The Dilemma of Team Learning:
An Assessment from the SQL Programming Classroom

KEYWORDS

team learning, collaborative learning, active learning, pedagogy, programming instruction

ABSTRACT

Team Learning is one of the most thoroughly researched of all instructional methods. Many studies report students learn better in Team Learning classes than in the traditional lecture-only format. A smaller number of studies report no difference in student achievement. How does all of this apply to undergraduate programming instruction? This paper presents a method for using Team Learning in SQL programming classes, along with an evaluation of results over the past several years.

INTRODUCTION

Programming is one of the mainstays of undergraduate IS education. Today's IS students study at least one procedural language, along with Structured Query Language (SQL) for database applications and several scripting languages for tasks such as network administration and interactive web pages.

Most programming courses are built upon components: objects and action structures specific to each language. Objects, such as forms and database objects, consist of properties, methods and events. Action structures carry out tasks such as repeating a sequence of instructions a certain number of times or grouping the rows returned by a query according to the values found in a certain column. The programmer's job is to

express a given application in terms of tasks that can be implemented by the components of his/her language.

Programming instructors hope to introduce the components of a language and then move on to applications. An ideal course would involve initial applications that are straightforward uses of the components and subsequent problems that involve seeing interrelationships, producing something original from the components, and making judgements.

The traditional model for programming courses involves x number of lectures, followed by y number of assignments and z number of exam questions. Repeat until course ends. Almost all programming texts emphasize the importance of the assignments. These authors suggest that it is impossible to learn programming passively and that the active practice portion of the course is where the most important learning takes place.

Team learning provides us with the hope of expanding active practice in our courses. Extensive research [Cooper, et al] [Johnson, Johnson and Smith] shows that courses designed with this strategy offer the following benefits. First, the active nature of team learning promotes higher level thinking skills in students. Second, student achievement is increased. Students do more work than in the lecture-only format. Third, student retention of material is improved.

In a SQL programming class, this could mean that students do more work on complex applications, where interrelationships and original insights are required, without sacrificing lower-level mastery of the facts of objects and structures. Also, basic objects

and structures would be remembered throughout the term and beyond, resulting in a solid and robust foundation for later, more complex, applications.

A smaller body of research [Ryan, et al] [Lieux] reports no significant difference in student performance in team learning classes and traditional lecture classes. A comprehensive study [Fellers] of the issue as it relates to IS courses calls for more research in evaluating outcome measures such as performance or achievement.

TEAM LEARNING

Team Learning depends upon the idea that students, working in small groups on a professional-level problem, can teach themselves and each other the material of a course. The instructor of such a course shifts his/her role from information dispenser to course designer and classroom manager.

Team Learning is "one of the most thoroughly researched of all instructional methods [Slavin]." According to Johnson, Johnson and Smith:

> During the past 90 years more than 600 studies have been conducted by a wide variety of researchers in different decades with different age subjects, in different subject areas and in different environments. We know far more about the efficacy of cooperative learning than we know about lecturing, departmentalization, the use of instructional technology or almost any other facet of education.

Exhibit 1 summarizes the key findings of the research on the effectiveness of team learning as it relates to programming courses [Cooper, et al.].

Exhibit 1 Research on the Effectiveness of Team Learning	
Findings	**Authors**
More effective for promoting student higher-level thinking skills than lecture. More effective for promoting student learning and achievement than traditional instructional methods. More effective in increasing student retention than traditional learning methods.	Kulik and Kulik Smith (1977, 1980) McKeachie Johnson et al. Slavin Dansereau Tinto Astin Wales & Saeger Treisman
Reference: Adapted from Cooper, et al. 1990.	

In Exhibit 1, higher-level thinking refers to Bloom's taxonomy of Cognitive Levels [Bloom]. Exhibit 2 lists the levels of the Bloom model along with parallel student activities [Duch and Allen].

