The Invention Lab: Using a Hybrid of Model Tracing and Constraint-Based Modeling to Offer Intelligent Support in Inquiry Environments

Ido Roll[1], Vincent Aleven[2], and Kenneth R. Koedinger[2]

[1] Carl Wieman Science Education Initiative, University of British Columbia, Vancouver, BC
[2] Human-Computer interaction Insititute, Carnegie Mellon University, Pittsburgh, PA
`ido@phas.ubc.ca, aleven@cs.cmu.edu, koedinger@cmu.edu`

Abstract. Exploratory Learning Environments (ELE) facilitate scientific inquiry tasks in which learners attempt to develop or uncover underlying scientific or mathematical models. Unlike step-based Intelligent Tutoring Systems (ITS), and due to task characteristics and pedagogical philosophy, ELE offer little support at the domain level. Lacking adequate support, ELE often fail to deliver on their promise. We describe the Invention Lab, a system that combines the benefits of ELE and ITS by offering adaptive support in a relatively unconstrained environment. The Invention Lab combines modeling techniques to assess students' knowledge at the domain and inquiry levels. The system uses this information to design new tasks in real time, thus adapting to students' needs while maintaining critical features of the inquiry process. Data from an in-class evaluation study illustrates how the Invention Lab helps students develop sophisticated mathematical models and improve their scientific inquiry behavior. Implications for intelligent support in ELE are discussed.

Keywords: intelligent tutoring systems; exploratory learning environments; invention as preparation for learning; model tracing; constraint-based modeling.

1 Introduction

Exploratory Learning Environments (ELE) facilitate inquiry tasks in which students are instructed to develop or uncover an underlying scientific or mathematical model [1]. Adhering to constructivist instructional principles [2], ELE give the learners more responsibility over controlling the learning process, compared with step-based problem-solving environments [3]. For example, students in ELE are expected to analyze data, raise hypotheses, monitor their progress, and in general, behave the way scientists do [1,2]. This is hypothesized to enhance transfer [4], facilitate acquisition of meta-cognitive and self-regulation skills [5], and increase motivation [6]. However, classroom evaluations have repeatedly demonstrated that students often exhibit unproductive inquiry behaviors, subsequently failing to acquire the desired learning goals [1]. These disappointing outcomes have led to an increased interest in supporting students while working with ELE [6,7].

In order to support students at the domain-independent inquiry level, many ELE scaffold the inquiry process using *cognitive tools* [8]. For example, *Rashi*, *Smithtown*,

and *SimQuest* include inquiry notebooks with templates in which students raise hypotheses, document observations, make conjectures, etc. [5,6,9]. Using cognitive tools to scaffold the inquiry process decreases the rate of unproductive behaviors and makes the inquiry process visible, thus helping students internalize the desired inquiry skills [10]. Cognitive tools can also be used to label students' inputs and linearize the inquiry process, making it easier for the ELE to trace students' progress in the task. Consequently, a number of ELE give students feedback on their domain-independent inquiry behavior. For example, Rashi gives feedback to students who make circular arguments [9], the *Science Learning Spaces* gives feedback on experimental designs that do not use the control of variables strategy [11], and *ACE* prompts students who have not explored the interaction space sufficiently [12].

