below were computed based on these 23 students.

5.1.1. Hypothesis 1 (transfer phase deep modelling)

Run Model button usage: Of the 23 students, there were only 9 Meta-tutored students and 4 control students that finished the first problem in the transfer phase. Among the 13 students, meta-tutored students used the Run Model button 4.88 times per problem, while students in control group used it 6.67 times. Due to the limited number of subjects, it is not surprising that the difference is not significant ($p = 0.72$, $d = 0.12$). This *T*-test, and all other tests reported here, are two-tailed.

Extra nodes: The first problem of the transfer phase allowed 2 extra nodes, thus allowing us to measure the number of extra nodes created by students who completed that problem. Meta-tutor students defined 0.44 (SD = 0.8) extra nodes vs. 0.75 (SD = 0.96) extra nodes for the control students. The difference was not reliable ($p = 0.61$, $d = 0.58$) although it was in the expected direction.

Number of problems completed: The average number of problems completed during the transfer phase was 0.82 (SD = 0.60) for the meta-tutor students vs. 0.35 (SD = 0.52) for the control students. The difference was marginally significant ($p = 0.057$) even though the effect size was large ($d = 0.88$) and in the expected direction. These figures show that on average, students completed less than one problem. More specifically, 9 meta-tutored and 4 control students finished one or more problems, which was a marginally reliable difference ($\chi^2 = 3.486$, $p = 0.06$).

In short, trends in the data support Hypothesis 1 but the differences were not reliable, probably due to the small sample size.

5.1.2. Hypothesis 2 (training phase deep modelling)

Help button usage: This measure is a weighted sum of help button uses divided by the total number of nodes completed. On this measure, the meta-tutor students averaged 4.78 (SD = 2.25) vs. 7.03 (SD = 3.44) for the control students. The difference was marginally significant according with a large effect size ($p = 0.08$, $d = 0.82$).

Training efficiency: The average training efficiency measure the amount of correct work done by the student during the fixed-length training period. For the meta-tutor group, training efficiency was 13.00 (SD = 6.94) vs. 11.45 (SD = 6.77) for the control group. The difference was not significant ($p = 0.30$; $d = 0.23$).

Thus, although there was a trend in the data supporting Hypothesis 2, the reliability was poor, perhaps due to the small sample size.

5.1.3. Hypothesis 3 (meta-strategy usage)

Hypothesis 3 was that meta-tutored students would voluntary the Target Node Strategy during the transfer phase more frequently than the student who was not meta-tutored. During the transfer phase, 0.39 (SD = 0.35) of the meta-tutor students' steps matched the Target Node Strategy's steps, vs. 0.34 (SD = 0.30) for the control students. This difference was quite small ($p = 0.70$, $d = 0.16$), suggesting that hypothesis 3 may be false for this study.

5.1.4. Summary of results and next step forward

Although there were some trends in the expected directions, the students in both conditions performed quite poorly. This was probably due to the large number of slides in the introduction phase as well as the difficulty of the tasks themselves.

In order to increase the number of training and transfer problems solved by students, thus allowing us to differentiate their performance statistically, we reduced the number of slides from 101 to 64, and simplified some of the tasks. As the second experiment confirmed, these changes led to an improvement on the performance students in both phases.

5.2. Experiment 4 results

Experiment 4 took place in July 2011. Data from all 44 participants (22 in each condition) were used in the analyses here.

5.2.1. Hypothesis 1 (transfer phase deep modelling)

Run Model button usage: To complete the first problem in the transfer phase, meta-tutored students used the Run model button 3.05 times on average. In comparison, students in the control group used the Run model button 5.13 times. However, the difference was not significant ($p = 0.31$, $d = 0.32$). Because the standard deviation was high (SD = 6.77) due to extreme values, we divided all the students into two types with the threshold being 2, which is the median. The students who used the run model button once or twice were considered deep modellers. The rest of the students were considered shallow modellers. Chi-square test is then used to compare the number of deep modellers in meta-tutored group to the number in control group. There was a trend in the expected direction, but the significance was marginal ($\chi^2 = 3.36$, $p = 0.067$).

