Learning by Teaching SimStudent: Technical Accomplishments and an Initial Use with Students

Noboru Matsuda[1,*], Victoria Keiser[1], Rohan Raizada[1], Arthur Tu[1], Gabriel Stylianides[2], William W. Cohen[1], and Kenneth R. Koedinger[1]

[1] School of Computer Science, Carnegie Mellon University
5000 Forbes Ave. Pittsburgh PA 15213 USA
[2] School of Education, University of Pittsburgh
5517 Posvar Hall, Pittsburgh PA 15260 USA
{noboru.matsuda,keiser,atu,wcohen,koedinger}@cs.cmu.edu,
rohan@gmai.com, gstylian@pitt.edu

Abstract. The purpose of the current study is to test whether we could create a system where students can learn by teaching a live machine-learning agent, called SimStudent. SimStudent is a computer agent that interactively learns cognitive skills through its own tutored-problem solving experience. We have developed a game-like learning environment where students learn algebra equations by tutoring SimStudent. While Simulated Students, Teachable Agents and Learning Companion systems have been created, our study is unique that it genuinely learns skills from student input. This paper describes the overview of the learning environment and some results from an evaluation study. The study showed that after tutoring SimStudent, the students improved their performance on equation solving. The number of correct answers on the error detection items was also significantly improved. On average students spent 70.0 minutes on tutoring SimStudent and used an average of 15 problems for tutoring.

Keywords: SimStudent, Learning by teaching, tutor-learning effect, algebra equation solving, machine learning.

1 Introduction

There is ample evidence that students learn when they teach their peers [1]. Such an *effect of tutor learning* has been observed across different subjects, age groups, format of tutoring, and so forth. Yet, little is known about when tutors' learning would be facilitated and why. A scientific contribution of the current study is at our exploratory effort to study cognitive and social factors for tutor learning. Even when tutor

* This study is supported by National Science Foundation Award No. DRL-0910176 and by Department of Education (IES) Award No. R305A090519. This work is also supported in part by the Pittsburgh Science of Learning Center, which is funded by the National Science Foundation Award No. SBE-0836012.

V. Aleven, J. Kay, and J. Mostow (Eds.): ITS 2010, Part I, LNCS 6094, pp. 317–326, 2010.

learning is effective, there are practical difficulties to exercise peer tutoring in an actual classroom – not only would it be time consuming (the students must take turns) but, also, the tutees might not learn as much as tutors do. Thus, on the engineering side of our contribution, building an effective and efficient learning environment that facilitates tutor learning is one of our primary research goals.

We have developed a game-like learning environment where students learn by interactively tutoring a computer agent, called *SimStudent*. SimStudent is a machine-learning agent that learns cognitive skills from examples and through its own tutored problem-solving experiences [2]. Our long-term research goal is to investigate the effect of tutor learning with SimStudent as a teachable agent.

The aim of this paper is to provide an overview of our learning by teaching system and discuss results from an evaluation study. The primary research question in the current paper addressed whether or not the students learn by teaching SimStudent at all, and if so, how effective the system is.

2 Learning by Teaching

2.1 Type and Domain of the Proposed Learning by Teaching Environment

The *effect of tutor learning* has been studied in many different domains, across ages, and in various tutoring settings [3]. Various forms of tutoring have been observed including reciprocal teaching [4] and collaborative passage learning [5]. The effect of tutor learning has been observed for all age groups including college [6], high school [7], middle school [8], and elementary school students [9]. The tutor learning effect has been shown to be relatively more effective in math than reading [3, 10].

In the current study, we focus on one-on-one tutoring where a single student acts as a tutor and a computer agent plays the tutee's role. Although the SimStudent technology and the overall framework of the proposed learning environment are domain independent, the current learning system is built for algebraic linear equations – one of the more challenging subjects in mathematics.

2.2 Related Studies

There have been a number of simulated students (also called *teachable agents*) developed so far [11-14]. VanLehn et al. [11] developed one of the earliest simulated students and demonstrated its benefit for *teacher training* in physics. Betty's Brain [15] and its variations are the most recent examples of a teachable agent used to study the tutor learning effect. Betty's Brain learns causal relations from a conceptual map created by student by entering nodes (each representing a concept) and links (each representing a causal relation among the concepts). Students can also quiz Betty's Brain with a problem asking a causal relation (e.g., "If dead organisms increase, what happens to the animals?").

