The Effect of Adapting Feedback Generality in ITS

Brent Martin and Antonija Mitrovic

Intelligent Computer Tutoring Group
Department of Computer Science and Software Engineering
University of Canterbury
Private Bag 4800, Christchurch, New Zealand
{brent,tanja}@cosc.canterbury.ac.nz

Abstract. Intelligent tutoring systems achieve much of their success by adapting to individual students. One potential avenue for personalization is feedback generality. This paper presents two evaluation studies that measure the effects of modifying feedback generality in a web-based Intelligent Tutoring System (ITS) based on the analysis of student models. The object of the experiments was to measure the effectiveness of varying feedback generality, and to determine whether this could be performed *en masse* or if personalization is needed. In an initial trial with a web-based ITS it appeared that it is feasible to use a mass approach to select appropriate concepts for generalizing feedback. A second study gave conflicting results and showed a relationship between generality and ability, highlighting the need for feedback to be personalized to individual students' needs.

1 Introduction

Intelligent tutoring systems (ITS) achieve much of their success by adapting to individual students. One potential avenue for personalization is feedback generality. Feedback in ITS is usually very specific. However, in some domains there may be low-level generalizations that can be made where the generalized concept is more likely to be what the student is learning. For example, Koedinger and Mathan [2] suggest that for their Excel Tutor (one of the cognitive tutors [1]), the concept of relative versus fixed indexing is independent of the direction the information is copied; this is a generalization of two concepts, namely horizontal versus vertical indexing. We hypothesized that this might be the case for our web-based tutor (SQL-Tutor). For example, an analysis of the feedback messages found that often they are nearly the same for some groups of concepts. Other concepts may differ only by the clause of the SQL query in which they occur (for example, the WHERE and HAVING clauses of an SQL query have substantially similar concepts).

Some systems use Bayesian student models to represent students' knowledge at various levels (e.g. [8]) and so theoretically they can dynamically determine the best level to provide feedback, but this is difficult and potentially error-prone: building Bayesian belief networks requires the large task of specifying the prior and conditional probabilities. We were interested in whether it was possible to infer a set of high-level concepts that generally represent those being learned while

avoiding the difficulty of building a belief network, by analyzing past student model data to determine significant subgroups of knowledge units that represent such general concepts. Feedback can then also be attached to these more general concepts and selected according to students' needs.

One method of analyzing knowledge units is to plot learning curves: if the objects being measured relate to the actual concepts being learned, we expect to see a "power law" between the number of times the object is relevant and the proportion of times it is used incorrectly [6]. Learning curves can be plotted for all knowledge units of a system to measure its overall performance. They can also be used to analyze groups of objects within a system, or to "mine" the student models for further information. We used this latter approach to try to determine which groups of domain knowledge units appear to perform well when treated as a single concept. To decide which ones to group, we used a (man-made) taxonomy of the learning domain [3], and grouped knowledge units according to each node of the taxonomy. This enabled us to measure how well these units, when combined into more general ones of increasing generality, still exhibited power laws and hence represented a single concept that the students were learning. We then used this information as the basis for building a new version of the domain model that gave more general feedback.

In the next section we describe the system we used in the study. In Section 3 we present our hypotheses and discuss how we used the student models to predict the performance of groups of knowledge units. We then give the results for an initial experiment that tested a new feedback scheme based on these general concepts. Section 4 describes a second study, in which we modified the delivery of the generalized feedback, with some surprising results. Section 5 then discusses differences between the results of the two studies, while the conclusions are given in Section 6.

2 SQL-Tutor

The initial goal of this research was to investigate whether we can predict the effectiveness of different levels of feedback by observing how well the underlying group of knowledge units appears to measure a single concept being learned. We performed an experiment in the context of SQL-Tutor, a web-based intelligent tutoring system that teaches the SQL database language to university-level students. For a detailed discussion of the system, see [4, 5]; here we present only some of its features. SQL-Tutor consists of an interface, a pedagogical module—which determines the timing and content of pedagogical actions—and a student modeler, which analyses student answers. The system contains definitions of several databases and a set of problems and their ideal solutions. SQL-Tutor contains no problem solver: to check the correctness of the student's solution, SQL-Tutor compares it to an example of a correct solution using domain knowledge represented in the form of more than 650 constraints. It uses Constraint-Based Modeling (CBM) [7] for both domain and student models.