While domain-independent support of the general inquiry cycle is important, evidence suggests that students are also in need for support at the domain level [7]. To do that, ELE should evaluate the content of students' actions. Many Intelligent Tutoring Systems (ITS) evaluate students' responses by tracing their actions using a comprehensive set of rules that outlines common correct or buggy solution paths (termed *model tracing*, [3,13]). However, applying a similar mechanism to ELE faces a twofold challenge. First, ELE should evaluate answers that vary a lot in content and complexity, compared with most step-based ITS. For example, Figure 1 shows an inquiry task in which students are asked to invent a method for calculating variability. Every algebraic procedure is a potential response to this task, and thus should be evaluated by the system. For instance, one common error that students often make is to use the range function as a measure of variability (using "range" implies that variability is determined only by the extreme values). However, students may use different morphs of range, such as "range+1" or "2*range" (all example methods in this paper are taken from students' inventions during the classroom studies). While simplifying students' methods algebraically may simplify the modeling task, it often fails to capture students' misconceptions. For example, several students added up the distances between all subsequent pairs of numbers, which is mathematically equivalent to range: $(a_1-a_2)+(a_2-a_3)+\ldots+(a_{n-1}-a_n)=a_1-a_n$. However, this method reveals a different conceptual error compared with range, since the more complex (yet mathematically equivalent) method uses all data points (and not merely the extreme values) to determine range.
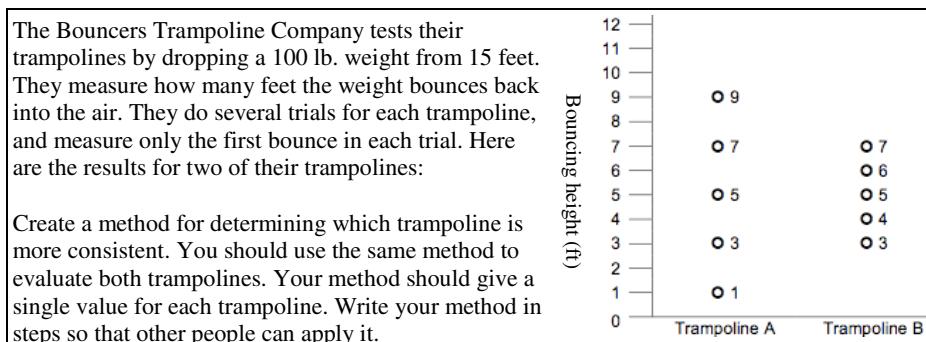

Fig. 1. An example of an invention task. Data is presented in the form of contrasting cases that direct students' attention to deep features of the domain.

In addition to the virtually intractable interaction space, ELE should also deal with an under-defined solution space. This means that not all classes of solutions (whether correct or not) can be defined in advance. While most step-based ITS assume that a solution that is not part of the cognitive model is incorrect [13], this assumption cannot be made with inquiry tasks, in which students may develop methods that do not correspond to classes of solutions identified in advance by the expert modeler.

Due to these challenges, most ELE do not assess students' knowledge at the domain level. Those who do often limit the vocabulary students can use and map the entire solution space. For example, SimQuest evaluates the complete subset of potential experiments that can support or refute each stated hypothesis [6]. Similarly, *Eco-Lab* and Smithtown map all possible nodes in the interaction space [5,14]. Naturally, this approach is not scalable for ELE that facilitate complex tasks or that allow for a large variety of inputs.

In addition to the challenge of analyzing students' errors, ELE designers face the challenge of responding to these errors, that is, designing effective domain-level support that does not undermine the exploratory nature of the inquiry task [6]. While ITS often set sub-goals for students and give them immediate feedback on errors [3,13], the pedagogical philosophy behind ELE suggests that students, and not external agents, should have responsibility over these tasks [2]. Therefore, many ELE offer no support at the domain level [7]. Other ELE give students immediate feedback, thus potentially hindering the benefits of inquiry learning [5]. A better solution would be to support students by adapting the task to their demonstrated proficiencies. For example, EcoLab directs students to one of three canned sets of directions and hints [14]. While this approach adheres to the pedagogical principles of ELE, having canned versions of the tasks is not a scalable solution. Adapting the task to a wide range of knowledge deficits, as students demonstrate in a wide range of situations, remains to be solved.

This paper addresses the two research questions outlined above. First, we describe a novel approach for evaluating students' knowledge at the domain level in ELE. We demonstrate this approach using the Invention Lab, an ELE for invention tasks. Second, we describe how the Invention Lab adapts the task to students' demonstrated proficiencies, thus supporting students while maintaining the exploratory nature of the task. Last, we illustrate how intelligent support can aid learning at the domain and scientific reasoning levels using log-files from a classroom evaluation of the lab.