While Simulated Students, Teachable Agents and Learning Companion systems have been created, some were never used with real students and others do not genuinely learn from student input. While the VanLehn's system [11] incorporated machine learning and could be used for theory generation and to analyze instructional materials, it was not designed for use with students. On the other hand, while Betty's Brain has been used extensively by students and subject to numerous evaluations, it

does not have a machine-learning component. Students teach Betty's Brain by editing a concept map, but the system does not learn from the concept map any more than making straightforward inferences from following the links in the map. This paper provides perhaps the first demonstration of a machine learning system being used as a teachable agent by real students and with significant pre-to-post learning outcomes.

3 SimStudent as a Teachable Peer Learner

3.1 Overview of SimStudent

The underlying technology for SimStudent's learning is inductive logic programming in the form of programming by demonstration [16, 17].

There are two learning strategies implemented for SimStudent so far – *learning from examples* and *learning by tutored problem solving* [18]. The learning strategy used for the current research context is learning by tutored problem solving, which requires a *tutor agent* that interactively tutors SimStudent. The tutor agent first poses a problem for SimStudent to solve. To solve the problem, SimStudent attempts to apply production rules that are already learned. When a rule is applied (i.e., a step is performed), the tutor agent provides flagged (binary) feedback that merely shows a correctness of the rule application. When the feedback is negative (*regardless of the accuracy of this tutor feedback*), SimStudent attempts to apply another rule.

When SimStudent cannot perform a step "correctly," then SimStudent asks the tutor agent for a hint about what to do next.[1] The tutor agent then *demonstrates* the step for SimStudent as a hint, which is equivalent to a so-called bottom-out hint.

SimStudent learns skills by generalizing the examples demonstrated. There are two types of examples: a *positive example* is generated either when the tutor agent provides affirmative feedback on a step that SimStudent performed, or when the tutor agent demonstrates a step as a hint. A *negative example* is generated when SimStudent receives negative feedback from the tutor agent on a step that SimStudent performed. As a result of generalization, SimStudent generates a *production rule* that covers all positive examples (i.e., an application of the rule yields the same step mentioned in a positive example) but does not cover any of the negative examples. In other words, SimStudent generates a set of production rules sufficient to solve problems that share the underlying domain principles that have been demonstrated. SimStudent is given a set of *background knowledge* that allows SimStudent to interpret the examples. See [18] for details of the SimStudent learning algorithm.

Although, the SimStudent's learning algorithm is domain independent, we use algebra equation solving for the current study. In a particular tutoring interface used in the current study (as shown in **Fig. 1**), a single *equation-solving step* is implemented with three *tutoring steps* that are modeled with two skills. For example, $3x+2=8$ is transformed into $3x=6$ by subtracting 2 from both sides, which, by definition, is a single equation-solving step. In our tutoring interface, this equation-solving step consists of (a) specifying a *transformation skill*, which in this case is "subtract 2," and (b) *typing in* a left- and a right-hand side of the transformed equation. The tutoring

[1] The correctness of the step, by definition, is determined by the feedback from the tutor agent. Thus, SimStudent could fail to perform a step "correctly" when the tutor agent disconfirms the step regardless of the true correctness of the rule application.

interface thus has three columns (two under "Equation" and one under "Transformation"), corresponding to these three steps.

The above-mentioned examples are accumulated for each of the skills demonstrated. In other words, when a step is demonstrated, it must be annotated with a *skill-name*. Such annotation drastically reduces the complexity of the search space when learning rules. There is another piece of information that improves the search – the *focus of attention* (FoA, for short), which is knowledge about where to pay attention when performing a step. When tutoring SimStudent, an FoA specifies the elements on the tutoring interface. In the tutoring interface shown in **Fig. 1**, FoA is a set of cells representing either the left- or right-hand side of an equation, or a transformation. Technically, the FoA composes the left-hand side of a production rule encoding a pattern matching for a rule application. Without FoA, SimStudent must search for such a pattern matching, which significantly increases the search complexity.

