A User-Driven and Data-Driven Approach for Supporting Teachers in Reflection and Adaptation of Adaptive Tutorials

Dror Ben-Naim, Michael Bain, Nadine Marcus
(drorb, mike, nadinem)@cse.unsw.edu.au
School of Computer Science and Engineering
University of New South Wales, Sydney, Australia

Abstract: It has been recognized that in order to drive Intelligent Tutoring Systems (ITSs) into mainstream use by the teaching community, it is essential to support teachers through the entire ITS process: Design, Development, Deployment, Reflection and Adaptation. Although research has been done on supporting teachers through design to deployment of ITSs, there is surprisingly little discussion about support for teachers' Reflection - the ability to draw conclusions from ITS usage, and Adaptation - adapting the content to better meet the needs of students. We describe our work on developing analysis tools and methodologies that support reflection and adaptation by teachers. The work was done in the context of helping teachers understand student's behavior in Adaptive Tutorials by post-analysis of the system's data-logs. We used a hybrid solution – part of the data-mining effort is teacher driven and part is automated. We tested our approach by comparing the results of expert analysis of two Adaptive Tutorials with and without an automated Refinement Suggestion Tool, and found it to be a useful teacher's aid. By using this tool, teachers act as 'action researchers', confirming or disproving their hypotheses about the best way to use ITS technology.

1 Introduction

Intelligent Tutoring Systems (ITSs) can dramatically increase learners' comprehension by adapting the learning activity to the learners' needs, based on an intelligent assessment of their level of knowledge. This is the "Dream of ITS" (cf. "The Dream of AI") – that one day a system will be "smart" enough to teach better than human teachers. Whether this dream is to become a reality is arguable, even as ITS technologies are being intensively researched by the scientific community. In recent years, it has been recognized that whether or not the dream is realized, we must make ITSs as widely available as traditional web based educational systems. However, this is not a straightforward task, partially due to the sheer amount of content existing in traditional web based systems, compared with the relatively small amount of specialized content existing in ITSs[4], and also due to the complex nature of ITS's and their relative inaccessibility to teachers. In order to address this issue, teachers require better support through the entire ITS process: Design, Development, Deployment, Reflection and Adaptation.

To-date, research on supporting teachers in the ITS process has been focused on aiding teachers to author intelligent content, mainly through the advent of ITS authoring tools[10], but it is now clear that the ITS design paradigm needs to be updated. A new design paradigm offers teachers a different place in the ITS process; while the core authoring is in the hands of well-prepared design teams, teachers can extend the system and fine tune it to meet their specific needs[4].

This shift in the teacher's role is also acknowledged in the work of Diana Laurillard who proposed the Conversational Framework for the effective use of educational technology[6]. The Conversational Framework (CF) can be considered both a learning theory and a practical framework for designing educational environments. It models the interaction between teachers and learners as a stepwise "conversation" across four dimensions: discussion, adaptation, interaction and reflection. In [7], Laurillard describes the role of the teacher as an "action researcher", "collaborating to produce their own development of knowledge about teaching with technology". However, she also argues that support for reflection and adaptation is severely lacking with regards to eLearning content. This is because teachers rarely have the ability to reflect on (analyze and conclude) and adapt (change or edit) software based instructional material. The argument is even stronger for intelligent content offered by specialized systems such as ITSs.

This paper presents work that aims to support teachers through the process of the reflection and adaptation of Adaptive Tutorials (AT's) running on the Adaptive eLearning Platform (AeLP)[2]. An important challenge we faced in analyzing the Adaptive Tutorials in the AeLP was how to develop data-mining tools for the purpose of aiding teachers, without becoming too domain-specific or overwhelming them with a large number of association rules or classifiers which are difficult to understand. In particular, we aim to ensure the tool is easy to use and do not want to cognitively overload the teachers[14]. Moreover, students' interaction in the AeLP can vary dramatically between different AT's. Our contribution is through developing a refinement and adaptation strategy that can scale across different domains. We achieve this though a hybrid approach – user-driven and data-driven. The user-driven approach manifests itself in the development of an interactive analysis and discovery tool called the Adaptive Tutorial Analyzer (ATA). Teachers use the ATA for the purpose of analyzing students' performance in Adaptive Tutorials. The data-driven approach manifests itself in the development of a Refinement Suggestion Panel that draws teachers' attentions to patterns in the data that requires their attention. In this paper we show how both of these strategies complement each other.

