The Affective Meta-Tutoring Project: Lessons Learned

Kurt VanLehn[1], Winslow Burleson[1], Sylvie Girard[2],
Maria Elena Chavez-Echeagaray[1], Javier Gonzalez-Sanchez[1],
Yoalli Hidalgo-Pontet[1], and Lishan Zhang[1]

[1] Arizona State University, Tempe, AZ, USA
{kurt.vanlehn,winslow.burleson,mchaveze,javiergs,
yoalli.hidalgopontet,Lishan.zhang}@asu.edu
[2] University of Birmingham, Birmingham, UK
s.a.girard@bham.ac.uk

Abstract. The Affective Meta-Tutoring system is comprised of (1) a tutor that teaches system dynamics modeling, (2) a meta-tutor that teaches good strategies for learning how to model from the tutor, and (3) an affective learning companion that encourages students to use the learning strategy that the meta-tutor teaches. The affective learning companion's messages are selected by using physiological sensors and log data to determine the student's affective state. Evaluations compared the learning gains of three conditions: the tutor alone, the tutor plus meta-tutor and the tutor, meta-tutor and affective learning companion.

Keywords: Tutoring, meta-tutoring, learning strategies, affective learning companion, and affective physiological sensors.

1 Introduction

A *learning strategy* is a method used by a student for studying a task domain and doing exercises; a good learning strategy tends to increase the learning of students who follow it, whereas a poor learning strategy tends to decrease learning. A learning strategy is a kind of meta-strategy or meta-cognition. That is, it is knowledge about knowledge acquisition. For example, when studying a worked example, a good learning strategy is to self-explain every line of the example [1]. When working on a tutoring system that gives hints, a good learning strategy is to ask for hints when and only when one is unsure about what to do [2].

A perennial problem is that after students have mastered a learning strategy, they may still choose not to use it [3]. The AMT (Affective Meta-Tutoring) project tested whether an affective learning companion (ALC) could persuade students who were taught a learning strategy to continue using it after instruction in the learning strategy had ceased. The project built a system composed of four modules:

- An *editor*, which was used by students to take the steps needed to solve problems.
- A *tutor*, which taught students a problem-solving skill by giving hints and feedback on each step as the problem is being solved.

S. Trausan-Matu et al. (Eds.): ITS 2014, LNCS 8474, pp. 84–93, 2014.

- A *meta-tutor*, which taught a learning strategy by giving hints and feedback about it as the students' used the tutor.
- An *affective learning companion*, having the goal of persuading students to use the learning strategy even after the *meta-tutor* is turned off.

The evaluation of the system focused on students' learning gains. We hypothesized that when ranked by learning gains, the three conditions we studied would exhibit this pattern:

$$tutor < meta\text{-}tutor + tutor < ALC + meta\text{-}tutor + tutor$$

We also tested whether students instructed with the affective pedagogical agent persisted in using the learning strategy when the meta-tutoring ceased.

This paper summarizes the AMT system and its evaluation, and concludes by discussing similar work. Many details are suppressed in order to keep the paper short, but can be found in the project publication referenced herein.

2 The Task Domain: System Dynamics Modeling

Recent standards for K-12 science and math education have emphasized the importance of teaching students to engage in modeling [4, 5]. Although "modeling" can mean many different things [6], we are interested in teaching students to construct models of systems that change over time (dynamic systems) where the model is expressed in a graphical language that is equivalent to sets of ordinary temporal differential equations. Stella (www.iseesystems.com), Vensim (vensim.com), Powersim (hwww.powersim.com) and similar graphical model editors are now widely used in education as well as industry and science. Much is known about students' difficulties with "systems thinking" and how it improves when students learn how to construct models [6]. The practical importance and strong research base motivated our choice of task domain.

However, even with kid-friendly editors [7], students still require a long time (tens of hours) to acquire even minimal competence in the task. Most science and math classes cannot afford to dedicate this amount of time to learning a modeling tool, so this path to deeper understanding of systems, too often, remains closed. One of the long-term practical goals of this work is to reduce the time-to-mastery from tens of hours to just an hour or two.