Feedback in SQL-Tutor is attached directly to the knowledge units, or "constraints", which make up the domain model. An example of a constraint is:

```
(147
; feedback message
"You have used some names in the WHERE clause that are not
from this database."

; relevance condition
 (match SS WHERE (?* (^name ?n) ?*))

; satisfaction condition
 (or   (test SS (^valid-table (?n ?t))
       (test SS (^attribute-p (?n ?a ?t))))

"WHERE")
```

Constraints are used to critique the students' solutions by checking that the concept they represent is being correctly applied. The relevance condition first tests whether or not this concept is relevant to the problem and current solution attempt. If so, the satisfaction condition is checked to ascertain whether or not the student has applied this concept correctly. If the satisfaction condition is met, no action is taken; if it fails, the feedback message is presented to the student. The student model consists of the set of constraints, along with information about whether or not it has been successfully applied, for each attempt where it is relevant. Thus the student model is a trace of the performance of each individual constraint over time. Constraints may be grouped together, giving the average performance of the constraint set as a whole over time, for which a learning curve can then be plotted. Figure 2 shows the learning curves for the two groups of the first study, for all students and all constraints. This is achieved by considering every constraint, for every student, and calculating the proportion of constraint/student instances for which the constraint was violated for the first problem in which it was relevant, giving the first data point. This process is then repeated for the second problem each constraint was used for, and so on. Both curves in Figure 2 (Section 3) show an excellent power law fit ($R^2 > 0.92$). Note that learning curves tend to deteriorate as *n* becomes large, because the number of participating constraints reduces.

3 Study 1: Does Feedback Generality Have an Effect?

We hypothesized that some groupings of constraints would represent the concepts the student was learning better than the (highly specialized) constraints themselves. We then further hypothesized that for such a grouping, learning might be more effective if students were given feedback about the general concept, rather than more specialized feedback about the specific context in which the concept appeared (represented by the original constraint). To evaluate the first hypothesis, we analyzed data from a previous study of SQL-Tutor on a similar population, namely second year students from a database course at the University of Canterbury, New Zealand. To decide which constraints to group together, we used a taxonomy of the SQL-Tutor domain model that we had previously defined [3]. This taxonomy is very fine-grained, consisting of 530 nodes to cover the 650 constraints, although many nodes only cover a single constraint. The deepest path

in the tree is eight nodes, with most paths being five or six nodes deep. Figure 1 shows the sub tree for the concept “Correct tables present”.

We grouped constraints according to each node in the taxonomy, and rebuilt the student models as though these were real constraints that the system had been tracking. For example, if a node N1 in the taxonomy covers constraints 1 and 2, and the student has applied constraint 1 incorrectly, then 2 incorrectly, then 1 incorrectly again, then 2 correctly, the original model would be:

(1 FAIL FAIL)
(2 FAIL SUCCEED)

while the entry for the new constraint is:

(N1 FAIL FAIL FAIL SUCCEED)

Note that several constraints from N1 might be applied for the same problem. In this case we calculated the proportion of such constraints that were violated. We performed this operation for all non-trivial nodes in the hierarchy (i.e. those covering more than one constraint) and plotted learning curves for *each* of the resulting 304 generalized constraints. We then compared each curve to a curve obtained by averaging the results for the participating constraints, based on their individual models. Note that these curves were for the first four problems only: the volume of data in each case is low, so the curves deteriorate relatively quickly after that. Overall the results showed that the more general the grouping is, the worse the learning curve (either a poorer fit or a lower slope), which is what we might expect. However, there were eight cases for which the generalized constraint had superior power law fit and slope compared to the average for the individual constraints, and thus appeared to better represent the concept being learned, and a