2 Related work

Analyzing student behavior in an ITS is a complex problem, and the task of making sense of the data in ITS's logs is within the domain of educational data-mining[13]. Generally speaking, educational data mining is a data-driven field motivated to augment human-programmed knowledge, e.g. to ease the modeling of the correct way a problem should be solved ([8]), or to accurately predict a student's performance based on analysis of previous years' logs ([9]). However, some researchers previously highlighted the fact that patterns found in educational systems' data-sets are only useful if interpreted in the pedagogical context of the educational activity. In the work of [5] the researchers used an iterative process of discovery and interpretation with the goal of making sense of patterns discovered by data-mining algorithms they used.

We followed similar reasoning: patterns in the data-logs of Adaptive Tutorials are senseless without a teacher's pedagogical and domain specific insights. However, unlike [5] who rely solemnly on analysis of click-streams, the AeLP logs the entire system's internal state per each student's 'check' event (student pressing the 'check' button). As such, the data-logs are extremely multidimensional, up-to hundreds of attribute-values per student action. Furthermore, the system's snapshot depends on the specifics of the Virtual Apparatus (VA) that was used for the Adaptive Tutorial (see [2] for a description of how Adaptive Tutorials are constructed from Virtual Apparatuses), and as such we need tools that are domain independent but that can be utilized for the purpose of domain specific inquiry.

Another comprehensive study on analyzing ITS's data-logs was carried by [11] where data-mining algorithms were used in order to analyze the logs of a Constraint-Based ITS called SQL-Tutor. The researchers used a variety of tools such as WEKA and SQL in order to carry out multiple analysis tasks that resulted in some refinement suggestion to their system. One difference in our work is that the AeLP is a platform on which 10 different adaptive tutorials, each equivalent to SQL-Tutor in its scope and depth, are currently running. Our approach is thus to enable teachers to conduct analysis tasks, rather than specialist data-mining researchers. Furthermore, while the AeLP does use constructs analogues to Constraints (called trap-states), for the authoring of adaptive activities, it also uses solution traces, that are closer to Model Tracing based ITS's. This suggests that a richer knowledge representation is required for automated analysis.

Work on employing mining and visualization in order to analyze students' trails in a web-based educational system is also discussed in [12]. The data-set is again a navigation pattern or a "click-stream" and the researchers' approach was to interpret the student's navigation as a graph – considering each hypertext page as a node and transition between pages as edges. The tool is meant to be used as an aid for teachers to better understand student navigation. While similar to our concept to the AT-Analyzer, our efforts differ again in that the trails, or traces we are concerned with are not simply HTML pages requested, but traces through an entire solution state-space within an Adaptive Tutorial (see [3] for detailed explanation).

3 The Adaptive eLearning Platform

The Adaptive eLearning Platform (AeLP) is a web-based implementation of Virtual Apparatus Framework for eLearning content development[2]. The AeLP is used for authoring Adaptive Tutorials, deploying them to students or into LMSs, monitoring student progress and analyzing student behavior. The AeLP has been fielded since 2006 at the University of New South Wales, where Adaptive Tutorials developed using the AeLP have been incorporated into the syllabi of 10 major courses (ranging between 50 to 600 students per semester), and are accessed by over 2000 students per semester.