Extra nodes: Because most of the students finished at least two tasks in the transfer phase (only 3 students in the control group did not) and the second task allowed up to two extra nodes, we used the second task to count extra nodes. As predicted by Hypothesis 1, meta-tutored students produced fewer extra nodes (0.27, SD = 0.70) than control students (0.95, SD = 1.03). The difference was significant with a large effect size ($p = 0.02$, $d = 0.80$).

Problems completed: The meta-tutor students solved 3.27 (SD = 1.03) transfer problems vs. 3.23 (SD = 1.57) for the control students. The difference was small and not significant ($p = 0.65$, $d = 0.04$).

5.2.2. Hypothesis 2 (training phase deep modelling)

Help button usage: Consistent with Hypothesis 2, meta-tutor students' help button usage averaged 2.35 (SD = 1.75) vs. 3.55 (SD = 1.85) for the control students. The difference was significant ($p = 0.04$, $d = 0.68$).

Training efficiency: Contrary to Hypothesis 2, the control students had higher training efficiency 72.77 (SD = 32.12) than the meta-tutor students, 54.36 (20.17), and the difference was marginally significant ($p = 0.05$, $d = -0.70$).

These results suggest that meta-tutor students did increase deep modelling in the training phase than the control students, but they also moved slower than control students.

5.2.3. Hypothesis 3 (use of Target Node Strategy)

Missing data precluded testing this hypothesis in experiment 4.

5.2.4. Summary of results, revisions and the next iteration

Both hypothesis 1 (transfer) and hypothesis 2 (training) were supported, albeit by one measure each. We were not satisfied with the pace of the students in the training phase, especially the meta-tutored students. Watching the screen-capture video suggested that meta-tutored students spent a lot of time in reading and answering the tutorial pop-ups. Students also became "stuck" (Burleson & Picard, 2007) when filling out the current Inputs tab and needing to create new nodes. In order to do so, students had to close the node editor and click on the Create Node button on the canvas in the Model tab. However, many students could not realize that they had to close the node editor. They complained that no button in the Inputs tab could help them get out of the stuck state. Thus, we spent several months refining both the tutoring system and the meta-tutor. We installed the Plan tab in order to reduce the time spent reading the meta-tutor's pop-ups. We also improved the interface, adding a "create a new node," button on the Input tab with the goal of reducing students' state of stuck in this stage.

The transfer phase proved to be extremely challenging for the students. This was evident in the relatively small number of problems solved as well as direct observation of the students. The null result on our productivity measure, number of problems solved in the transfer phase, might be due to students in both conditions becoming discouraged and ceasing to try hard. Thus, we modified the transfer phase so that besides colouring the "g" indicator, the system coloured the "i" indicator and "c" indicator as well to show the correctness of the corresponding tabs. This was intended to make it easier to locate errors while leaving unchanged the logic required for fixing the errors, and thus allowing students to make faster progress through the test problems while still allowing us to assess their skill.

5.3. Experiment 5 results

Experiment 5 was conducted in June 2012. Of the 34 participants, data from 33 were used in the analyses below (16 in the control group and 17 in meta-tutor group). One student was excluded due to his extraordinary performance. He finished all 7 problems in the transfer phase well before the end of the transfer phase, while the second fastest person only completed 4 problems.

5.3.1. Hypothesis 1 (transfer phase deep modelling)

Run Model button usage: On average, students in the control group used the Run model button 7.76 times, while meta-tutored students used the Run model button 7.82 times. So they almost had the same performance ($p = 0.98$, $d = -0.0093$).