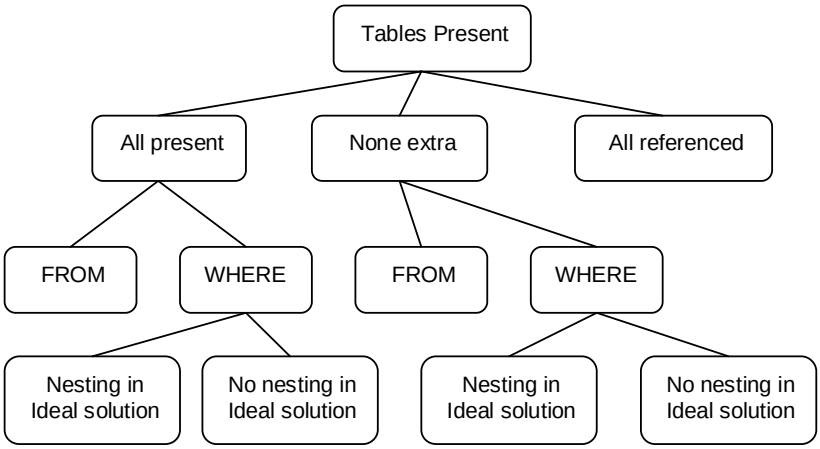

Fig. 1. Example sub tree from the SQL –Tutor domain taxonomy

further eight that were comparable. From this result we tentatively concluded that some of our constraints might be at a lower level than the concept that is actually being learned, because it appears that there is “crossover” between constraints in a group. In the example above, this means that exposure to constraint 1 appears to lead to some learning of constraint 2, and vice versa. This supports our first hypothesis.

We then tested our second hypothesis: that providing feedback at the more general level would improve learning for those high-level constraints that exhibited superior learning curves. Based on the original analysis we produced a set of 63 new constraints that were one or two levels up the taxonomy from the individual constraints. This new constraint set covered 468 of the original 650 constraints, with membership of each generalized constraint varying between 2 and 32, and an average of 7 members (SD=6). For each new constraint, we produced a tuple that described its membership, and included the feedback message that would be substituted in the experimental system for that of the original constraint. An example of such an entry is:

```
(N5 "Check that you are using the right operators in
numeric comparisons." (462 463 426 46 461 427 444 517 445
518 446 519 447 520 404 521 405 522))
```

This generalized constraint covers all individual constraints that perform some kind of check for the presence of a particular numeric operator. Students for the experimental group thus received this feedback, while those in the control group were presented with the more specific feedback from each original constraint concerning the particular operator.

To evaluate this second hypothesis we performed an experiment with the students enrolled in an introductory database course at the University of Canterbury. Participation in the experiment was voluntary. Prior to the study, students attended six lectures on SQL and had two laboratories on the Oracle RDBMS. SQL-Tutor was demonstrated to students in a lecture on September 20, 2004. The experiment was performed in scheduled laboratories during the same week. The experiment required the students to sit a pre-test, which was administered online the first time students accessed SQL-Tutor. The pre-test consisted of four multi-choice questions, which required the student to identify correct definitions of concepts in the domain, or to specify whether a given SQL statement is appropriate for the given context.

The students were randomly allocated to one of the two versions of the system. The course involved a test on SQL on October 14, 2004, which provided additional motivation for students to practice with SQL-Tutor. A post-test was administered at the conclusion of a two-hour session with the tutor, and consisted of four questions of similar nature and complexity as the questions in the pre-test. The maximum mark for the pre/post tests was 4.

Of the 124 students enrolled in the course, 100 students logged on to SQL-Tutor at least once. However, some students looked at the system only briefly. We therefore excluded the logs of students who did not attempt any problems. The logs of the remaining 78 students (41 in the control, and 37 in the experimental group) were then analyzed. The mean score for the pre-test for all students was 2.17 out of 4 (SD=1.01). The students were randomly allocated to one of the two versions of

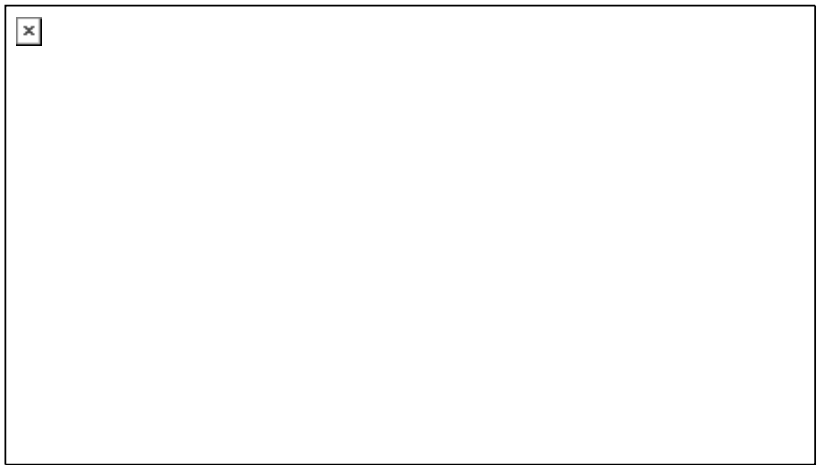

Fig. 2. Learning curves for the two groups

the system. A t-test showed no significant differences between the pre-test scores for the two groups (mean=2.10 and 2.24 for the control and experimental groups respectively, standard deviation for both=1.01, p=0.53).