From a pedagogical point of view, AT's are similar in nature to teaching laboratory activities and are analogous to the concept of Tutorial Simulations as described in [6]. AT's exhibit three levels of adaptivity: students experience adaptive feedback with remediation targeted to their intrinsic misconceptions, while their activities are also sequenced adaptively based on performance. The third level of adaptivity is content adaptation through analysis and reflection. Teachers are provided with analysis tools that enable reflection and adaptation of their content. By analyzing students' behavior, teachers can refine and adapt their content, to better meet the needs of their students, e.g.: changing questions, adding new adaptive feedback or changing the sequence of activities. The work described in this paper concerns development of tools and processes to better facilitate this level of adaptation.

4 User-Driven and Data-Driven Analysis Strategy

We presented our work on the AT-Analyzer in [3]. The analysis of adaptive tutorials is always performed with the purpose of refining and improving them for the next time they run. Teachers perform analysis on past AT-Sessions (instances of running an AT on a group of students), while the changes are saved to the next AT session. In that sense we support the Conversational Framework notion of teachers acting as "action researchers", interested in affirming or disproving their hypotheses regarding their content and its effect on learners[7]. Based on their analysis, teachers then need to be able to revise and change - to adapt - their content.

4.1 The Interaction-Snapshot Data Log

For each student interaction event, the AeLP stores a student-identifiable, time-stamped snapshot of the entire system's inspectable state-space. This state-space contains generic AeLP properties (e.g. *session.attemptNumber*, or *inputPanel.selectedChoice*) and the entire internal state the VA is in (e.g. *VA.propertyA* and *VA.propertyB*). The combined set of attribute-values is the student's Interaction-Snapshot-Vector. In addition to the interaction snapshot, the data also contains a trap-state ID. This ID is a unique identifier of the trap-state that was fired when processing the student's interaction. This trap-state can either be "correct" thus allowing the student to progress in their activity, or it could be an error-state, which contains some feedback to be shown to the student. In this way, the log database contains not only what the students were doing, but also the system's decision over their interactions.

4.2 An Example Adaptive Tutorial

As an example, consider an Adaptive Tutorial that was developed for a 1st year course in Solid Mechanics: the Bridge Inspection Simulator [Figure 1]. This AT features a bridge simulation, in which students can "drive" a car on a 3 section bridge. Students can position the car in different locations on the bridge sections, and take load and shear stress measurements on the bridge's poles and cables using virtual sensors. Here is an illustrative example question in this Adaptive Tutorial: "A second car C2 of mass m2 is positioned on section C (right hand side cantilever) of the bridge at x=250m. Position your car C1 of mass m1 on section A (left hand side cantilever) such that the tension on both sections' cables is the same. Enter the tension in Newtons in the input panel." The correct trap-state is defined as: *car1.x = 60 AND userInput = 60.* The teacher then defines an error trap-state that targets a familiar misconception. For example if a student positions the car at *car1.x = 50*, the teacher knows that they answered under the false assumption that *m1 == m2*, which is incorrect. A trap-state called *sameMassError* will target the condition *car1.x = 50* and will feature a hint feedback that will tell the students to look carefully for the masses in the question. The other trap-state for this system will be just the empty *defaultWrong* trap-state, which means that any student who did not enter 50 or 60 will be given some generic default feedback (e.g. “Wrong, Try Again.”). We will return to this example in the following sections.