Exhibit 2 Bloom's Taxonomy of Cognitive Levels	
Cognitive Level	**Student Activity**
Evaluation	Making a judgement based on a pre-established set of criteria
Synthesis	Producing something new or original from component parts.
Analysis	Breaking material down into its component parts to see interrelationships.
Application	Using a concept to solve a problem.
Comprehension	Explaining/interpreting the meaning of material.
Knowledge	Remembering facts, terms, concepts, definitions, principles.
Reference: Duch and Allen 1996.	

ADAPTING TEAM LEARNING TO SQL PROGRAMMING

Team Learning, as adapted in this paper, has as its first goal to put the design of a SQL programming course on a foundation of objectives expressed in terms of the Bloom taxonomy. The key design question is "What do I want the students to be able to do when they finish this course?" In addition, we strove to maximize active practice that engages students at the levels of the Bloom taxonomy appropriate for the objectives.

The key attributes of our class were 1) the instructor designs the course to fulfill objectives expressed in terms of the Bloom taxonomy, 2) the instructor develops problems to support the objectives, 3) students teach themselves the language by working in groups on the problems, 4) on a day-to-day basis, the instructor manages the classroom work environment instead of acting as the source of all knowledge and activity.

For example, consider the SQL WHERE clause. This is usually taught immediately after students learn to specify a single table in a FROM clause. Low-level objectives might include the syntax of the clause and the operators and functions commonly used in the clause. Mid-level objectives could include understanding the role of the clause as a qualifier of rows, debugging syntax errors and using vendor documentation, while high-level objectives might deal with debugging logic errors, the kinds of problems that the clause can and cannot solve and what to do when faced with an unsolvable problem.

We taught the WHERE clause through the following problem posed to the student groups after they have read the text on the WHERE clause and taken a readiness assessment quiz (Readiness Assessment is discussed later in this paper).

> A university database contains a table named COURSE that contains the id and title of all courses offered by the institution. A person calls the registrar asking, "What courses on Unix do you offer?" How does the registrar answer the question?

The facetious, but sensible answer, to transfer the caller to the chair of the IS department came up early, and we allocated time to allow students to discuss this option. It was worth spending some time, since the solution the database eventually provides was not satisfactory to the caller.

Once students began thinking about a query, they eventually settled on the idea of scanning the title column for the string ' Unix.' In arriving at this strategy, they examined the role of the WHERE clause and the functions and operators used in the clause. Getting the clause to work taught syntax and the debugging of syntax errors.

The COURSE table contained titles such as “Introduction to Unix” and “Unix Administration,” which were returned by the query most students developed. The table also contained a title such as “Korn Shell Scripting,” which was not returned. The ensuing discussion eventually brought groups to decide that the database, as it is designed currently, cannot do an accurate job of answering the caller' s question. Suggestions of expanding the table to include a keywords column gave additional insights into the role of the WHERE clause and the choices among operators and functions that are involved it its use. Groups also questioned whether the scenario was important enough to merit a change in the design of the database. The final deliverable of the activity was a decision on this question accompanied by a written analysis.

We hoped that including Team Learning in this course would engage students more actively and at a higher level of thinking. Activities were designed so that working at this higher level affected the quality of the work at the lower levels as well. A second

purpose of the problem assignment was to get students to take ownership of their learning. Teaching IS students how to teach themselves a new technology, while facing a deliverable deadline, may be the most important thing we do in the classroom.

We also hoped to improve retention. In the WHERE clause problem mentioned above, the titles in the database were all caps, so scanning for the string "Unix" gives "no rows returned," the most instructive error message in the SQL environment. Once students applied functions to the WHERE clause to address this problem, we hoped they would remember to use these functions throughout the rest of the semester. This turned out to be the case, so the trading of lecture time for group work on a realistic problem increased the opportunity for these instructive "errors" to occur.

THE INSTRUCTIONAL ACTIVITY SEQUENCE

Michaelsen and Black present a framework for course design called the Instructional Activity Sequence [Michaelsen and Black]. The sequence begins with the desired educational outcomes. What does the instructor want students to be able to do when they have completed this unit of instruction? Bloom' s taxonomy serves as a guide for choosing objectives and for deciding at what level the instructor wants to engage the students during this unit. These decisions affect the choice of application-oriented activities presented later in the sequence.