Figure 2 plots the learning curves for the control and experimental groups. Note that the unit measured for both groups is the *original* constraints, because this ensures there are no differences in the unit being measured, which might alter the curves and prevent their being directly compared. Only those constraints that belong to one or more generalized concepts were included. The curves in Figure 2 are comparable over the range of ten problems, and give similar power curves, with the experimental group being slightly worse (control slope = -0.86, R^2 = .94; experiment slope = -0.57, R^2 = 0.93).

Although the generalized constraints used were loosely based on the results of the initial analysis, they also contained generalizations that appeared feasible, but for which we had no evidence that they would necessarily be superior to their individual counterparts. The experimental system might therefore contain a mixture of good and bad generalizations. We measured this by plotting, for the control group, individual learning curves for the generalized constraints and comparing them to the average performance of the member constraints, the same as was performed for the *a priori* analysis. The cut-off point for these graphs was at n=4, because the volume of data is low and so the curves rapidly degenerate, and because the analysis already performed suggested that differences were only likely to appear early in the constraint histories. Of the 63 generalized constraints, six appeared to clearly be superior to the individual constraints, a further three appeared to be equivalent, and eight appeared to be significantly worse. There was insufficient data about the remaining 46 to draw conclusions. We then plotted curves for two subsets of the constraints: those that were members of the generalized constraints classified as better, same or 'no data' (labeled “acceptable”), and those classed as worse or 'no data' (labeled “poor”). Figure 3 shows the curves for these two groups.

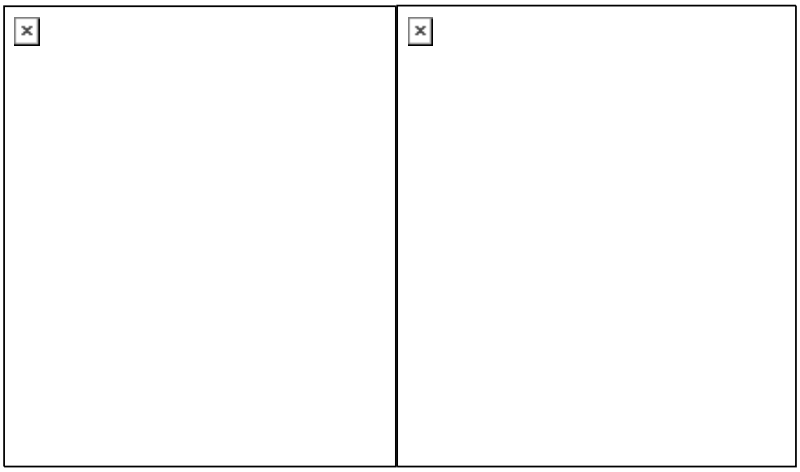

Fig. 3. Power curves based on predictions of goodness

For the "acceptable" generalized constraints, the experimental group appears to perform considerably better for the first three problems, but then plateaus; for the "poor" generalized constraints the experimental group performs better for the first two problems only. In other words, for the "acceptable" generalizations the feedback is more helpful than the standard feedback during the solving of the first two problems in which it is encountered (and so students do better on the second and third one) but is less helpful after that; for the "poor" group this is true for the first problem only. We tested the significance of this result by computing the error reduction between n=1 and n=3 for each student and comparing the means. The experimental group had a mean error reduction of 0.058 (SD=0.027), compared to 0.035 (SD=0.030) for the control group. The difference was significant at p=0.01. In contrast, there was no significant difference in the means of error reduction for the "poor" group (experimental mean=0.050 (SD=0.035), control mean=0.041 (SD=0.028), p>0.3). This result again suggests that the individual learning curves do indeed predict to some extent whether generalized feedback at this level will be effective. It also suggested that personalization may not be necessary; simply applying the same feedback to all students appeared to (initially at least) improve learning performance.