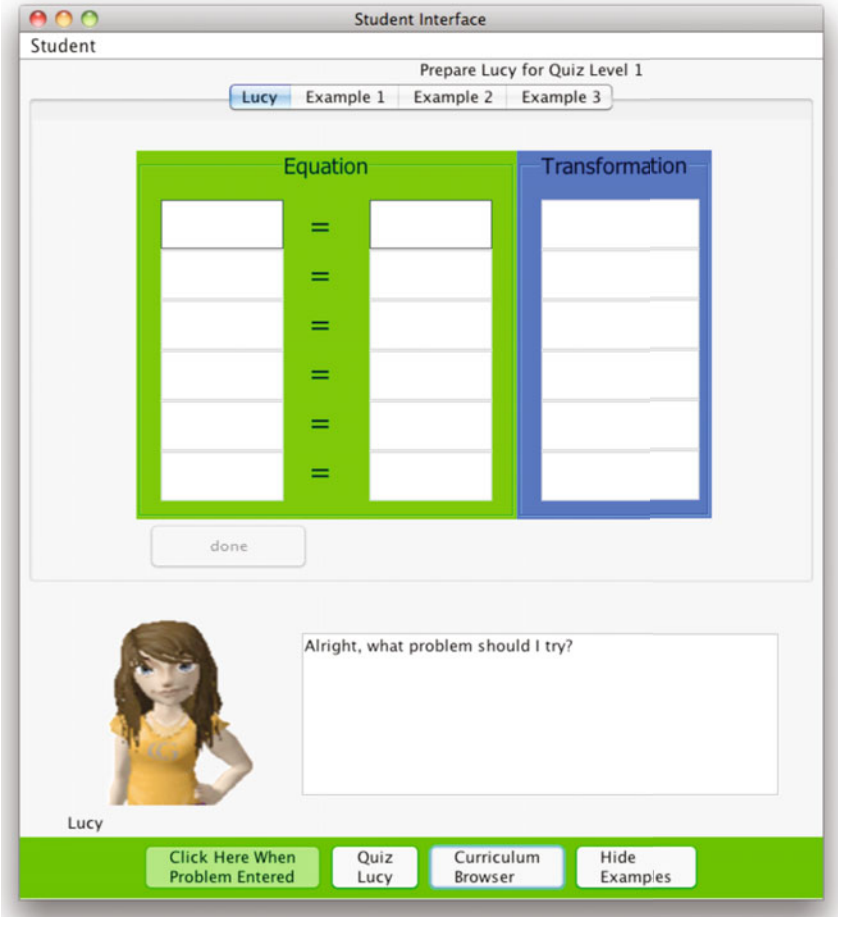

Fig. 1. Learning by Teaching environment. Worked-out examples appear in the interface by clicking the [Example 1,2, and 3] tabs at the top of the interface. SimStudent is called Lucy in this example.

An earlier pilot study showed that asking students to annotate a skill name each time they demonstrate a step is unnatural and confusing. Likewise, asking the students to specify FoA is not practical. Thus, in the current learning by teaching environment, both skill names and FoA are implicitly provided to SimStudent using domain dependent ad-hoc heuristics. When a transformation skill is performed, a skill-name is given as the same name as the transformation (e.g., "subtract"). When a type-in skill is performed following a particular transformation, a skill-name is given as a combination of the transformation and a fixed post-fix (e.g., "subtract-typein"). An FoA for a transformation skill is always the left- and right-hand sides of the equation, upon which the transformation was applied. An FoA for a type-in skill is the immediate transformation skill and the equation on which the transformation skill was applied.

3.2 Learning-by-Teaching Environment

Fig. 1 shows a screenshot of our Learning-by-Teaching learning environment (the *LBT environment*, for short). As shown in the figure, the LBT environment is composed of (1) a tutoring interface, (2) a Curriculum Browser, (3) example problems, (4) a communication window, (5) a SimStudent avatar, called Lucy in this example, and (6) the buttons to control the flow of tutoring. SimStudent and the student share the tutoring interface.

In the LBT environment, a student performs the role of the tutor agent mentioned in section 3.1. The goal of the LBT environment is to tutor SimStudent well enough to pass the quiz, which provides a bit of the flavor of an on-line game to the LBT environment.

To pose a problem for SimStudent to solve, the student simply enters an equation into the tutoring interface, and clicks the [Solve the Equation] button. The *communication window*, the box next to the SimStudent avatar, is used to display a message from SimStudent (e.g., a request for feedback on a step performed or a request for a hint). The student provides feedback by clicking on a [Yes]/[No] button that also appears in the communication window. To provide a "hint," the student simply performs a step in the tutoring interface.