2   The Invention Lab

The Invention Lab facilitates a type of inquiry activities called invention tasks. In invention tasks students are asked to invent novel methods for calculating target properties of data [4,15]. Figure 2 shows the Trampoline problem (from Figure 1) as it appears in the lab. In this example students are asked to invent a method for comparing the variability of two datasets. Invention tasks use contrasting cases to direct students' attention to deep features of the domain [4]. For example, the contrasting cases in Figure 2 (region (2c)) share the same average and sample size, but differ in their range. Following the invention attempt, students receive direct instruction on the canonical solutions for the same problem, and practice applying these solutions to
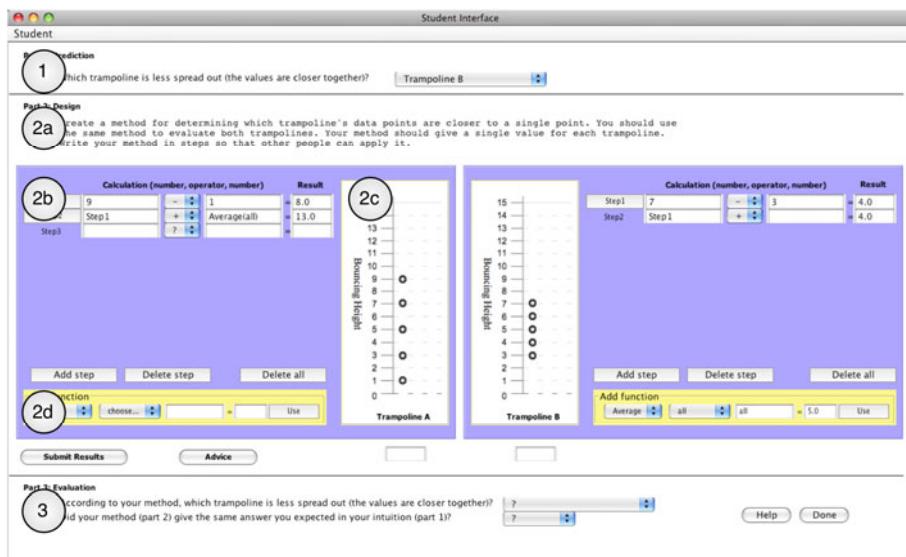
Fig. 2. The Invention Lab interface. Students are first asked to rank the contrasting cases (1). Upon successful ranking, the system asks the students to invent a method that will reflect their ranking (2a). The students express their method in steps (2b), using points from the contrasting cases (2c), and using basic functions (2d). Last, students evaluate their method (3) and revise it as needed. Students repeat the cycle using different contrasting cases.

different problems. The instruction and practice are done outside the lab. For example, the invention task from Figure 2 is followed by direct instruction on mean absolute deviation (the average distance from the mean). Multiple classroom studies have shown that invention tasks followed by direct instruction and practice lead to more robust learning compared with direct instruction and practice alone [4,15]. This effect was also termed "productive failure" [16], since the benefits of invention tasks were found even though most students failed to invent valid methods.

The Invention Lab facilitates invention tasks at the middle- and high-school levels. We first describe the interaction flow in the Invention lab from the students' point of view. We then describe the intelligent components of the system.

Students begin their invention activity by ranking two given contrasting cases according to the target concept (e.g., the data for Trampoline B is less spread out; region (1) in Figure 2). This qualitative ranking serves as the baseline against which students can later evaluate their inventions [4].

Upon successful ranking students move on to the design phase (region (2a)). In this phase students design a method for calculating the spread of the data. In previous studies we found that students prefer to express their methods as a sequence of steps (rather than a unified formula, see Figure 3). The lab retains this characteristic by supporting design in steps (region (2b) in Figure 2). Each step has the simple form of number - operator - number. While students need to invent a general method, they need not express it as such. Instead, in order to reduce cognitive load, students
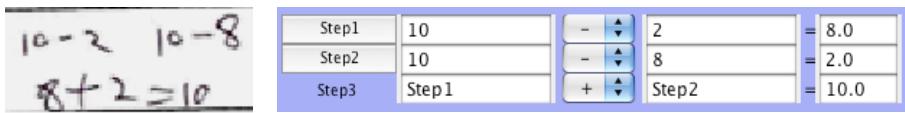
Step1	10	−	2	= 8.0
Step2	10	−	8	= 2.0
Step3	Step1	+	Step2	= 10.0