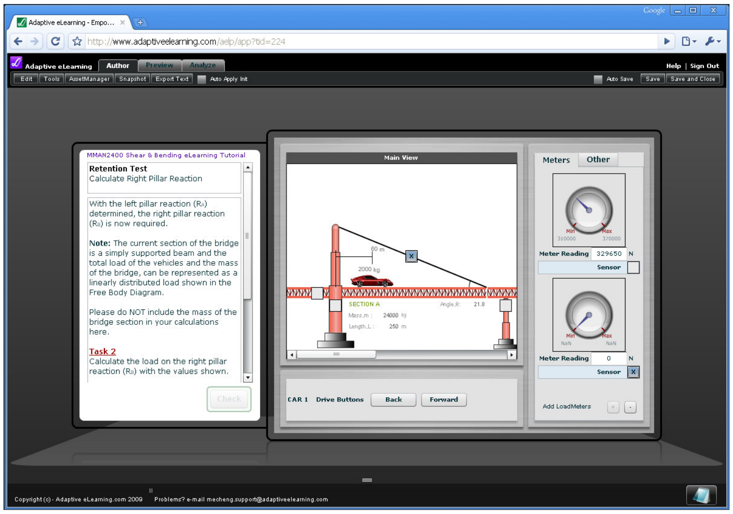

Figure 1: An AT in Mechanical Engineering: students can “drive” a car through different sections of a bridge and use sensors to inspect the load on different elements.

4.3 User-Driven Analysis Support – The Interactive Solution Trace Graph

In the Virtual Apparatus Framework, one can think of the process a student takes in order to solve a task as a trace through the problem’s state-space. The idea behind the Solution Trace Graph [Figure 2] is to visualize the time-based vector of interaction-snapshots as a graph transition where each column is a solution attempt and each edge represents a transition between solution trap-states. Working with the Solution Trace Graph, teachers drill down on interaction data in order to gain insights regarding students’ behavior, of which some of the most important are:

Finding adaptive feedback that was ineffective: if a high proportion of interactions entering a trap-state ended up landing back in the same trap-state on the next attempt column, it might imply that the feedback was not helping the students. We call this condition a trap-state’s self-loop. For example, if 50% of students who landed in *sameMassError* landed again in *sameMassError* in their next attempt, the teacher might conclude that his feedback did not help the students to understand their mistake, and might change it, to be more specific.

Specializing an overly general trap-state: let’s assume that 50% of students answered the aforementioned question correctly in the first attempt, 20% landed in *sameMassError*; the teacher will be interested to inspect what happened with the remaining 30% of students who landed in the *defaultWrong* trap-state. Using the STG, the teacher will inspect the interaction-snapshots leading into *defaultWrong*, and might notice a pattern, say, that 70% of those entered 100m as the answer. When researching why such a mistake was so prominent, the teacher might notice that students' mistake was that they used a + instead of a – in their calculation. The teacher will then simply add a new trap-state: *plusMinusConfusionError*, targeting this misconception.

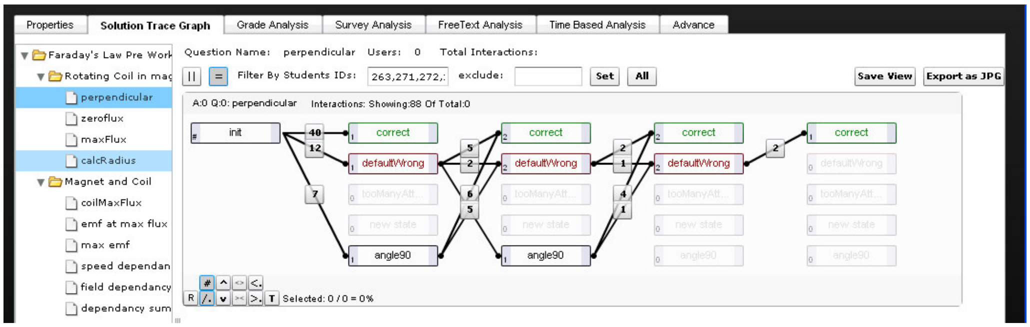

Figure 2 – A Solution Trace Graph is used to visually analyze students' solution-traces through the problem's state-space. In this example it is easy to see that 40 out of 59 students attempting this question answered correctly on the first attempt and that 6 out of the 7 who landed in the *angle90* trap-state proceed to *correct* after given the adapted feedback.

4.4 Data Driven Analysis Support - The Refinement Suggestion Panel

Based on our experience with the AT-Analyzer and the STG, it became clear that some aspects of the reflection work could be automated. Subsequently, a Refinement Suggestion Panel (RSP) was designed [Figure 3]. It offers teachers a list of the most relevant issues that might need their attention. Such suggestions, for example, highlight the fact that a question is too easy, or too hard, that a particular adaptive feedback seems to be ineffective, that a new trap-state should be defined, and more.