4 Study 2: Does Generalization Help?

Based on the results of the 2004 study, we concluded that generalized feedback seemed to work well initially but if feedback is needed too many times there reaches a point where it no longer helps the student. We hypothesized therefore that starting with more general feedback and later switching to specific feedback might yield the best results. We modified the experimental system to behave in this way: students in the experiment group received general feedback the first two times a constraint was violated, then the same feedback as the control group (i.e.

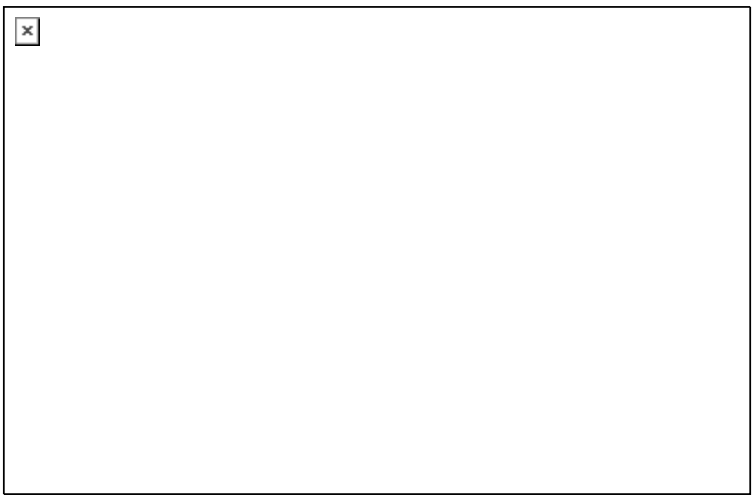

Fig. 4. Learning curves for the two groups

specific) thereafter. We included only those generalizations deemed "acceptable" in the previous study. If our hypothesis were correct we would expect the curve for the experimental group to be steeper at the beginning than the control, and then the same once the feedback has reverted to the same feedback as the control. Overall the experimental group should learn faster.

The experiment was run in October 2005, again using students from a year 2 database course at the University of Canterbury, New Zealand. The number of students participating in the experiment was lower this time; after we excluded those students who did not attempt any problems there were 21 students in the control group and 25 in the experimental group. The mean score for the pre-test for all students was slightly lower than in 2004: 2.02 out of 4 (SD=0.98) compared to 2.14 (SD=1.01) in 2004. The students were randomly allocated to one of the two versions of the system. The experimental group had a higher average than the control group (2.22, SD=1.04 Compared to 1.81, SD=0.87), although in an independent-samples T-test the result was not statistically significant (p=0.2).

Figure 4 again plots the learning curves for the control and experimental groups, for the first 10 problems that each knowledge unit was relevant. This time the experimental group performed much more poorly than the control; the control group reduced their error by 64% on average after receiving (specific) feedback for two problems, whereas the experimental group only reduced their error by 8% after receiving the more general messages. Further, on average they *increased* their error by 16% between the first and second problem. This result directly contradicts the previous experiment.

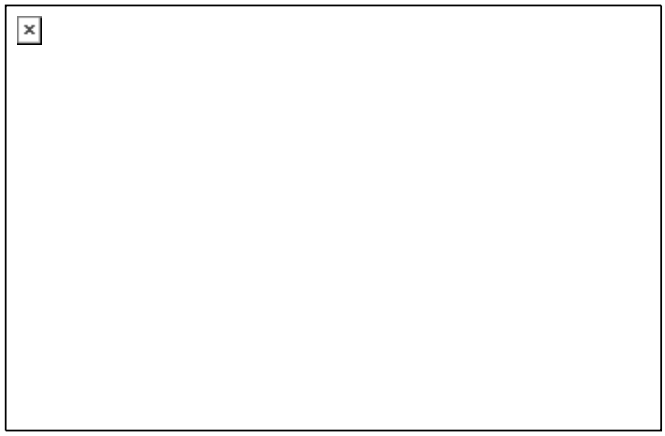

Fig. 5. Error reduction versus pre-test score for the experimental group, study 2

5 Discussion

At first glance the second study suggests that the method used to determine which concepts to make more general is not robust. However, another possibility is that feedback generality is not something that can be applied *en masse* to all students. Figure 5 plots error reduction over the first two problems versus pre-test score for the experimental group. Error reduction for this group is quite strongly correlated with pre-test score (slope = 23, R^2 = 0.67), indicating that poorer students may have difficulty understanding more general feedback. This trend was also observed for the 2004 study, although the effect was much weaker (slope = 3, R^2= 0.016). In contrast, for the control groups in both years error reduction is slightly *negatively* correlated: poorer students reduce their error more. The results for the control group for both years were nearly identical (slope = -4, R^2 = 0.015). This suggests that the system may need to adapt generality to the ability level of the student, perhaps varying the level over time as the student gains proficiency.