One obvious question is "What if the student cannot provide a hint?" We have two supporting materials for the student to prepare for tutoring. The *Curriculum Browser* shows an overview of the curriculum to learn ("equations with variables on both sides," in the current study). It has descriptions on the goal of the subject, skills to be learned, and an example that explains how to solve a typical problem. The Curriculum Browser appears when the student clicks on the [Curriculum Browser] button. A set of examples is also available when the [Example] button is clicked. Unlike the example in the Curriculum Browser, these examples are shown inside the *tutoring interface* with the exact tutoring steps. Thus, the student can learn how to use the tutoring interface as well.

4 Evaluation Study

To evaluate the LBT environment, we have conducted a lab study. This section describes the details of the study and the results.

The goal of the study is to measure the degree of tutor learning (the effectiveness) and the efficiency and usability of the system. The effectiveness of the study was measured as learning gain using pre- and post-tests. For the efficiency and usability, we conducted a protocol analysis by video-recording the full learning sessions.

4.1 Method

The study involved 12 students ranging from 6th to 8th grade. The participants were recruited from local middle schools for monetary compensation.

All 12 participants followed the same procedure. The total of 12 study sessions were run individually. Before a study session began, an experimenter explained to a participant that for the whole study session was to think aloud. During the study session, the experimenter gave an occasional prompt to the participant to think aloud.

The participant first took an on-line pre-test (the details of the test are described in the next section). The pre-test took 46 minutes on average. The participant then watched a 10-minute video to learn how to use the LBT environment. Next, the participant was told the goal of the study session – to have Lucy (the name of SimStudent in this particular study as shown in **Fig. 1**) pass the quiz. The participant then tutored Lucy. After an hour, the experimenter told the participants that they were welcome to quit the session if they wanted even when Lucy had not passed the quiz.

Finally, the participants took a post-test. The pre- and post-tests were isomorphic, and the difference in the tests was counter-balanced across the participants. After the post-test, all of the participants completed a post-study questionnaire (due to space limitations, we will not discuss the results of the questionnaire in this paper).

All of the study sessions, including the pre- and post-tests as well as the tutoring sessions, were video recorded. The participants' activities during the tutoring session were logged into an open data repository, called DataShop, maintained by the Pittsburgh Science of Learning Center [19].

4.2 Study Materials

The pre- and post-tests were implemented as on-line tests authored using the Cognitive Tutor Authoring Tools. When taking a test, the participants were given a piece of paper to write down their work.

The on-line test consists of question items for (1) equation solving, (2) term identification, (3) what to do next, (4) equivalent expressions, and (5) error identification. The *equation solving* items are to solve equations (e.g., $-3y+6 = 8+5y$) on paper and then only enter the final answer (e.g., $y = -1/4$) into the on-line test form. The *term identification* questions are to identify variable and constant terms in a given expression (e.g., $3 = 4 - 5b$). There are six term-identification questions, which are implemented as multiple-choice questions with six or seven choices. The *what-to-do-next* questions are to identify an appropriate next step for a given equation. There are three what-to-do-next questions, are implemented as multiple-choice questions with four choices. The *equivalent expression* questions are to find an expression that is equivalent to a given expression (e.g., $4x+3$). There are two equivalent-expression questions, which are implemented as multiple-choice questions with five choices. The *error identification* questions are to identify the incorrect step in a given worked-out

example of equation solving. The participants are also asked to provide a reason why the step is incorrect in a free text response. An error-identification question is scored as correct when the incorrect step is correctly selected and the corresponding reason is correctly provided. There are five error-identification questions with four to six steps.

4.3 Results

This section shows the participants' learning outcomes, the analysis of the tutoring activities, and a qualitative analysis of the participants' think aloud protocols, which were recorded during the tutoring sessions.

4.3.1 Overall Learning Gain

We first compared the overall test scores, which were computed as a percent of correct answers to the total number of question items on the test. There was no main effect on the test (pre- vs. post-) for the overall test scores – average 0.58 on pre-test and 0.61 on post-test (paired-t=2.36, p=0.29). There was, however, a main effect found on the test for the equation solving (the average number of equations solved correctly increased from 1.5 out of 6 for pre-test to 2.5 for post-test; paired-t=2.36, p=0.03) and for the error identification test items (average of 1.38 out of 5 correct on pre-test and 2.63 correct on post-test; paired-t=2.36, p=0.01).

After tutoring SimStudent, the participants had improved their skills with solving equations and identifying errors in given solutions, although their conceptual understanding related to equation solving (i.e., the scores on the term identification and equivalent expressions test items) did not improve significantly.