Once the desired educational outcomes are identified, the instructor specifies the course content that students must know before beginning to work on the unit in class. In a programming class, this course content could include readings from the text on the new

material and programming exercises dealing with the material that led up to the current unit.

Readiness Assurance Process

The in-class activity on a new unit begins with what Michaelsen calls the Readiness Assurance Process. The purpose of Readiness Assurance is to determine what students have already learned on their own or from each other so that group work can build from that point. Readiness Assurance begins with a short quiz taken by students individually. Short answer and multiple choice questions are favored because they allow quick grading turnaround. The individual quiz ensures individual accountability and serves as a diagnostic tool for the instructor in planning later activities.

After the individual quiz is taken, but before it is graded and returned, students work on the same set of questions in their groups. During group tests, students exchange information with their peers. In the roles of teacher and learner, a student' s understanding of course concepts is broadened. Both types of quizzes count toward a student' s final grade. The weighting is up to the individual instructor.

After the individual and group quizzes, solutions or a list of key points that should have been covered should be distributed and discussed in class. During this feedback session, the instructor should be evaluating how well the students know the course content that the instructor identified as necessary to meet the desired educational outcomes. At this point, the instructor can also present any related material that may not have been covered in the readings.

A final feature of the Readiness Assurance Process is the Written Group Appeal. Appeals are an effective way to increase both learning and group cohesiveness. If a

group feels that their answer to a question should receive credit, even though it differs from the instructor' s answer, then the group should prepare a written appeal stating the question involved, their preferred correct answer, the basis for their appeal and the evidence that supports their point of view. When an appeal is granted, Michaelsen recommends giving the appropriate number of points to both the group and each individual in that group, but not members of other groups.

Individual and group quizzes, written group appeals and instructor feedback comprise the Readiness Assurance Process and lead to the Application-Oriented Activities. Students and instructor have confidence that the concepts necessary for the desired educational outcomes are understood, and students are ready to apply these concepts to concrete problems.

Application-Oriented Activities

Designing application-oriented activities that support learning objectives is challenging, but there is extensive research [Michaelsen and Black] [Duch and Allen] [Duch et al] [Delisle] [Paulson] to support the instructor developing his/her first team learning course. Common recommendations include

- Must require the groups to produce a tangible output.
- Should give students the opportunity to practice dealing with the same kinds of issues and situations they will encounter in future jobs
- Must be impossible to complete unless students understand course concepts.
- Should allow groups to spend the majority of their time doing what groups do well, e.g., identifying problems, formulating strategies, making decisions.

- Must be difficult enough so that very few students can complete the assignment working alone.

Student Evaluation

The Team Learning instructor has individual quizzes, group quizzes, group activity deliverables and peer evaluations to use in the construction of a final grade for each student. The weights assigned to each type of grade should reflect the instructor's preferences along with his/her assessment of the students' independence, honesty and experience with team projects. If an instructor chooses, he/she can add individual mid-term and/or final exams to the course requirements.

PERSONAL EXPERIENCES & OBSERVATIONS

Our use of Team Learning began slowly after attending a workshop. Over time, we built our collection of activities and our experience with classroom management. As of this writing, we have taught four programming classes from start to finish using Michaelsen's Instructional Activity Sequence as the regular mode of class operation.

We found it necessary to adhere carefully to the Instructional Activity Sequence to keep the class moving forward. We spent about one third of class time on Readiness Assurance and two thirds on Application-Oriented Activities. On average, students spent about one hour on out-of-class work for every hour of in-class work, and they reported a strong need to attend all classes.

Research [Fiechtner and Davis] suggests that instructor-formed groups are preferable to groups chosen by the students themselves. We agree, even though students argue that getting together will be easier if they are teamed with someone with whom they work or who lives in their dorm. Part of the first class must be devoted to team

building [Cooper and Mueck]. We used a team contract: a list of norms, developed by each group, that articulates the performance and behavior expectations. Most conflicts that arose later in the term were caused by someone or everyone breaking the team contract. Conflict resolution was easier in the face of a signed statement declaring the behavior in question to be unacceptable.