The issues discovered are ordered by calculated relevance, and teachers can chose to dismiss an issue or act on it. Relevancy is measured by functions specified for particular aspects of the type of suggestion.

Finding adaptive feedback that was ineffective: automating the detection of this type of issue is relatively simple: an algorithm that exhausts all edge transitions in the entire AT's STG, and sorts the results on self-loop ratios across solution attempts, was developed. The RSP then presents the teacher with a list sorted in descending order. The top case is the trap-state with the highest self-loop ratio (weighted relative to edge count size, so that a 40 out of 50 ratio will appear before an 8 out of 10).

Specifying an Overly General Trap-State: in order to detect this type of issue, we need an automated way to search for association rules in each trap-state's interaction-snapshot data. In the example above, we are interested that the RSP will show the teacher the fact that 70% out of the 30% who landed in *defaultWrong*, entered 100m. Remembering that snapshots can contain tens or hundreds of attributes (defined by the VAs API, e.g. *car1.mass*, *sectionA.mass* etc.), the immediately apparent problem is how to get rid of all the non-interesting rules, containing attributes that are meaningless from an educational point of view. In other words, how do we target the *car.x* attribute?

One way to solve this is to look for what extra information is available to us. If we look at the question-attributes-set - the set of all attributes that are used in a question's existing trap-states, we will then see that the teacher targeted *car.x* and *userInput*. This information gives us a clue about what attributes are useful and we will only perform an association rule and feature selection search on this limited set of attributes. For the example shown in [Figure 3], we identified an overly general trap-state - *defaultWrong* (the antecedent of *defaultWrong* is the negation of all other custom trap-states, in other words- if no other trap-state fired, *defaultWrong* is fired). The association rule that was found has the antecedent containing the condition: *….angleControl.value == 70* with coverage = 12/59 students and confidence = 0.42 (5/12). By capturing this new rule as new trap-state, the teacher will refine the overly general *defaultWrong*.

Figure 3 – The Refinement Suggestion Panel draws teachers' attention to possible issues that might need their attention. Teachers can act on those suggestions by capturing them as new trap-states.

Ranking the relevancy of this type of refinement suggestions is based on coverage and confidence, (thresholds of these parameters are user defined). For each coverage-level cohort we sort results by confidence before adding it to the RSP.

An obvious down-side of this approach is that patterns including attributes that are not in the question-attribute-set cannot be found in this way, and brought to the attention of the teacher. For example, it is possible that some of the students who made the *sameMassError* also put a "virtual load sensor" on the wrong "docking station", and thus were reading an erroneous value for their calculation. In this case, the interaction-snapshots of these students will contain the attribute value *sensor1.dockStation=1*. This association rule between the two attributes values *sensor1.dockStation=1 -> userInput=100m* is extremely important from a pedagogical point of view, but cannot be found by the RSP.

However, using the STG, the teacher can initiate an “all-in” rule search, that might find this association rule. Furthermore, the teacher can choose to look for associations and features on any subset of snapshot attributes, e.g. adding to the two sensors’ “dock stations” attributes to the searched attribute set. If the teacher then decides to add a new trap-state using a new attribute, this new attribute now belongs to the question-attributes-set, and subsequently *will* be used by the RSP’s data-driven analysis.

In this way, the user-driven and data-driven analyses complement each other, leveraging expert knowledge with data-mining efficiency, yielding a powerful yet teacher-friendly analysis tool.

5 Results and Analysis

For the purpose of a preliminary study regarding its usefulness, we used the RSP to generate suggestions for two Adaptive Tutorials: the Bridge Inspection Simulation and the Faraday’s Law Tutorial in Physics (also described in [2]). The former was already analyzed by the teacher, using the ATA and the STG, while the bridge activity was not. In the case of the Faraday’s law AT, we compared the refinement suggestions given by the RSP to the teacher’s analysis in order to see if we were able to replicate their refinement actions. In the case of the bridge tutorial the teacher worked with the RSP in their analysis and investigated its usefulness.