Extra nodes: Most of the students finished at least two tasks in the transfer phase (one in each group didn't), so we again used the second task to measure extra nodes. As predicted by Hypothesis 1, the meta-tutor students produced fewer extra nodes (0.88, SD = 0.96) than the control students (1.13, SD = 0.99). However, the difference was not reliable ($p = 0.47$, $d = 0.26$).

Problems solved: Contrary to Hypothesis 1, the meta-tutor students solved 2.18 (SD = 0.53) transfer problems vs. 2.56 (SD = 0.78) for the control students. Students in the control group outperformed meta-tutored student with marginal significance ($p = 0.094$, $d = -0.57$).

Once again, the meta-tutor students tended to work more slowly than the control students. This time, there was only a trend to show that they might be doing more deep modelling than the control students.

5.3.2. Hypothesis 2 (training phase deep modelling)

Help button usage: As expected, meta-tutored students' help button usage averaged 3.92 (SD = 2.19) vs. 6.13 (SD = 2.73) for the control students. The difference was significant with a large effect size ($p = 0.016$, $d = 0.89$).

Correct on first attempt: To provide an additional test of Hypothesis 2, we calculated the percentage of time that the first Check on a tab was correct. Meta-tutored students achieved a higher percentage (0.77, SD = 0.068) than control students (0.68, SD = 0.11), and the difference was significant with a large effect size ($p = 0.015$, $d = 0.98$).

Training efficiency: Meta-tutor students scored 73.18 (SD = 27.53), a little bit higher in training efficiency than control students, 68.88 (SD = 17.16), but the difference was not reliable ($p = 0.59$, $d = 0.19$).

So students in both groups kept the same pace in the training session this time, and there was strong evidence that the meta-tutor students were engaged in deeper modelling than the control students.

5.3.3. Hypothesis 3 (use of Target Node Strategy)

Contrary to hypothesis 3, the meta-tutored students had nearly the same level of Target Node Strategy usage (0.66, SD = 0.23) as control students (0.70, SD = 0.19), and the difference is not reliable ($p = 0.59$, $d = -0.19$).

6. Discussion

Our results are summarized in Table 1. This section discusses our interpretation of them.

As mentioned in the introduction, whenever a new kind of instruction is developed, there are four classic questions to ask about learning strategies that are specific to it:

Table 1
Summary of the results.

Measure (predicted dir.)	Experiment 3 ($N = 23$)	Experiment 4 ($N = 44$)	Experiment 5 ($N = 33$)
Transfer phase (Hypothesis 1)			
Run model button usage (E < C)	Not available	E < C ($p = 0.31$, $d = 0.32$)	E ≈ C ($p = 0.98$, $d = -0.0093$)
Extra nodes (E < C)	E < C ($p = 0.61$, $d = 0.58$)	**E < C ($p = 0.02$, $d = 0.80$)**	E < C ($p = 0.47$, $d = 0.26$)
Probs completed (E > C)	E > C ($p = 0.06$, $d = 0.88$)	E ≈ C ($p = 0.65$, $d = 0.04$)	E < C ($p = 0.09$, $d = -0.57$)
Training phase (Hypothesis 2)			
Help button usage (E < C)	E < C ($p = 0.08$, $d = 0.82$)	**E < C ($p = 0.04$, $d = 0.68$)**	**E < C ($p = 0.02$, $d = 0.89$)**
Correct on 1st Chk (E > C)	Missing data	Missing data	**E > C ($p = 0.015$, $d = 0.98$)**
Efficiency (E > C)	E > C ($p = 0.30$, $d = 0.23$)	E < C ($p = 0.05$, $d = -0.70$)	E > C ($p = 0.59$, $d = 0.19$)
Transfer phase use of Target Node Strategy (Hypothesis 3)			
Usage (E = C)	E ≈ C ($p = 0.70$, $d = 0.16$)	Missing data	E ≈ C ($p = 0.59$, $d = -0.19$).

E stands for the meta-tutor group, and C stands for the control group. Reliable results are bold.