Fig. 3. A method designed by a student on paper (left) and the same method in the lab (right)

demonstrate their method by instantiating it using the given contrasting cases. For example, in order to "design" range, students enter "9-1" in the left graph by clicking on the "9" data-point (region (2c)), choosing the minus sign, and clicking on the "1" data-point. Students can also add basic functions to their methods (sum, average, median, and count; region (2d)). Last, students can add and delete steps, and use the results from previous steps in a current step.

Students apply their method to both contrasting cases and then submit it. No feedback is given during the design phase with the single exception of checking upon submission whether the same method was applied to both contrasting cases.

The evaluation phase (region (3) in Figure 2) asks students to compare the outcomes of their methods to their initial ranking. Feedback on evaluation incorporates an intelligent novice model, in that students are given the opportunity to notice the limitations of their method and revise it prior to receiving feedback [17]. When the invented method fails to generate the correct ranking (as established in the qualitative analysis), the system points that out and prompts the students to revise their method. When the method generates the correct ranking the system analyzes the method and identifies conceptual errors (for example, using range does not take into account other data points). The system uses this information to generate new contrasting cases that target the identified knowledge gaps. The number of ranking > design > evaluation cycles is not limited, and tasks are designed to engage students for 30 minutes. Students usually use the lab in pairs, though no explicit support for collaboration was implemented.

2.1 Intelligent Support in the Invention Lab

Support at the inquiry level. The Invention Lab scaffolds the inquiry process using cognitive tools (such as the step-based formula builder). The explicit scaffold of the inquiry process (i.e., *ranking -> design -> evaluation*) makes it a good candidate for applying a model-tracing approach. By tracing students' actions using a cognitive model of the inquiry process, the Invention Lab can give students adaptive and domain-independent feedback on their progress in the inquiry process. For example, when students fail to notice that predictions derived from their methods do not match their initial ranking, the tutor responds by explicitly pointing out that "your answer to the last question is not the same as your initial prediction."

Identifying errors at the domain level. As described above, evaluating students' methods at the domain level is particularly challenging. The virtually intractable interaction space makes it hard to trace students' actions, and the under-defined solution space makes it hard to evaluate complete solutions as a whole. Therefore, instead of pre-defining the complete set of solution classes, we chose to define the set of

requirements from a valid solution (without defining the solutions themselves). This is done using Constraint-Based Modeling (CBM, [18]). CBM is a modeling approach that evaluates whether students' answers satisfy (or violate) a set of domain-specific constraints. Each constraint is associated with a characteristic that is required from all correct solutions. Therefore, all methods that violate a specific constraint reflect a similar knowledge gap. This quality of CBM makes it suitable for ill-defined domains and tasks [19]. The Invention Lab uses CBM to test whether the invented methods capture the deep features of the domain. For example, valid solutions should use all data points to calculate variability. However, many of the methods described above use only the extreme data points (e.g., range). The Invention Lab need not represent all the possible ways of demonstrating this knowledge gap. Rather, it can identify when the only arguments used by a method are the extreme values. Therefore, one of the constraints in the Invention Lab specifies that students should not use only the extreme values. A comprehensive list of 6 target features with 14 common errors (i.e., violated constraints) was compiled based on students' inventions in a previous paper-and-pencil study [4]. Table 1 shows a subset of these features. Notice that each solution can violate more than one constraint. Additional constraints help the lab give feedback on general mathematical errors, such as inconclusive methods.

To the extent that every algebraic method (at the middle school level) can be expressed using the Invention Lab interface and every invention can be analyzed according to the features described above, the cognitive model of the lab can analyze any method. Both components of the cognitive model (model tracing and CBM) were implemented with 59 production rules written in Jess using the Cognitive Tutor Authoring Tools [20]. The entire cognitive model could have also been implemented using a CBM approach.