Although it is inconvenient for announcements and instructor feedback, We believe it is necessary to hold the class in a room equipped with a computer for each student. That being said, we find ourselves often asking students to leave the keyboard to think or to discuss a pertinent issue or to make certain of the course of action before writing code.

Students perceived that they worked harder in these classes than in others, so motivation was always on our minds. We tried to make the demands of the entry-level job market as real as possible. Several times during the semester, we reviewed sample questions from various professional certification examinations, and we discussed technical job interviews often. We found it helpful to let students who have been "teched out" in applying for a job share the experience with the class. This knowledge of the market helped students to see the workload as a reasonable means to the end they desired.

Advantages of Team Learning

In the four courses we' ve taught with Team Learning, we' ve seen the best student work ever, and we' ve had the best interactions with students ever. Students asked deep questions. Since they were always working on a significant problem, their questions were focused and highly motivated. More work was done, and the classroom was more energetic and professional.

Initially, we feared spending time on higher level issues, such as evaluating a request to change our data model, in a programming class. We believed that the students' knowledge of objects and structures would suffer. Those fears were unfounded. Questions at the higher levels force the lower level work to get done. We believe the Readiness Assurance Process is responsible for this. This is a solid, workable methodology that does not let the lower level material slip away, despite the fact that it spends quite a bit of time at the higher levels of the Bloom taxonomy.

Individual student achievement, as measured by individual quiz grades, was about the same; certainly no worse than earlier lecture-only classes. We are motivated to continue because there seems to be no upper limit on what the best students can do in a Team Learning class. The opportunity to be actively engaged so often during the class provides the means for these students to exercise their gifts at a very high level. Their response, in turn, energizes us, and we truly look forward to reading their work and meeting with them in class.

Disadvantages of Team Learning

Developing objectives, quizzes and activities takes a lot of time, more time than preparing lectures. There is also a lot of grading to do, although less than expected because a large part of the grading is group submissions rather than individual work. The quick turnaround can also be difficult. We found we had to reserve the same periods each week for grading in order to keep the turnaround constant and not fall behind.

Managing the groups demands diligence, resourcefulness and experience on the part of the instructor. We did not have much experience with groups when we began

using Team Learning. We read the literature, but more importantly, we talked with my colleagues who did have the experience. They provided valuable assistance.

The most difficult problem to address involves the groups in which one good student is doing most, if not all, of the work on the activities. We believe early detection is the best defense. Diligent follow-up in the form of observation and questioning of each student in the group by the instructor can put the group back on track. This extra attention does not have to be punitive. Most students want to do well, so a straightforward statement of the perceived problem and desired outcome creates a workable context for the students' response.

One of the principles of Team Learning is that good students benefit from teaching poorer students. We believe this to be true because we learn something new every time we teach any course, but some good students complained that working with poorer students prevented them from getting all they could out of the course. We found the complaint to be valid if the difference in ability was great. We try to counteract the problem by limiting the difference between the best and worst students in each group.

CONCLUSION

Although we did not do a controlled experiment with data analysis, we are confident that the grades coming out of our Team Learning classes did not justify the effort to develop such a delivery. The class atmosphere, however, did. Students were more actively engaged and more articulate as a result of their Application-Oriented Activities. The improved work from the best students and the improved and more frequent interactions

with all students lead us to conclude that Team Learning is preferable to a lecture-only format.

In the classes discussed in this paper, we had a Readiness Assurance quiz every week, and two queries similar to the Unix course example were due from the groups each week. As stated above, only the best students were able to cope with this demanding environment. Mid-range students and poor students were eventually worn out by the pace. In the future, we plan to cut that schedule back to three team exercises each term and to return to a lecture format with traditional quizzes, i.e. given after the corresponding lectures, for the remaining time of the course. We hope that this strategy will continue to engage our top students, and, perhaps, allow less gifted students to do more than they would in a lecture-only offering.