4.3.2 Students Activities

In addition to the test scores, we have analyzed the activity logs recorded during the sessions. At the beginning of the experiment, there was a technical issue in logging and we excluded the first five participants from the activity analysis.

On average, the participants spent 70.0 minutes tutoring SimStudent. On average, the participants posed 15 problems to SimStudent. Five out of six participants reviewed examples an average of 7.8 times (counted as the number of times they switched the examples in the tutoring interface). Given that most of the participants did not actually know how to solve equations, these results show effectiveness of the LBT environment.

SimStudent requested hints 31.8 times on average during a single tutoring session. SimStudent applied rules and asked for their correctness 79.5 times on average. Finally, the participants quizzed SimStudent 4.3 times on average. In average there were 1.6 interactions per minutes during the tutoring sessions, which indicates that the participants were actively involved in the LBT environment.

4.3.3 Protocol Analysis

To see differences in tutoring activities among the students who improved their performance in equation solving and those who did not, we performed a protocol analysis with the videos taken during the tutoring sessions. We divided the students into three categories: those who showed significant improvement on the *equation solving* test-items (SI in **Table 1**), those who showed moderate improvement (not included in

Table 1), and those who showed minimal improvement (MI). The number of the equation solving test-items correctly solved by SI and MI students are shown in **Table 1**. Recall that there were six equation-solving items on each of the tests.

Table 1. Number of correctly solved equation-solving test items. There were six equation-solving questions.

	Pre-test	Post-test	Category
Student A	0	3	SI
Student B	1	3	SI
Student C	0	0	MI
Student D	1	1	MI

It turned out that the SI students used equations in the examples and quiz items for tutoring more often than the MI students. *The SI students often copied those equations to tutor SimStudent, hence more likely tutored SimStudent correctly*. On the other hand, the MI students tended to make equations by themselves and failed to solve them correctly. One particularly interesting finding is that *being able to pose equations for tutoring is one task, but being able to solve them is another*. When making a new problem, one of the MI students always started with a number in his mind and some arithmetic to make another number and then performed those operations to construct an equation. For example, he said, "*I'll start with 4, that is an x. If I multiply it with 3, I get 12. I'll add 5, which is 17. So, 3x+5 = 17. This is the equation*." However, he could not solve equations by reverse engineering them when he did not know the "starting" place.

Another interesting comparison is that *the SI students went back and forth between examples and the Curriculum Browser contents when they got stuck more often than the MI students did*. On the other hand, the MI students did not seem to learn much from the examples and the contents of the Curriculum Browser. The SI students also tended to pose the *exact same problems* used in the Quiz and examples to tutor SimStudent, so that students could at least show SimStudent what to do next without getting stuck.

5 Discussion

5.1 Learning by Teaching SimStudent

Overall, the students in the evaluation study did improve their skills in equation solving after tutoring SimStudent for about 70 min in average. A purpose of the evaluation study was to see if the students learn by teaching SimStudent at all, and so we did not compare learning by teaching with other existing interventions. We plan to conduct a further study to compare learning by teaching with learning by being tutored, in which an existing Algebra I Cognitive Tutor will be used for the control condition.

5.2 Low Proficiency Students Do Not Learn from Examples

The protocol analysis often showed that the participants with low proficiency on equation solving often got stuck when providing a hint in response to SimStudent's request for what to do next. It is particularly interesting to see that participants could pose a problem for tutoring, but could not solve it (as mentioned in 4.3.3).

The current LBT environment is designed to encourage students to ask a classroom teacher or consult a textbook when they get stuck. The learning environment also has the Curriculum Browser and example problems. However, it turned out that these supplemental materials are not enough for some of the low proficiency participants. In particular, these participants apparently needed more direct instruction (as opposed to learning from worked-out examples) on how to solve equations.

5.3 How Would SimStudent's Prior Knowledge Affect Tutor Learning?

The current evaluation study used a version of SimStudent that starts with a "tabula rasa." Namely, it does not know anything about equation solving. Therefore, the students must start with teaching very basic equations (i.e., one-step equations), which forces more time for tutoring. It would be worth investigating how the difference in SimStudent's prior knowledge affects the tutor's learning outcome. A related issue is has to do with the *quality* of SimStudent's prior knowledge. A prior study shows that differences in the quality of the prior knowledge affects SimStudent's learning in both speed and the types of errors made on the test [20], which would also affect the tutor learning.

6 Conclusion

In this paper, we presented an on-line, game-like learning environment where students learn algebra equations by teaching a computer agent, called SimStudent. A pilot study showed that the students did improve their performance on equation solving by tutoring SimStudent.

