Work in Progress - Lessons-Constructor-Analyzer Paradigm (LCA) and the Animated Database Courseware (ADbC)

Mario Guimarães and Meg Murray
Department of Computer Science and Information Systems
Kennesaw State University
1000 Chastain Rd.
Kennesaw, GA, 30144
mguimara@kennesaw.edu, mcmurray@kennesaw.edu,

Abstract - The teaching learning process can be fairly complex, especially when students encounter abstract concepts. Research shows that student learning is facilitated when students are provided with multiple channels of instruction, opportunity for practice and accompanying feedback loops. One supplemental instructional vehicle is the use of educational software. We propose a paradigm by which instructional software can be designed, modeled, implemented and evaluated. The paradigm, referred to as Lessons-Constructor-Analyzer (LCA), divides classroom activities into three groupings, Lessons, where material is presented to the students and a set of basic questions asked, Constructor, where the student has a microworld or a simulator in which to practice and develop skill, and Analyzer, in which assessment and evaluation of learning takes place. In this paper, we describe how the LCA paradigm can be applied against the evaluation of instructional software through the example of our development of a set of software animations designed to support the teaching and learning of database concepts known as the Animated Database Courseware (ADbC). Faculty and student evaluations indicate that the application of the LCA paradigm to the development and incorporation of the ADbC software has been effective. ADbC is supported by NSF Grant # 0717707

Index Terms - animation, database, educational software, courseware, design, SQL, security, transactions, concurrency.

1. THE LESSONS-CONSTRUCTOR-ANALYZER (LCA)

We propose a paradigm for educational (instructional) software. The paradigm is a Lessons-Constructor-Analyzer (LCA). This paradigm divides classroom activities into the Lessons, where the material is presented to the students and a set of basic questions are asked, the Constructor, where the student has a microworld or a simulator in which to develop his/her skills, and the Analyzer, that helps the instructor evaluate learning performed with the Lessons and the Constructor.

Lessons may include anything from an introductory tutorial to sophisticated animations containing multiple windows where students associate prior knowledge with new knowledge. The association is done by highlighting correspondence between windows and animating concepts in a step by step view. Lessons allow students to learn at their own pace, through a rich set of material in a short amount of time. If the concepts of these animations are shown manually, it consumes a huge amount of chalk and time to write on a board, and would be impossible to show the material any other way. However, the Lessons consist of basically passive learning, where the student is visualizing, instead of actively trying out new ideas. We believe the Lessons have great value if they are integrated with Construction and Evaluation.

This paradigm is compatible to the Report of the ITCSE Working Group on “Improving the Educational Impact of Algorithm Visualization in Computer Science Education” [1], visualization is most valuable when it is used to engage learners in active learning activities. Note that this paper is not suggesting that all educational software should have the three modules built in. Our Animated Database Courseware (ADbC), as well as many other softwares that aid teaching, are not intended to be used in isolation. Educational Software should be incorporated into a classroom to supplement lectures and labs. Thus, the active learning component may be present inside the software or within the greater context of a lab, homework or in-class exercise.
Although our goal is for ADbC to always be used within a greater context, we are incorporating changes that will add to ADbC all the components of our Paradigm.

2. **ADbC UPDATES – 2007-2008 ACADEMIC YEAR**

ADbc currently consists of four modules, 20 sub-modules and over 100 programs. There are many ways that this software may be applied: 1) teachers use it while explaining a new topic in a classroom setting; 2) students use it as a reinforcements of old topics; for example, a student in an advanced database class or a graduate class reviews the software to recall what they learned in the past. 3) students reference it when they are doing homework and they can’t find a solution to their problem. 4) students use it to learn new topics; use of it as a stand-alone, self explanatory software. The best way to use each animation depends on the specific sub-module the student is in.

ADbc has four major modules: Database Design, SQL, Transactions and Security. During the academic year of 2007-2008, the following improvements were made to ADbC:

1) A user guide Containing over 100 pages describing the animations available as well as best practices on how to use the Database Courseware
2) A re-designed navigational interface
3) 8 new animations for the Scenario to ER sub-module of the Database Design module,
4) Two new sub-modules, Functional Dependencies and Anomalies of the Database Design Module.
5) A new Security Module with six sub-modules.
6) 23 programs/animations enhanced.
7) A tool to construct ER Diagrams (sub-module) of the Database Design Module
8) 25 quiz questions for the Interactive SQL sub-module of the SQL Module.

Note that the last two items on the list above are currently under testing, while all the other items have been tested and incorporated into ADbC.

3. **DATABASE DESIGN AND LCA**

The Database Design module consists of ER Notations, Scenario to ER, ER to Tables, Functional Dependencies, Normalization, Denormalization and Anomalies. Like all other modules of the system it is rich in the Lessons component of the LCA where the user is presented to new concepts through text and animations/visualizations. The first sub-module of the system, Database Design -> ER notation, is an example of the Lessons component. Since there is no standard ER Notation, this sub-module introduces ER Diagrams and allows the user to visualize six different notation sets for seven different types of relationships used in constructing ER Diagrams. Figure 2 displays the ER Notation sub-module where a 1 to many binary relationship is displayed in two different notations. This is an example of the Lesson Component of the ER Diagram.

In order to supplement the Lessons component of ER Diagrams, we recently developed a tool for constructing ER Diagram (Constructor component). There is a main menu at the top and a drag able toolbox that is shown on the right. From the Scenario Menu on the left, the user may currently select a scenario. Both scenarios are problem descriptions that require a user to construct an Independent Relationship, a Dependent Relationship, a Recursive/Unary Relationship and a super-type, sub-type relationship. The Check menu will verify if the Entity Relationship Diagram that the user constructed with the toolbox corresponds to the specific Scenario.