3 The AMT System

This section introduces the main parts of the AMT system: the editor, tutor, meta-tutor and ALC.

3.1 The Model Editor

The model editor had two tabs. One presented the problem to be solved, such as:

A bottle initially holds 100 bacteria. They grow quickly. On average, 40% of the bacteria reproduce each hour, each creating one new bacterium. Graph the number of bacteria in the bottle each hour.

The second tab was for constructing the model, which was done by creating nodes (see Figure 1). Each node represented a quantity. A node was defined in 3 steps, each achieved by filling out a form. The first step in defining a node was selecting a quantity from a large menu that included both relevant and irrelevant quantities. The second step (which was actually split into two forms) required qualitative planning and consisted of deciding whether the quantity was a numerical constant (e.g., the bacteria birthrate is 0.4), a quantity that was a function of other quantities (e.g., the number of bacteria births per hour is a function of the bacteria birthrate and the current bacteria population), or a quantity that changed by a specified amount per unit time (e.g., bacteria population increased each hour by the number of bacteria births per hour). The third step was stating a specific formula for calculating the quantity represented by the node.

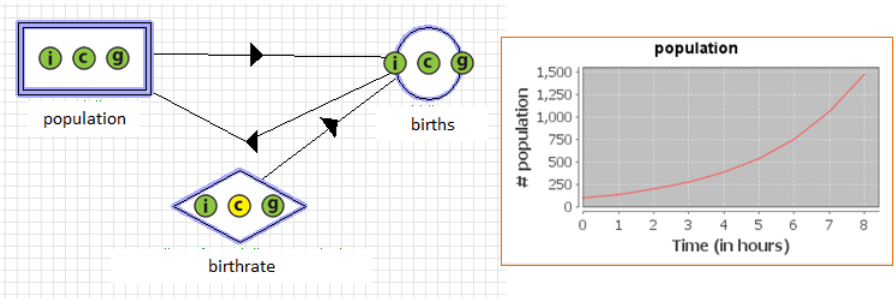

Fig. 1. Model for the bacteria problem, and graph of node "population"

When students had finished constructing a model, they could click on a button to execute it. This added a graph to each node, which students could see by clicking on the node (See Figure 1). Even if the tutoring system was turned off, students were given feedback on the correctness of their graphs. They could see both their graph and the correct graph. The little "g" circle inside the node icons (see Figure 1) was green if the students' graph matched the correct graph and red otherwise. Students could go on to the next problem only when all the nodes' graphs were correct.

3.2 The Tutor

When the tutor was turned on, students could get minimal feedback and bottom out hints. When they were filling out forms, the student could click on a button labelled "Check." This would color the student's entries either green for correct or red for incorrect. The color coding comprised minimal feedback on correctness. Students could also click on a button labelled "Solve it for me." It would finish filling in the form, but would also color the entries yellow. This comprised a bottom-out hint. In order to discourage overuse of the Solve-it-for-me button, when the student finished

editing a node, the status of their work was visible as colors on the little circles inside the node icons ("i" means input; "c" means calculation).

When the tutor was turned off, its feedback and hints were disabled. In particular, the Solve-it-for-me button was always disabled, and the Check button was disabled on all forms except the first one. The Check button was enabled during the first step because the system needed to know which node the student was trying to define so that it could associate a correct graph with the node.

3.3 The Meta-Tutor

When the meta-tutor was turned off, students tended to first define nodes for all the numbers in the problem statement, even if the numbers were irrelevant. Next, they tried to guess definitions of more nodes using keywords such as "initially," "increase" or "altogether." Sometimes they used methodical guessing. Indeed, some students seldom looked at the tab containing the text of the problem. These represent a few of the practices called "shallow modeling" in the literature [6]. The purpose of the meta-tutor is to prevent shallow modelling and encourage deep, thoughtful modeling.