The Faraday’s law AT was run on a group of 59 students in the second half of 2007 and resulted in 982 interactions in the database, each containing a snapshot of 14 to 18 attribute-values representing the state of the system and VA per a user ‘check’ event. We ran the RSP on the Faraday AT’s data log with a limit of 2 suggestions per question and we got a total of 28 suggestions. The top 3 ranked suggestions matched the same exact three improvements that were found by the teacher. For example: in a question that asked students to rotate a magnetic coil situated in constant magnetic field to the angle that will result in maximum magnetic flux through it (*correct* trap-state is *VA.angleControl.value = 0*), the RSP suggested for refinement the *defaultWrong* trap-state. It appeared that 5 students out of the 12 landing on the *defaultWrong* trap-state had the *VA.angleControl.value = 70*, which is in fact the question’s *initState* [Figure 3]. In other words - those 5 students did not attempt to solve the question at all, and just pressed ‘check’, possibly attempting to game the system[1]. The teacher added a trap-state targeting the following conditions: *VA.angleControl.value == 70 AND session.timeOnQuestion < 15 seconds*, and attached feedback that politely asked the students to actually attempt solving the question by manipulating the VA’s control. Out of the remaining new 25 suggestions, the teacher chose to use 5 and dismissed the rest. The teacher’s impressions were very positive and they found the tool both easy to use and understandable.

The Bridge AT was used by 220 students during the second half of 2008 and generated a total of 7014 interaction entries in the database. A typical interaction entry included a system snapshot of around 18 attribute-value pairs. This time we limited the RSP to show only the top 10 suggestions, and 5 of them were accepted by the teacher as valid. Again, most suggestions targeted the *defaultWrong* trap-state – the state that most needed further specifications. All 5 states accepted by the teacher were found to be dealing with students not properly attempting a question, or selecting ‘check’ prematurely. This analysis has two conclusions: it supports the development approach of continual-refinement where an AT is developed initially to only remediate to the most obvious misconceptions and misbehaviors, and using the analysis tool the teacher gradually refine its rule base. The second conclusion is that the RSP algorithm described above does not work well when the tutorial questions are parameterized: in the Bridge AT, each student was given a different set of initial parameters (bridge length and height, masses of cars etc) and thus the *correct* trap-states are defined as functional dependencies between attributes-values and not constants. We discuss this further in the next section.

While further work still remains, we found that overall the teachers response was positive and the RSP is an important step forward in helping teachers understand how the AT’s are being used by their students.

6 Future work

Based on our analysis, it appears that further work needs to be done on dealing with functional dependency between attributes. Consider the following simple example question: “Calculate the force the car is applying on the bridge (in Newtons), and enter it in the input panel.” But this time, assume the Virtual Apparatus is set to randomize the car mass for each student. We now need to define the *correct* trap-state as: *userInput == car.mass*9.81*. In this case, the snapshot attributes will contain functional relationships between attributes, and a simple attribute association rule or feature selection test will not be able to identify any patterns. Possible future work is to allow that when functional relationship between attributes appears in a question’s condition set, we provide the data-mining algorithm with a test that encodes that functional dependency. For example, we can define a new variable, *V=inputPanel.userInput / car.mass* and do feature selection on *V*. The RSP can then discover high probability for *V=1* which occurs when students forgot to multiply by the gravitational constant. This is an important feature, because it lets us incorporate relational logic in the rule association search in a manner that is easy for the teacher to understand.