The Database Design Module also contains an Analyzer/Evaluation component. There are 9 exercises. For each exercise, a user is given a scenario and asked to identify the ER Diagram which best depicts the information provided. Figure 4 shows one of the exercises.
In the ER to tables sub-module, there is an Introduction (Lessons component) and several exercises (Constructor component). The user is asked to identify a correct way to convert and ER to tables. For each choice the user makes, an animation is displayed explaining the advantages or disadvantages of each solution. Both the Database Courseware and Database textbooks emphasize the fact that the main goal of a good Database Design is to avoid Insert, Update and Delete anomalies. We reinforced this, by developing an Anomalies sub-module (Constructor Component). In this sub-module, the user is asked to Insert, Update and Delete both a well design table (the preferred solution for converting an ER to a table) and the table that was a result of an erroneous solution for converting an ER. In all cases, the correct solution involves a simple Insert, Update or Delete, while the operations on the badly designed table required multiple Inserts, Updates and Deletes.

In the functional dependency sub-module also consists of a Lessons component and a Constructor component. In the Functional Dependency sub-module, the user is given a scenario and asked to identify functional dependencies. In the Normalization sub-module there is also a tutorial (Lessons) and a set of exercises (Constructor and Analyzer). For each exercise, the user is given functional dependencies and asked to convert the table to 3 NF.

4. SQL AND LCA

The SQL Module can be seen in Figure 5 consists of four sub-modules: Interactive SQL, Embedded SQL, Stored Procedures and Functions, and Referential Integrity. The Interactive SQL module is the most refined sub-module. This sub-module is rich in the Lessons component and has several programs that have the Constructor Component. However, the constructor component must also be supplemented with an Interactive SQL lab. For each question in the lab that we assign, we also display the Module, Sub-Module and Program in the ADbC courseware that illustrate the same concept provided in the question the student is trying to add. Since this module was the richest in Lessons-Constructor components of the Lessons-Constructor-Analyzer, we chose to start implementing our Analyzer Component in this sub-module.

A twenty five questions quiz was elaborated on SQL. The student has the option of saving his/her profile and submit the work done to the instructor. Although initially, we considered this quiz by providing feedback on every question, for pedagogical purposes, we decided to follow the same structure found in the quizzes at www3 schools [2].

Figure 5 below shows the Referential Integrity sub-module. As the user tries to update a department record, he/she visualizes what happens to the corresponding Employees record. Depending on the clause that the table was created with (Restrict, Set Null, Cascade), the result of the operation will differ. This is an example of a Constructor component. The Evaluator/Analyzer component is performed manually.

Unlike the Database Design module, many of its sub-modules lack a Constructor or an Analyzer/Evaluation built into the software. However, the use of Lab Exercises and homework where the Constructor and/or Analyzer components are present has allowed this module to be as efficient as the Database Design module [3].
5. TRANSACTIONS AND LCA

The Transaction Module, as shown by Figure 4, consists of the Concurrency, Recovery and Triggers sub-modules. Except for improving the Lessons component by adding the help and hit on all programs and adding more functionality to the introduction to triggers program, there was no special effort to incorporate the LCA framework into this module. Although all three sub-modules have rich animations, only the Recovery sub-model has a Constructor and an Evaluation component. Preliminary evaluation results indicated that students were learning more from the Recovery sub-module. Therefore, we introduced Lab Exercises and homework (adding a Constructor and Analyzer component) that supplemented the other two sub-modules (that only has a Lesson component).

Figure 6 shows an example of the Lesson Component in the Concurrency Sub-module. Although this is a rich animation where the user can click the next button and visualize two different transactions accessing the same database table (Accounts table on the top of the screen) concurrently, it was not improving student learning until a constructor and an analyzer component was inserted through labs and homework (outside the software) [3].

6. SECURITY AND LCA

Security and its six sub-modules (Figure 5) have been recently implemented. For the security matrix, the Constructor component is done through a Project where students need to built the matrix and the evaluation is done through an on-line quiz as well as grading the projects. For the other four sub-modules, the Constructor component is done in the other sub-modules through labs. Figure 7 shows an example of the SQL injection.

Like the Concurrency displayed in Figure 6, there is a rich set of animations to introduce SQL injections that are controlled in a step by step. Also like the Transactions-Constructor sub-module, the animations in these sub-modules have little results if they are not complemented with the proper lab exercises that introduce the Constructor (active learning) component.

7. EVALUATION AND CONCLUSION

The LCA paradigm may be very useful for designing educational software. The three components supplement each other. Although it is desirable that an educational software contains all three components, the insertion of labs, homework and tests that supplement and integrate the missing components of the instructional software may work equally as well.

Many of ADbC’s sub-modules focus on only one or two components of the LCA paradigm. Other database coursewares that we have researched do the same. The focus is usually on the lessons component with animations [4, 5] or on the constructor component [6].

ADbC has gone through a series of evaluations. These include student evaluations, teacher evaluation, control groups and usability tests. These evaluations have indicated a need to add one of the missing components from the LCA paradigm to one or more of ADBC’s sub-modules. The missing components have either been incorporated into the software or added through labs, homework and tests. We conclude by emphasizing that how educational software is used and integrated into the teaching environment is of as much or more relevance than the educational software itself.

REFERENCES


[6] WINRDBI Educational Tool;
http://www.eas.asu.edu/~winrdbi/.