Inspired by the success of the Pyrenees meta-tutor [8], our meta-tutor explicitly taught students a general goal decomposition method. For the students' benefit, we called it the Target Node Strategy and described it as follows:

1. Pick the quantity that the problem asks you to graph, create a node for it, and call it the *target node*.
2. Define the target node *completely*. If the node needs some inputs that haven't been defined yet, create those nodes but don't bother filling them in yet. Return to working on the target node, and don't stop until it's finished.
3. When the target node is finished, if there are nodes that have been created but not defined, then pick *any of them* as the new target node, and go to step 2. If every node has been defined, then the model is complete and you can execute it.

When the meta-tutor was on, it required the student to follow the Target Node Strategy. It also complained if the students overused the Solve-it-for-me or Check buttons, just as other meta-tutors do [9]. The meta-tutor also advised students on how to debug models (e.g., if several nodes have incorrect graphs, examine first those whose input nodes have correct graphs). We use the term "meta-strategy" to refer to this whole collection of strategic advice about how to use the tutor and the editor.

3.4 The Affective Learning Companion (ALC)

The main job of the ALC was to persuade students that the meta-strategy was worth their time and effort, and thus they should use it frequently not only when the meta-tutor was nagging them, but also when the meta-tutor and the ALC were turned off. To achieve this persuasion, we used both affect-based and motivation-based designs for the agent and its behavior. These designs are discussed in the order in which they were encountered by the student.

Following Dweck and others [10], all students began their training by reading a brief text introducing the "mind is a muscle" paradigm: the more you exercise your mind, the more competent you become. The ALC often referred to this concept, whereas the non-ALC interventions never mentioned it again.

After reading the "mind is a muscle" passage, students in the ALC condition first encountered the agent. The agent's appearance and initial behavior were designed to help establish rapport with the student. Following Gulz [11], the agent was a cartoon of a human. Following Arroyo et al. [12], its gender matched the gender of the student. Given the mixed results of D'Mello and Graesser [13], the agent display a fixed neutral expression. Following Gulz et al. [14], the agent introduced him/herself, and engaged the student in light conversation about the student's interests. The agent's dialogue turns were text, and the student's turns were selected from menus.

The student's next activity was to study a series of PowerPoint slides interwoven with simple exercises. These taught the basics of modeling and the user interface. This activity was the same for both the ALC intervention and the non-ALC intervention, and the agent was absent during it.

When the student had finished the introduction and was about to begin problem solving with the tutor, the ALC appeared and expressed enthusiasm about the upcoming challenges. It also reminded the student that the "mind is a muscle."

Once the student began solving a problem, the ALC "spoke" via a pop-up text approximately once a minute. If the student was practicing deep modeling frequently, then the agent remained silent.

When the agent "spoke," its message was selected based on log data and physiological sensor data that were interpreted by machine-learned models. The sensors were a facial expression camera and a posture-sensing chair. The sensor data were cleaned, synched and input to a regression model that predicted the student's emotional state. The emotional state and the output of the log data detectors drove a decision tree that selected one of the following 7 categories, whose message was then presented to the student:

1. *Good Modeling*: Students exhibit frequent deep modeling behaviors and low variation among affective states. ALC: "You really nailed it efficiently! It seems like you are using the strategy and that all your efforts are helping you to make strong connections in your brain. Nice work!"
2. *Engaged*: Students make few errors and show high level of excitement and confidence. ALC: "That's it! By spending time and effort verifying your answers and planning ahead as you use the strategy, your brain is creating more connections that will help you in your future modeling."
3. *New Understanding*: Students show some shallow behaviors without making too many errors, and some may show some frustration. ALC: "You're getting good at this. Planning ahead is the way to go. I can almost see the connections forming in your brain."
4. *Inconsistent*: Students make many shallow behaviors and show high level of frustration. ALC: "Remember to stay focused and use the strategy and your plan. Your actions seem to be inconsistent with the plan you picked earlier. If you

planned on having a <fixed value> node, then why are you trying to create a <function>? It's OK; sometimes it can be confusing; just remember to always try to do your best…"