A. Without instruction in good learning strategies, do students often exhibit poor learning strategies?
B. Can good learning strategies be easily taught?
C. When students use the taught learning strategies, does their domain learning increase compared to students who are not taught to use the learning strategies?
D. When instruction in the learning strategy ceases, do students revert to poor learning strategies?

After we developed a step-based tutoring system for model construction, experiments 1 and 2 answered question A by finding that student did indeed exhibit poor learning strategies when using it. This led us to develop a meta-tutor that taught students a meta-strategy that we hoped would increase their acquisition of skill in model construction.

Answering question B requires that even during the training phase, students have the freedom to choose between following or not following the learning strategy. This freedom allows experimenters to measure the growth in compliance during the training phase. Of the four systems reviewed earlier (Betty's Brain, the Help Tutor, Co-Lab and Pyrenees; see Table 2), Betty's Brain and the Help Tutor provide such freedom, and thus were able to show that while students were being meta-tutored, their behaviours were more often consistent with the learning strategy than the behaviours of students who were not being meta-tutored. This suggests that their meta-tutoring was effective at getting students to use the taught learning strategy. AMT, Co-Lab and Pyrenees all *required* students to follow the taught learning strategy during the training phase, so they could not answer question B.

Given a learning strategy that designers *think* is good, question C asks whether it really does increase domain learning. Of the four systems reviewed earlier, only Pyrenees' learning strategy increased domain learning. Thus, we adapted its learning strategy for use in AMT. Because the goal of our system is teach students to construct models properly, most of the measures addressed the frequency of deep modelling. During the training phase, on the measures "help button usage" and "correct on first check" meta-tutored students scored reliably higher than students who were not meta-tutored, except on experiment 3, which appears to have been underpowered. Moreover, the effect sizes were large ($d = 0.89$; $d = 0.98$) or moderately larger ($d = 0.68$). Thus, the AMT results for domain learning in the training phase were nearly as good as those from Pyrenees, and substantially better than the other three meta-tutoring systems.

Unfortunately, neither the Target Node Strategy nor the domain learning advantages transferred. During the transfer phase, on the measures "Run model button usage", "extra nodes" and "Target Node Strategy usage", the meta-tutored students were no better than the students who had not been meta-tutored earlier, with one exception. On the measure "extra nodes" in experiment 4, meta-tutored students outscored the control students. It is always difficult to get learning to transfer from a supported context to an unsupported one, so this lack of transfer should perhaps not be surprising. However, it does appear to be a bit weaker than the transfer obtained by Pyrenees, Betty's Brain and the Help Tutor.

In both the training and transfer phases, we attempted to measure deep modelling using efficiency: the amount of modelling done in a fixed period of time. This measurement made the tacit assumption that deep modelling is faster than shallow modelling. That is, thinking hard to solve an atomic modelling problem should take less time than guessing, overusing the Give-up button, overusing the Run button, scanning the problem statement for keywords and other shallow model construction tactics. However, the efficiency measures all produced null results. If anything, there was trend for non-meta-tutored students to get more done than the meta-tutored students. This is consistent with our informal analyses of the log data. It appears that guessing was actually quite a bit faster than thinking, especially in the training phase when the Check button was enabled. Even when students could only use the Run Model button for feedback in the transfer phase, guessing was fast because the models were so small for the first few problems. After that, guessing became must less efficient, but few students got that far.

As the experience with Betty's Brain and the Help Tutor shows, not every supposedly good learning strategy actually turns out to increase domain learning. In the case of AMT, we have found a learning strategy that does increase domain learning, and thus deserves to be taught.

On the other hand, our method of teaching the learning strategy appears not to have had enough meta-cognitive or motivational impact on students, because their gains while being meta-tutor did not persist when the meta-tutoring was turned off.