A difference in the experimental groups' experiences is that in 2004 the students received general feedback for the same subset of concepts *all of the time*, whereas in 2005 feedback switched back to specific messages after the general message had been shown twice. Perhaps this led to confusion; the student might have thought they had corrected an error because the feedback changed, and were now looking for a different error to fix; the error messages did not reference each other, so the student might quite reasonably infer that they referred to two different problems. For example, *"Check whether you have specified all the correct comparisons with integer constants"* might change to *"Check the constants you specified in WHERE!"* In particular, the second feedback message does not specify the *type* of constant (integer) whereas the first message does, so might equally apply to string constants for example. Less able students may have suffered this misconception, whereas the better students did not.

6 Conclusions

Formatted: Bullets and Numbering

In this experiment we researched the effect of feedback generality on learning performance. We initially used past student model data for predicting the behavior of generalized feedback. We developed a more general feedback set that mapped to groups of underlying knowledge units, and found in an initial experiment that for some of these concepts learning performance appeared to improve, although only for the first two problems, after which it deteriorated. For other generalizations performance was worse. We also showed that we could predict to some extent which generalized constraints would produce better performance by analyzing their apparent performance in the control group. A second study contradicted the first; students given more general feedback initially exhibited *worse* performance. However, the experimental system differed between the two studies: in the first study generalized feedback was given all the time for selected concepts, while in the second it was only given initially and then the system reverted to giving specialized feedback.

In the second study the effect of the generalized feedback differed between students, with a strong trend indicating that less able students failed to cope with the feedback given. Since this trend was not observed in the first study, it also suggests that the less able students may have been confused when the feedback level *changed*. The problems observed with the second study might possibly have been obviated if it was made clear to the student that the feedback was still referring to the *same error*, e.g. if the later feedback included both messages, rather than switching from one to another.

The two studies show that feedback generality has a measurable effect on learning ability. Between them they also give hope that the general concepts can be inferred from past student model data, and indicate that the level of generality needs to be tailored to individual students. This motivates us to continue to explore how we can best personalize feedback to maximize student performance.

References

1. Anderson, J.R., Corbett, A.T., Koedinger, K.R., and Pelletier, R., *Cognitive Tutors: Lessons Learned.* Journal of the Learning Sciences, 1995. **4**(2): 167-207.
2. Koedinger, K.R. and Mathan, S. *Distinguishing Qualitatively Different Kinds of Learning Using Log Files and Learning Curves*. In: J. Mostow, P. Tedesco (eds) Proc. *Workshop on Data Mining of Student Logs at ITS2004*. 2004. Maceio, Brazil, pp. 39-46.
3. Martin, B., *Constraint-Based Modelling: Representing Student Knowledge.* New Zealand Journal of Computing, 1999. **7**(2): 30-38.
4. Mitrovic, A., *An Intelligent SQL Tutor on the Web.* Artificial Intelligence in Education, 2003. **13**(2-4): 173-197.
5. Mitrovic, A., Martin, B., and Mayo, M., *Using evaluation to shape ITS design: Results and experiences with SQL-Tutor*. User Modelling and User Adapted Interaction, 2002. **12**(2-3): 243-279.
6. Newell, A. and Rosenbloom, P.S., *Mechanisms of skill acquisition and the law of practice*, in *Cognitive skills and their acquisition*, J.R. Anderson, Editor. 1981, Lawrence Erlbaum Associates: Hillsdale, NJ. pp. 1-56.
7. Ohlsson, S., *Constraint-Based Student Modeling*, in *Student Modeling: The Key to Individualized Knowledge-Based Instruction*, J. Greer and G. McCalla, Editors. 1994, Springer-Verlag: New York. pp. 167-189.
8. Zapata-Rivera, J.D. and Greer, J.E., *Interacting with Inspectable Bayesian Student Models.* Artificial Intelligence in Education, 2004. **14**(2): 127-163.