5. *Guessing*: Students enter several answers before getting one correct, perform many shallow behaviors, and show low level of excitement: "Sometimes one must guess. But even if you've been guessing recently, try to figure out why the response that got green was correct. That way you can get there faster next time without guessing."
6. *Fluttering, Confused, Lost*: Students make many errors. While the student sometimes refers to instructions and the problem, the student only uses these features when stuck, not when planning the modeling activity. ALC: "You seem a little lost. Sometimes these activities can be confusing. Do you think you can go back to the strategy and use it to make a plan about the best way to spend your effort? This will probably help you make progress."
7. *Boredom*: Students make some errors and show consistently low level of interest. ALC: "If this activity seems boring, why not turn it into a game to make it more fun? For instance, do you think you can finish a node while getting green on your first try at every tab?"

The ALC messages quoted above were the ones presented initially. If the same message needed to be presented later, one of 10 short versions was presented instead.

When students finished a problem, a rectangular bar appeared alongside the agent in order to reify the student's meta-cognitive performance, following [15, 16]. The bar was divided into three segments that displayed the proportion of student actions that were deep (green), shallow (red) or neutral (yellow). The modeling depth bar was intended to shift students' motivational focus from correctness (the red/green coding of the tutor) to effort (the red/green coding of the bar). After the ALC explained what the bar meant, it presented a message based on a 6-way categorization that took into account the student's behavior throughout the solving of the problem [17]. The student was then prompted to begin the next problem.

When the training phase was completed, the ALC appeared for the last time and encouraged the student to continue to use deep modeling practice in the forthcoming transfer phase.

The ALC's messages turned out to be mostly motivational and meta-cognitive. The messages were designed "bottom up" by experienced human coders who were familiar with the affect and motivation literature. The messages were tailored to fit the student's state as the coders interpreted it rather than to cleave precisely to one affect/motivation theory or another.

However, the ALC did choose which message to present on the basis of the student's affective state, as detected by the sensors. As advocated by [18], some messages probably work best if they were delivered only in some affective states. For instance, criticizing the students' effort when they are frustrated may cause disengagement, but the same message delivered to a bored student might have a better chance at re-engaging them.

4 Evaluation

This section reports the outcomes (main results) of our experiments evaluating the meta-tutor (studies 3, 4 and 5) and the ALC (studies 6 and 7). Studies 1 and 2 were pilot studies that involved only the editor and the tutor, and will not be discussed here.

4.1 Methods

Procedure: All five experiments used the same procedure. There were two phases: A 75 minute training phase and a 30 minute transfer phase. During the training phase, all students studied PowerPoint slides which introduced them to system dynamics modeling, the model editor and the Target Node Strategy. They also engaged in a series of training problems of increasing complexity. The Check and Solve-it-for-me buttons were available to give them feedback and demonstrations, respectively, on each step in constructing a model. During the transfer phase, the tutor, meta-tutor and ALC were all turned off. Thus, the transfer phase allowed students to display both competence at system dynamics modeling and the Target Node Strategy.

Design: Students were randomly assigned to treatment groups. The treatment manipulation occurred only during the training phase and only while the students were solving problems. There were three treatment conditions: tutor alone; tutor + meta-tutor and tutor + meta-tutor + ALC.

Measures: The studies used basically the same measures, although there were improvements as the studies progressed. There were three types of measures, which were all calculated from log data:

- *Efficiency*: How much modeling were students able to complete in a fixed period of time?
- *Error rate*: How many mistakes did students make when defining a node? How often did they get green (correct) the first time they clicked the Check button?
- *Modeling depth*: Did students use deep or shallow modeling practices?
  - How frequently did students guess or otherwise "game the system?"
  - How frequently were their actions consistent with the Target Node Strategy?
  - How frequently did students refer to the problem statement?
  - How frequently did students refer back to the introductory PowerPoint slides?
  - How many irrelevant nodes did students create?
  - How many episodes were classified as deep by the log data detectors?

Participants: Because we aimed at evaluating affective interventions, we conducted the studies (except 6) in a classroom context, namely ASU summer schools for high school students. ASU summer school classes always had between 40 and 50 students each. Background questionnaires indicated that students varied in their mathematical preparation from Algebra I to Calculus. We attempted to deal with the high incoming variance using co-variants (studies 3, 4 and 5) and stratified sampling (studies 6 and 7).