Thus, the next stage of the AMT project is to augment the instruction with an affective learning companion. Its job will be to persuade students of the benefits of using the Target Node Strategy and of not abusing the Check and Give-up buttons. The agent cannot use the argument that the learning strategy and deep modelling will speed up the students' work, because we have found that shallow modelling strategies are actually faster at least on these simple problems. Thus, we plan on having the agent use Dweck's well-known argument (Dweck & Leggett, 1988) that the "mind is a muscle; the harder you exercise it, the stronger it becomes." Our summer school students presumably want stronger minds, so if they believe the agent, they should more often use both the learning strategy and deep modelling. In order to make the agent easier to believe, we plan to use attribution shifting, empathy, rapport-building chit-chat, and other non-cognitive

Table 2
Comparison of four meta-tutors of model construction.

Meta-tutor	Training phase		Transfer phase	
	Knowledge/skill	Meta-strategy	Knowledge/skill	Meta-strategy
AMT	+	Required	+	NS
Pyrenees	++	Required	++	NS
Help Tutor	NS	+	NS	+
Betty's Brain	NS	+	+	+
Co-Lab	NS	Required		

NS = non-significant difference, two-tailed. + = Significant but weak. ++ = Significant and strong. Required = meta-tutor required student to follow the learning strategy.

techniques. In order to optimize the timing and selection of these non-cognitive interventions, we plan to monitor the students' affective state using physiological sensors.

Lastly, by experiment 5, students seem to be acquiring decent amounts of competence in system dynamics modelling in only 1 h and 15 min total. This is a considerable reduction from the 5 or more hours required of Model-It students and others. Although we have no way to actually compare our results to the early ones, because the systems, students, domains and almost everything else were quite different, we nonetheless are quite encouraged. The combination of tutoring and meta-tutoring may be the key to getting model construction out into the classrooms at last.

Acknowledgements

This material is based upon work supported by the National Science Foundation, United States (_100000001) under Grant No. 0910221.