Table 1. Analyzing methods using constraints. The four bottom rows show methods invented by students during the classroom studies, when applied to sample data ({2,4,4,7,8}).

Target feature (constraint):	Variability is determined by all data points		Variability depends on sample size	Variability depends on distances
Common conceptual errors (violations):	Method uses only extreme values	Method uses a sequential subset of points	Method does not control for sample size	Method does not calculate distances
Range * 2 *(8-2)*2 = 16*	X		X	
Largest gap *(7-4) = 3*		X	X	
# of close points *N({2,4,4}) = 3*		X	X	X
(Min+max) / # of points *(2+8) / 5 = 2*	X			X

Designing domain level support. Like many other ITS, the Invention Lab uses its evaluation of students' knowledge to adapt the task to students' demonstrated proficiencies. The Invention Lab does so by giving students contrasting cases that direct students' attention to the limitations of their previous methods and help them encode the deep structure of the domain. However, using canned contrasting cases may not be the right way to go. Students in the Invention Lab reveal their conceptual errors in different ways, when analyzing different data. Canned contrasting cases may lead students to create a collection of ad-hoc methods, resulting in scattered bits of knowledge. Instead, the Invention Lab designs in real time new contrasting cases to match students' needs. Each common conceptual error has an associated method for generating new contrasting cases that target that error. The process is designed to ensure that new sets of cases are easy to compare with regard to the target concept (so ranking will be simple), and that the most recent lacking method would fail on them. For example, if the student uses only extreme values, the system will generate two new cases that share the same range but have distinctive variability. Last, the process uses the recent set of cases, to help students build upon their prior experiences and create more cohesive knowledge. Table 2 demonstrates this process.

Table 2. An example for the contrasting-cases generation algorythm. Steps 1 and 2 are generic. Step 3 changes based on the target conceptual error. The given example targets the use of only extreme values to determine variability.

	Case A	Case B	Comments
Original task			
Original cases:	1 3 5 7 9	3 4 5 6 7	
Invented method: range	9 - 1 = 8	7 - 3 = 4	
New task			
1. Keep the case with the higher variability from the previous cycle	**1 3 5 7 9**		This encourages students to transfer from previous experiences
2. Copy the values that were used by the student in her previous method to the other case	1 3 5 7 9	**1** **9**	This ensures that the pervious method fails to distinguish between the cases in the new set.
3. Populate the other case with values that are halfway between the original case and the average	1 3 5 7 9	1 **4 5 6** 9	This ensures that the two sets have distinct variability, the same average, and are easy to judge perceptually. (Halfway between 3 and average is 4; Halfway between 5 and average is 5; Halfway between 7 and average is 6)

2.2   Evaluating the Invention Lab

The Invention Lab was evaluated with 92 students in a public middle school in the Pittsburgh area. While the outcomes of the study are outside the scope of this paper,

log files from the study offer us a window into students' learning with the lab[1]. The following example spans the first 20 minutes of one pair of students trying to invent a method for calculating variability.

First, the students receive the contrasting cases shown in Figure 2: *(Trampoline A: 1,3,5,7,9; Trampoline B: 3,4,5,6,7)*. The students rank the cases correctly *("Trampoline B has lower variability")*, and "invent" range *(Trampoline A: "9-1=9"; Trampoline B: "7-3=4")*.

The lab analyzes the method and chooses what feature to focus on next. In this case, the highest priority is to help the students understand that variability is determined by more than merely the extreme values. The lab generates new contrasting cases that fix range and keep one of the previous cases intact *(Trampoline A: 1,3,5,7,9; Trampoline B: 1,4,5,6,9; see Table 2)*.

The students rank the cases correctly *("Trampoline B has lower variability")*, and begin their design by applying the previously successful method, range. *(Trampoline A: "9-1=8"; Trampoline B: "9-1=8")*. The students fail to notice that range gives the same result for both contrasting cases, which does not match their initial ranking. Therefore, the students receive detailed feedback from the system ("*your answer to the last question is not the same as your initial prediction. Please check your method*").