Nonetheless, the high variance in incoming attributes and the limited number of participants resulted in our studies being underpowered, which partly explains why several tests presented below turned out to be statistically unreliable.

4.2 Results of Comparing Meta-Tutor + Tutor to Tutor Alone

Studies 3, 4 and 5, which are fully described in [19], evaluate the impact of meta-tutoring using two treatment groups. The experimental group had both the meta-tutor and tutor turned on, whereas the control group had the meta-tutor turned off leaving only the tutor active. Our three main hypotheses and their evaluations follow.

During the training phase, meta-tutoring should improve students' efficiency, error rate and depth of modeling. In all three studies, on almost all measures, the results were in the expected direction, but the differences were statistically reliable only about half the time. The results for efficiency were weakest, probably because guessing often took less time than thinking hard. On the whole, we conclude that meta-tutoring probably did improve training phase performance.

During the transfer phase, efficiency and error rate should be better for the meta-tutored group because they should have acquired more skill in modeling during the training phase. Although there were weak trends in the expected direction, only one of the depth measures showed a statistically reliable difference. We conclude that meta-tutoring did not improve transfer phase performance enough to be detectable.

During the transfer phase, the meta-tutored group should not use deep model practices more frequently than the control group because the meta-tutor merely nags; it is the job of the ALC to persuade students to keep using deep modeling practices. This hypothesis predicts a null result, which was observed with all measures in all experiments, but the low power prevents drawing any conclusion from the null results.

4.3 Results of Adding the ALC to the Meta-Tutor + Tutor

Study 6 evaluated a preliminary version of the ALC that only intervened between modeling tasks and was not driven by the physiological sensors. None of the Study 6 measures showed benefits for this preliminary ALC compared to using the system without the ALC. Unlike the other studies, this was a lab study with university students intended mostly to collect data for calibrating the physiological sensors.

Study 7 compared the complete system to the same system with the ALC turned off. Our findings were:

- During the training phase, the ALC group was better than the non-ALC group on all measures, although the differences were reliable on only half the measures.
- During the transfer phase, the two groups tied on all error rate and efficiency measures, suggesting that they both learned the same amount during training.
- Also during the transfer phase, the ALC group was not different from the non-ALC group in this use of deep modeling practices.

Our interpretation of the results of Study 7 is that the ALC probably acted like an improved meta-tutor. That is, during training, it caused students to use deeper

modeling strategies, which increased their efficiency and decreased their error rates, but did not apparently affect their learning very much, because their advantage over the comparison group did not persist into the transfer phase. Although the AMT project has made many contributions, this finding is perhaps the main result of the project.

5 Discussion

While our studies were being conducted, other related studies were being done. There are now 12 studies in the literature besides our own where an ALC acted somewhat like ours [20], and only 4 had reliable main effects. Of them, 3 studies used memorization tasks, and the fourth study confounded instructional information with the affective intervention. On the other hand, all 8 studies with null effects used complex tasks, as did our studies. It is tempting to hypothesize that ALCs work best with simple, short tasks perhaps because there are more frequent opportunities for interacting with the ALC between tasks.

Overall, the good news is that we have discovered improvements to meta-tutoring that increase the frequency of deep modeling practices when the meta-tutoring is operating. This is important because modeling is becoming a more central part of the math and science standards, and students have strong tendencies to use shallow modeling practices. Unfortunately, we have not yet found a way to get this improved performance to persist when the meta-tutoring is turned off.

Another piece of good news is that students were able to achieve adequate competence in constructing system dynamics models with only 75 minutes of training. This is nearly an order of magnitude faster than earlier work with high school students [6].

In one key respect, the ALC's intervention could be improved. Our hypothesis was that using the affect sensors and detectors would allow the ALC's messages to be presented at emotionally optimal times. However, we did not actually vary the time of the messages enough. This would be a good topic for future work.

Acknowledgements. This material is based upon work supported by the National Science Foundation under Grant No. 0910221.