References

Alessi, S. M.. (December 2000). The application of system dynamics modeling in elementary and secondary school curricula. In *Paper presented at the RIBIE 2000 – The fifth Iberoamerican conference on informatics in education*. Viña del Mar, Chile.
Aleven, V., McLaren, B., Roll, I., & Koedinger, K. (2004). Toward tutoring help seeking (applying cognitive modeling to help-seeking skills). In J. C. Lester, R. M. Vicari, & F. Paraguacu (Eds.), *Intelligent tutoring systems: Seventh international conference: ITS 2005* (pp. 227–239). Berlin: Springer.
Aleven, V., Stahl, E., Schworm, S., Fischer, F., & Wallace, R. M. (2003). Help seeking and help design in interactive learning environments. *Review of Educational Research, 73*(2), 277–320.
Baker, R. S. J. d., Corbett, A., & Koedinger, K. R. (2004). Detecting student misuse of intelligent tutoring systems. In *Proceedings of the 7th international conference on intelligent tutoring systems* (pp. 531–540).
Baker, R. S., Corbett, A., Koedinger, K. R., Evenson, S., Roll, I., Wagner, A. Z., et al. (2006). Adapting to when students game an intelligent tutoring system. In *Intelligent tutoring systems* (pp. 392–401). Berlin: Springer.
Baker, R. S. J. d., Corbett, A., 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.
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*, 363–392.
Booth Sweeney, L., & Sterman, J. D. (2000). Bathtub dynamics: initial results of a systems thinking inventory. *System Dynamics Review, 16*(4), 249–286.
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.
Burleson, W., & Picard, R. W. (2007). Affective learning companions. *Educational Technology, Special Issue on Pedagogical Agents, Saddle Brook, N.J., 47*(1), 28–32.
CCSSO. (2011). The Common Core State Standards for mathematics. Downloaded from www.corestandards.org. on 31.10.11.
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. 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.
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.
Collins, A., & Ferguson, W. (1993). Epistemic forms and epistemic games: structures and strategies to guide inquiry. *Educational Psychologist, 28*(1), 25–42.
Doerr, H. M. (1996). Stella ten-years later: a review of the literature. *International Journal of Computers for Mathematical Learning, 1*, 201–224.
Donker, A. S., de Boer, H., Kostons, D., Dignath van Ewijk, C. C., & van der Werf, M. P. C. (2014). Effectiveness of learning strategy instruction on academic performance: a meta-analysis. *Educational Research Review, 11*, 1–26.
Du Boulay, B., Avramides, K., Luckin, R., Martínez-Mirón, E., & Rebolledo-Méndez, G. (2010). Towards systems that care: a conceptual framework based on motivation, metacognition and affect. *International Journal of Artificial Intelligence in Education, 20*(3), 197–229.
Dweck, C. S., & Leggett, E. L. (1988). A social-cognitive approach to motivation and personality. *Psychological Review, 95*(2), 256–273.
Gonzalez-Sanchez, J., Chavez-Echeagaray, M.-E., VanLehn, K., & Burleson, W. (2011). From behavioral description to a pattern-based model for intelligent tutoring systems. In *Paper presented at the Proceedings of the 18th international conference on pattern languages of programs (PLoP)*. Portland, OR.
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.
Hattie, J., Biggs, J., & Purdie, N. (1996). Effects of learning skills interventions on student learning: a meta-analysis of findings. *Review of Educational Research, 66*, 99–136.
Hestenes, D. (2007). Modeling theory for math and science education. In *Paper presented at the ICTMA-13: The international community of teachers of mathematical modelling and applications*. Indiana, IL.
Hogan, K., & Thomas, D. (2001). Cognitive comparisons of students' systems modeling in ecology. *Journal of Science Education and Technology, 10*(4), 319–345.
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., 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.
Mandinach, E. B., & Cline, H. F. (1994a). *Classroom dynamics: Implementing a technology-based learning environment*. Mahwah, NJ: Erlbaum.
Mandinach, E. B., & Cline, H. F. (1994b). Modeling and simulation in the secondary school curriculum: the impact on teachers. *Interactive Learning Environments, 4*(3), 271–289.
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.
Metcalf, S. J. (1999). *The design of guided learner-adaptable scaffolding in interactive learning environment* (Doctoral Dissertation). University of Michigan.
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).

Mulder, Y. G., Lazonder, A. W., de Jong, T., Anjewierden, A., & Bollen, L. (2011). Validating and optimizing 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.

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

Richmond, B. M. (1985). STELLA: software for bringing system dynamics modeling to the other 98%. In *Paper presented at the Proceedings of the 1985 international conference of the System Dynamics Society: 1985 International system dynamics conference*.

Roll, I., Aleven, V., McLaren, B., & Koedinger, K. R. (2007a). Can help seeking be tutored? Searching for the secret sauce of metacognitive tutoring. In *Proceedings of the international conference on artificial intelligence in education* (pp. 203–210). Amsterdam: IOS Press.

Roll, I., Aleven, V., McLaren, B., & Koedinger, K. R. (2007b). Designing for metacognition – applying cognitive tutor principles to the tutoring of help seeking. *Metacognition and Learning, 2*(2).

Roll, I., Aleven, V., McLaren, B., & Koedinger, K. R. (2011). Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and Instruction*, 267–280.

Roll, I., Aleven, V., McLaren, B., Ryu, E., Baker, R. S. J. d., & Koedinger, K. R. (2006). The Help Tutor: does metacognitive feedback improve student's help-seeking actions, skills and learning. In M. Ikeda, K. Ashley, & T.-W. Chan (Eds.), *Intelligent tutoring systems: 8th International conference, its 2006* (pp. 360–369). Berlin: Springer.

Russell, S., & Norvig, P. (2009). *Artificial intelligence: A modern approach* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

Schecker, H. (1993). Learning physics by making models. *Physics Education, 28*, 102–106.