The students then attempt several central tendency measures (such as mean and median), however, these methods fail to generate correct ranking (since the contrasting cases share the same mean and median). This time around the students notice the failure of their methods and do not submit them for evaluation.

The students then try range again, and scrap the method before submitting it.

At this point, the log-files reveal a mini a-ha moment, when the students realize that they can extrapolate range to the second-furthest pair of points in each graph. They first list the two distances without relating them to one another *(Trampoline A, step 1: "9-1=8", step 2: "7-3=4"; Trampoline B, step 1: "9-1=8", step 2: "6-4=2")*. They submit this method, at which point the system prompts them that their method is inconclusive, since it does not assign a single value for each graph. The students, possibly encouraged by their success to extrapolate distance to other numbers, apply the concept of distance to the distances themselves. In other words, they subtract step 2 from step 1: *(Trampoline A, step 1: "9-1=8", step 2: "7-3=4", step 3: "step1-step2=4"; Trampoline B, step 1: "9-1=8", step 2: "6-4=2", step 3: "step1-step2=6")*. The students submit this method, but then realize its shortcoming – the method predicts that Trampoline A has lower variability, which is different from their initial ranking. They delete the method before approving it and resume the drawing board.

The students invent few additional methods before trying to add up the distances: *(Trampoline A, step 1: "9-1=8", step 2: "7-3=4", step 3: "step1+step2=12"; Trampoline B, step 1: "9-1=8", step 2: "6-4=2", step 3: "step1+step2=10")*. This method produces the desired ranking, thus concluding the current cycle.

This snippet of interaction reveals an interesting learning trajectory. First, we can see that the cognitive model of the lab identifies valid features, and its contrasting

---

[1] Results, to be detailed elsewhere, show that students who designed methods in the Invention Lab acquired more robust learning compared with students who were not instructed to design new methods. These results echo the findings of an earlier paper-and-pencil study [23].

cases generator creates contrasting cases that achieve their goal. Second, we can see how students' thinking unfolds, and how their mental models of spread evolve (including the a-ha moment before extrapolating range). Last, in addition to students' progression at the domain level, the students came across rich experiences at the scientific inquiry level. The students improved their tendency and ability to evaluate their methods. They also encountered the limitations of inconclusive methods that do not give a single number, and found that the same method always gives the same result when applied to the same data. Other log files demonstrate the ability of the Invention Lab to interpret novel methods. For example, several students have continued the line of reasoning demonstrated above, and designed a method that averages all "recursive ranges" (i.e., distances between highest and lowest, second highest and second lowest, etc.) We did not anticipate this solution, which is not a common measure of spread. However, upon closer examination, we concluded that recursive-range is a valid measure of spread. Indeed, when encountered by the Invention Lab, the lab concluded that this method satisfies all required constraints[2].

3   Summary and Contributions

We describe the Invention Lab, an ELE for facilitating invention tasks. The lab uses a hybrid of modeling techniques: it applies a model tracing approach to trace students' inquiry behavior, and applies a CBM approach to evaluate domain-level inputs. The Invention Lab also creates contrasting cases in real time. This allows the system to adapt its tasks to individual students without reducing critical features of the inquiry process. Last, we demonstrate how the combination of scaffold and feedback at the inquiry and domain levels helps students develop sophisticated mathematical models and improves their understanding of scientific reasoning. While the lab supports one type of inquiry tasks, invention activities, it demonstrates the feasibility of adding intelligent support at the domain and inquiry levels to ELE, thus bridging two schools of thought in the field of educational technologies.

References

1. de Jong, T., van Joolingen, W.R.: Scientific discovery learning with computer simulations of conceptual domains. Rev. of Ed. Res. 68, 179–201 (1998)
2. Duffy, T.M., Cunningham, D.J.: Constructivism: Implications for the design and delivery of instruction. In: Jonassen, D.H. (ed.) Handbook of research for educational communications and technology, pp. 170–198. Macmillan, New York (1996)

---

2 To be exact, "averaged recursive range" works well only for data with an even number of points. No student was successful at extending this method to data with an odd number of points. A more complete (and conventional) solution is to use mean absolute deviation (the averaged distance from the mean).