Bibliography

Astin, A. (1985), Achieving Educational Excellence, San Francisco: Jossey-Bass.

Bloom, B.S. (1956), Taxonomy of Educational Objectives: The Classification of Educational Goals, New York: David McKay.

Cooper, J. and R. Mueck (1990), "Student Involvement in Learning: Cooperative Learning and College Instruction," Journal on Excellence in College Teaching, 1, 68-76.

Cooper, J., S. Prescott, L. Cook, L. Smith, R. Mueck and J. Cuseo (1990), Cooperative Learning and College Instruction: Effective Use of Student Learning Teams, Long Beach, CA: The California State University Foundation.

Dansereau, D. (1983), Cooperative Learning: Impact on Acquisition of Knowledge and Skills, Abilene, TX: U.S. Army Research Institute for the Behavioral and Social Sciences.

Deitel, H., P. Deitel and T. Nieto (2001), e-Business & e-Commerce - How To Program, Upper Saddle River, NJ: Prentice Hall.

Delisle, R. (1997), How To Use Problem-Based Learning in the Classroom, Association for Supervision and Curriculum Development.

Duch, B. and D. Allen (1996), "Problems: A Key Factor in PBL", About Teaching, 50 (Spring), 25-28.

Duch, B., S. Groh and D. Allen (2001), The Power of Problem-Based Learning: A Practical ' How To' for Teaching Undergraduate Courses in Any Discipline, Sterling, VA: Stylus Publications, LLC.

Fellers, J.W. (1996), "Teaching Teamwork: Exploring the Use of Cooperative Learning Teams in Information Systems Education," DATA BASE, Vol. 27, No. 2, pp. 44-60.

Fiechtner, S. and E. Davis (1985), "Why Groups Fail: A Survey of Student Experiences with Learning Groups," The Organizational Behavior Teaching Review, 9 (4), 58-73.

Johnson, D., R. Johnson and K. Smith (1991), Cooperative Learning: Increasing College Faculty Productivity, ASHE-ERIC Higher Education Report No. 4, Washington, DC: The George Washington University, School of Education and Human Development.

Kulik, J. and C. Kulik (1979), "College Teaching," in Research on Teaching: Concepts, Findings and Implications, Berkeley, CA: McCutcheon.

Lieux, E. (1996), "Comparison Study of Learning in Lecture vs. Problem-Based Format," About Teaching, 50 (Spring), 18-19.

McKeachie, W. (1988), "Teaching Thinking," NCRIPTAL Update, 2 (1), 1.

Michaelsen, L. and R. Black (1994), "Building Learning Teams: The Key to Harnessing the Power of Small Groups in Higher Education," Collaborative Learning: A Sourcebook for Higher Education, 2, State College, PA: National Center for Teaching, Learning and Assessment.

Paulson, D. (2001), http://curriculum.calstatela.edu/faculty/dpaulso/active

Ryan, S., Bordoloi, B. and Harrison, D. (2000), "Acquiring conceptual data modeling skills: The effect of cooperative learning and self-efficacy on learning outcomes," Database for Advances in Information Systems, http://129.62.162.209/~dbase/ab.html: ACM-SIGMIS.

Slavin, R.E. (1990), "Research in Cooperative Learning: Consensus and Controversy," Educational Leadership, 47 (4), 52-55.

Smith, D. (1977), "College Classroom Interactions and Critical Thinking," Journal of Educational Psychology, 69, 180-190.

Smith, D. (1980), "Instructions and Outcomes in an Undergraduate Setting," Annual Meeting of the American Educational Research Association, Boston.

Tinto, V. (1975), "Dropout from Higher Education: A Theoretical Synthesis of Recent Research," Review of Educational Research, 45 (1), 89-125.

Treisman, P. (1985), "A Study of the Mathematics Performance of Black Students at the University of California, Berkeley," Dissertation Abstracts International, 47, 1641-A, Doctoral Dissertation, University of California, Berkeley.

Wales, C and R. Sager (1978), The Guided Design Approach, Englewood Cliffs, NJ: Educational Technology Publications.