Segedy, J. R., Kinnebrew, J. S., & Biswas, G. (2012a). Relating student performance to action outcomes and context in a choice-rich learning environment. In S. A. Cerri, W. J. Clancey, & G. Papadourakis (Eds.), *Intelligent tutoring systems: 11th International conference its 2012* (pp. 505–510). Berlin: Springer-Verlag.

Segedy, J. R., Kinnebrew, J. S., & Biswas, G. (2012b). Supporting student learning using conversational agents in a teachable agent environment. In *Paper presented at the Proceedings of the 10th international conference of the learning sciences*. Sydney, Australia.

Shih, B., Koedinger, K. R., & Scheines, R. (2008). A response time model for bottom-out hints as worked examples. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S. J. d Baker (Eds.), *Handbook of educational data mining* (pp. 201–211). Boca Raton, FL: Taylor & Francis.

Steed, M. (1992). Stella, a simulation construction kit: cognitive process and educational implications. *Journal of Computers in Mathematics and Science Teaching, 11*, 39–52.

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.

Tan, J., Biswas, G., & Schwartz, D. L. (2006). Feedback for metacognitive support in learning by teaching environments. In *Proceedings of the twenty-eighth annual meeting of the Cognitive Science Society*. Mahwah, NJ: Erlbaum.

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.

Timms, M. J. (2007). Using item response theory (IRT) to select hints in an ITS. In R. Luckin, K. R. Koedinger, & J. Greer (Eds.), *Artificial intelligence in education* (pp. 213–221). Amsterdam: IOS Press.

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. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence and Education, 16*, 227–265.

VanLehn, K. (2013). Model construction as a learning activity: a design space and review. *Interactive Learning Environments, 21*(4), 371–413.

VanLehn, K., Burleson, W., Chavez-Echeagaray, M.-E., Christopherson, R., Gonzalez-Sanchez, J., Hastings, J., et al. (2011a). The level up procedure: how to measure learning gains without pre- and post-testing. In T. Hirashima (Ed.), *Proceedings of the 19th international conference on computers in education* (pp. 96–100). Chiang-Mai, Thailand: Asia-Pacific Society for Computers in Education.

VanLehn, K., Burleson, W., Chavez-Echeagaray, M.-E., Christopherson, R., Gonzalez-Sanchez, J., Hastings, J., et al. (2011b). The affective meta-tutoring project: how to motivate students to use effective meta-cognitive strategies. In *Paper presented at the 19th International conference on computers in education*. Chiang Mai, Thailand.

Vanlehn, K., & Chi, M. (2012). Adaptive expertise as acceleration of future learning: a case study. In P. J. Durlach, & A. Lesgold (Eds.), *Adaptive technologies for training and education*. Cambridge: Cambridge University Press.

Wagster, J., Tan, J., Biswas, G., & Schwartz, D. L. (2007). How metacognitive feedback affects behavior in learning and transfer. In *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 *Proceedings of the twenty-sixth annual meeting of the Cognitive Science Society*. Mahwah, NJ: Erlbaum.

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.

Wilensky, U. (2003). Statistical mechanics for secondary school: the GasLab multi-agent modeling toolkit. *International Journal of Computers for Mathematical Learning, 8*(1), 1–41.

Wilensky, U., & Reisman, K. (2006). Thinking like a wolf, a sheep, or a firefly: learning biology through constructing and testing computational theories – an embodied modeling approach. *Cognition and Instruction, 24*(2), 171–209.

Zaraza, R., & Fisher, D. (1997). Introducing system dynamics into the traditional secondary curriculum: the CC-STADUS project's search for leverage points. In *Paper presented at the 15th International system dynamics conference*. Istanbul, Turkey.

Zaraza, R., & Fisher, D. (1999). *Training system modelers: the NSF CC-STADUS and CC-SUSTAIN projects*. In W. Feurzeig, & N. Roberts (Eds.), *Modeling and simulation in science and mathematics education* (Vol. 1); (pp. 38–69). New York, NY: Springer.