7 Conclusion:

We have presented a hybrid analysis strategy for Adaptive Tutorials. A key aspect of our work is the fact the Adaptive Tutorials are constructed using the Virtual Apparatus Framework, thereby enabling rich content with a high degree of interactivity to be authored. This, however, presents challenges for the analysis of student activity in such complex environments. Towards this objective we implemented a user-driven analysis tool – the Solution Trace Graph, and complemented it with a data-driven analysis tool – the Refinement Suggestion Panel. We showed in this paper that one way in which the two approaches complement each other is that when a teacher adds a new attribute into a question’s condition-set the RSP includes this attribute in its automated rule-finding algorithm. Based on a preliminary study and analysis, we found that the combined strategy was successful in leveraging the experts’ domain knowledge to direct the data-mining process, improving effectiveness and efficiency.

By building such evaluation tools and techniques into ITS technology we allow teachers to understand and reflect on students' behavior, and subsequently adapt activities to better match student knowledge levels and address misconceptions. In that sense, and in accordance with the Conversational Framework, the teacher is acting as an active educational researcher, confirming or disproving their hypotheses about the best way to use ITS technology in pursuit of their pedagogical goals.

8 References

1. Baker, R.S., Corbett, A.T., Koedinger, K.R, *Detecting Student Misuse of Intelligent Tutoring Systems.* Proceedings of the 7th International Conference on Intelligent Tutoring Systems, 2004: p. 531-540.

2. Ben-Naim, D., N. Marcus, and M. Bain, *Virtual Apparatus Framework Approach to Constructing Adaptive Tutorials*, in *The 2007 International Conference on E-Learning, E-Business, Enterprise Information Systems, and E-Government*, A.B. Hamid R. Arabnia, Editor. 2007, CSREA Press: Las Vegas, Nevada, USW. p. 3-10.

3. Ben-Naim, D., N. Marcus, and M. Bain, *Visualization and Analysis of Student Interaction in an Adaptive Exploratory Learning Environment*, in *The 1st Int. Workshop in Intelligent Support for Exploratory Environments in the European Conference on Technology Enhanced Learning*. 2008, CEUR-WS: Maastricht, The Netherlands.

4. Brusilovsky, P., Knapp, J. and Gamper, J., *Supporting teachers as content authors in intelligent educational systems.* Int. J. Knowledge and Learning, 2006. **2**(3/4): p. 191-215.

5. Hübscher, R. and S. Puntambekar. *Integrating Knowledge Gained From Data Mining With Pedagogical Knowledge*. in *Educational Data Mining 2008: 1st International Conference on Educational Data Mining*. 2008. Montreal, Quebec, Canada.

6. Laurillard, D., *Rethinking University Teaching: A Conversational Framework for the Effective Use of Learning Technologies*. 2002: Routledge. 268.

7. Laurillard, D., *The teacher as action researcher: using technology to capture pedagogic form.* Studies in Higher Education, 2008. **33**(2): p. 139 - 154.

8. McLaren, B.M., et al. *Bootstrapping Novice Data: Semi-Automated Tutor Authoring Using Student Log Files*. in *Proceedings of the Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes. 7nth International Conference on Intelligent Tutoring Systems (ITS-2004)*. 2004.

9. Merceron, A. and K. Yacef. *Interestingness Measures for Association Rules in Educational Data*. in *1st International Conference on Educational Dat Mining (EDM08)*. 2008. Montreal, Canada.

10. Murray, T., *Authoring intelligent tutoring systems: an analysis of the state of the art.* International Journal of Artificial Intelligence in Education, 1999. **10**: p. 98-129.

11. Nilakant, K. and A. Mitrovic. *Application of data mining in constraint-based intelligent tutoring systems*. in *artificial intelligence in education, AIED*. 2005.

12. Romero, C., et al. *Mining and Visualizing Visited Trails in Web-Based Educational Systems*. in *Educational Data Mining 2008: 1st International Conference on Educational Data Mining*. 2008. Montreal, Quebec, Canada.

13. Romero, C. and S. Ventura, *Educational data mining: A survey from 1995 to 2005.* Expert Systems with Applications, 2008. **33**(1): p. 135-146.

14. Sweller, J., J. Van Merriengoer, and F. Paas, *Cognitive architecture and instructional design.* Educational Psychology Review, 1998. **20**: p. 251-296.