3. VanLehn, K.: The Behavior of Tutoring Systems. Int. J. Artif. Intell. Educ. 16(3), 227–265 (2006)
4. Roll, I., Aleven, V., Koedinger, K.R.: Helping students know 'further' - increasing the flexibility of students' knowledge using symbolic invention tasks. In: The 31st Annual Conference of the Cognitive Science Society, pp. 1169–1174. Cognitive Science Society, Austin (2009)
5. Shute, V.J., Glaser, R.: A Large-Scale Evaluation of an Intelligent Discovery World: Smithtown. Interactive Learning Environments 1, 51–77 (1990)
6. Veermans, K., de Jong, T., van Joolingen, W.R.: Promoting Self-Directed Learning in Simulation-Based Discovery Learning Environments Through Intelligent Support. Interact. Learn. Envir. 8(3), 229–255 (2000)
7. Mulder, Y.G., Lazonder, A.W., de Jong, T.: Finding out how they find it Oout: An empirical analysis of inquiry learners' need for support. Int. J. Sci. Educ., 1–21 (2009)
8. Jonassen, D.: Designing constructivist learning environments. In: Reigeluth, C.M. (ed.) Instructional-design theories and models, pp. 215–239. Routledge, Hillsdale (1999)
9. Woolf, B.P., Marshall, D., Mattingly, M., Lewis, J., Wright, S., et al.: Tracking student propositions in an inquiry system. In: The 11th International Conference on Artificial Intelligence in Education, pp. 21–28. IOS Press, Sydney (2003)
10. Roll, I., Aleven, V., McLaren, B.M., Koedinger, K.R.: Designing for metacognition - applying cognitive tutor principles to the tutoring of help seeking. Metacognition and Learning 2(2), 125–140 (2007)
11. Koedinger, K.R., Suthers, D.D., Forbus, K.D.: Component-Based construction of a science learning space. Int. J. Artif. Intell. Educ. 10, 292–313 (1999)
12. Bunt, A., Conati, C.: Probabilistic student modelling to improve exploratory behaviour. User Model. User-Adap. 13(3), 269–309 (2003)
13. Koedinger, K.R., Corbett, A.T.: Cognitive tutors: Technology bringing learning science to the classroom. In: Sawyer, K. (ed.) The Cambridge Handbook of the Learning Sciences, pp. 61–78. Cambridge University Press, New York (2006)
14. Luckin, R., du Boulay, B.: Ecolab: The development and evaluation of a vygotskian design framework. Int. J. Artif. Intell. Educ. 10(2), 198–220 (1999)
15. Schwartz, D.L., Martin, T.: Inventing to prepare for future learning: The hidden efficiency of encouraging original student production in statistics instruction. Cognition Instruct. 22(2), 129–184 (2004)
16. Kapur, M.: Productive failure. Cognition Instruct. 26(3), 379–424 (2008)
17. Mathan, S.A., Koedinger, K.R.: Fostering the intelligent novice: Learning from errors with metacognitive tutoring. Educ. Psychol. 40(4), 257–265 (2005)
18. Mitrovic, A., Koedinger, K.R., Martin, B.: A comparative analysis of cognitive tutoring and constraint-based modeling. In: Brusilovsky, P., Corbett, A.T., de Rosis, F. (eds.) UM 2003. LNCS (LNAI), vol. 2702, pp. 313–322. Springer, Heidelberg (2003)
19. Mitrovic, A., Weerasinghe, A.: Revisiting ill-definedness and the consequences for ITSs. In: The 14th Conference on Artificial Intelligence in Education, pp. 375–382. IOS Press, Marina Del Ray (2009)
20. Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.) ITS 2006. LNCS, vol. 4053, pp. 61–70. Springer, Heidelberg (2006)