The study did not confirm that students improved their conceptual understanding by teaching. It is an important open question how we could help students learn such conceptual knowledge by teaching. We plan to conduct a series of studies in actual classroom settings, including a self-explanation study where SimStudent will ask the student to provide justifications for the student's tutoring activities. Such self-explanation might facilitate conceptual understanding during teaching.

In the current study, we only focus of equation solving. Both the architecture of the LBT environment and the learning algorithm of SimStudent are domain generic. Thus, we can apply the proposed framework to other domains to study a generality of the tutor learning effect.

References

1. Roscoe, R.D., Chi, M.: Tutor learning: the role of explaining and responding to questions. Instructional Science 36(4), 321–350 (2008)
2. Matsuda, N., Cohen, W.W., Koedinger, K.R.: Applying Programming by Demonstration in an Intelligent Authoring Tool for Cognitive Tutors. In: AAAI Workshop on Human Comprehensible Machine Learning (Technical Report WS-05-04), pp. 1–8. AAAU Association, Menlo Park (2005)
3. Cohen, P.A., Kulik, J.A., Kulik, C.I.C.: Education outcomes of tutoring: A meta-analysis of findings. American Educational Research Journal 19(2), 237–248 (1982)

4. Palincsar, A.S., Brown, A.L.: Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. Cognition and Instruction 1(2), 117–175 (1984)
5. Bargh, J.A., Schul, Y.: On the cognitive benefits of teaching. Journal of Educational Psychology 72(5), 593–604 (1980)
6. Annis, L.F.: The processes and effects of peer tutoring. Human Learning 2(1), 39–47 (1983)
7. Morgan, R.F., Toy, T.B.: Learning by teaching: A student-to-student compensatory tutoring program in a rural school system and its relevance to the educational cooperative. Psychological Record 20(2), 159–169 (1970)
8. Jacobson, J., et al.: Cross-Age Tutoring: A Literacy Improvement Approach for Struggling Adolescent Readers. Journal of Adolescent & Adult Literacy 44(6), 528–536 (2001)
9. Fuchs, L.S., et al.: Enhancing Students' Helping Behavior during Peer-Mediated Instruction with Conceptual Mathematical Explanations. Elementary School Journal 97(3), 223–249 (1997)
10. Cook, S.B., et al.: Handicapped Students as Tutors. Journal of Special Education 19(4), 483–492 (1986)
11. VanLehn, K., Ohlsson, S., Nason, R.: Applications of simulated students: An exploration. Journal of Artificial Intelligence in Education 5(2), 135–175 (1994)
12. Chan, T.-W., Chou, C.-Y.: Exploring the Design of Computer Supports for Reciprocal Tutoring. International Journal of Artificial Intelligence in Education 8, 1–29 (1997)
13. Hietala, P., Niemirepo, T.: The competence of learning companion agents. International Journal of Artificial Intelligence in Education 9, 178–192 (1998)
14. Biswas, G., et al.: Learning by teaching: a new agent paradigm for educational software. Journal Applied Artificial Intelligence 19(3&4), 363–392 (2005)
15. Biswas, G., et al.: Incorporating self regulated learning techniques into learning by teaching environments. In: Proceedings of the 20th Annual Meeting of the Cognitive Science Society, Chicago, pp. 120–125 (2001)
16. Muggleton, S., de Raedt, L.: Inductive Logic Programming: Theory and methods. Journal of Logic Programming 19-20(Suppl. 1), 629–679 (1994)
17. Cypher, A. (ed.): Watch What I Do: Programming by Demonstration. MIT Press, Cambridge (1993)
18. Matsuda, N., et al.: Why Tutored Problem Solving may be better than Example Study: Theoretical Implications from a Simulated-Student Study. In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoie, S. (eds.) ITS 2008. LNCS, vol. 5091, pp. 111–121. Springer, Heidelberg (2008)
19. Koedinger, K.R., et al.: An open repository and analysis tools for fine-grained, longitudinal learner data. In: Proceedings of the International Conference on Educational Data Mining (2008)
20. Matsuda, N., et al.: A Computational Model of How Learner Errors Arise from Weak Prior Knowledge. In: Taatgen, N., van Rijn, H. (eds.) Proceedings of the Annual Conference of the Cognitive Science Society, pp. 1288–1293. Cognitive Science Society, Austin (2009)