References

1. Fonseca, B., Chi, M.T.H.: The self-explanation effect: A constructive learning activity. In: Mayer, R.E., Alexander, P. (eds.) The Handbook of Research on Learning and Instruction, pp. 296–321. Routledge, New York (2011)
2. Aleven, V., et al.: Help seeking and help design in interactive learning environments. Review of Educational Research 73(2), 277–320 (2003)
3. Hattie, J., Biggs, J., Purdie, N.: Effects of learning skills interventions on student learning: A meta-analysis of findings. Review of Educational Research 66, 99–136 (1996)
4. National, R.C.: A Framework for K-12 Science Education: Practices, Crosscutting concepts, and Core Ideas. National Academies Press, Washington (2012)

5. CCSSO, The Common Core State Standards for Mathematics (2011), http://www.corestandards.org (October 31, 2011)
6. VanLehn, K.: Model construction as a learning activity: A design space and review. Interactive Learning Environments 21(4), 371–413 (2013)
7. Metcalf, S.J., Krajcik, J., Soloway, E.: Model-It: A design retrospective. In: Jacobson, M.J., Kozma, R.B. (eds.) Innovations in Science and Mathematics Education: Advanced Designs for Technologies of Learning, pp. 77–115 (2000)
8. Chi, M., VanLehn, K.: Meta-cognitive strategy instruction in intelligent tutoring systems: How, when and why. Journal of Educational Technology and Society 13(1), 25–39 (2010)
9. Roll, I., et al.: Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. Learning and Instruction, 267–280 (2011)
10. Dweck, C.S., Leggett, E.L.: A social-cognitive approach to motivation and personality. Psychological Review 95(2), 256–273 (1988)
11. Gulz, A.: Benefits of virtual characters in computer-based learning environments: Claims and evidence. International Journal of Artificial Intelligence and Education 14(3), 313–334 (2004)
12. Arroyo, I., et al.: The impact of animated pedagogical agents on girls' and boys' emotions, attitudes, behaviors and learning. In: International Conference on Advanced Learning Technologies (ICALT 2011), Athens, Georgia (2011)
13. D'Mello, S., Lehman, B., Sullins, J., Daigle, R., Combs, R., Vogt, K., Perkins, L., Graesser, A.: A time for emoting: When affect-sensitivity is and isn't effective at promoting deep learning. In: Aleven, V., Kay, J., Mostow, J. (eds.) ITS 2010, Part I. LNCS, vol. 6094, pp. 245–254. Springer, Heidelberg (2010)
14. Gulz, A., Haake, M., Silvervarg, A.: Extending a teachable agent with a social conversation module – Effects on student experiences and learning. In: Biswas, G., Bull, S., Kay, J., Mitrovic, A. (eds.) AIED 2011. LNCS, vol. 6738, pp. 106–114. Springer, Heidelberg (2011)
15. Arroyo, I., et al.: Repairing disengagement with non-invasive interventions. In: Luckin, R., Koedinger, K.R., Greer, J. (eds.) Artificial Intelligence in Education, pp. 195–202. IOS Press, Amsterdam (2007)
16. Walonoski, J.A., Heffernan, N.T.: Prevention of off-task gaming behavior in intelligent tutoring systems. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.) ITS 2006. LNCS, vol. 4053, pp. 722–724. Springer, Heidelberg (2006)
17. Girard, S., Chavez-Echeagaray, M.E., Gonzalez-Sanchez, J., Hidalgo-Pontet, Y., Zhang, L., Burleson, W., VanLehn, K.: Defining the behavior of an affective learning companion in the affective meta-tutor project. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS, vol. 7926, pp. 21–30. Springer, Heidelberg (2013)
18. D'Mello, S.K., Graesser, A.C.: Dynamics of affective states during complex learning. Learning and Instruction 22, 145–157 (2012)
19. Zhang, L., et al.: Evaluation of a meta-tutor for constructing models of dynamic systems. Computers & Education (in press)
20. Girard, S., et al.: How can Affect be used to improve the Learning outcomes of Interactive Instructional Systems? (in prep.)