COLLABORATIVE TEACHING USING CONSTRUCTIONISM IN COMPUTER INFORMATION SYSTEM COURSES

Dr. Bonita M. McVey, St. Norbert College, bonnie.mcvey@snc.edu
Dr. Kathleen K. Molnar, St. Norbert College, kathy.molnar@snc.edu

ABSTRACT

Using examples from CS and CIS capstone courses, realized benefits of collaborative teaching using constructionism are outlined. We believe that the lessons we have learned in the process are transferable across academic disciplines

Keywords: constructionism, collaboration, teaching, CS/CIS

INTRODUCTION

If we assume the goal of an educational institution is to make students self-educating people, then we need to develop in students the ability to synthesize knowledge from various sources and to train students in methods of scholarly inquiry. "Educators face a challenging situation. Merely helping students become technically qualified to recall, recite, and apply predefined classroom routines to predefined classroom problems is not adequate preparation for the business environment" (15). The traditional method of instruction focuses on individuals and individual courses as if they are not connected. However, computer information systems (CIS) are not created in a vacuum and most CIS problems are solved in a team environment. We cannot simply teach students CIS concepts in isolation devoid of interrelated concepts and theories. We must ensure that our students will be able to see the interrelationships between functional area courses and be able to synthesize and apply this knowledge to form an integrated organizational perspective that goes beyond the individual and individual course. This paper suggests that one way to achieve these objectives is by employing a constructionistic approach to teaching CIS by using informal collaborative methods. We believe that the lessons we have learned in the process are transferable across academic disciplines.

Background and Literature Review

What is a constructionistic approach to teaching CS/CIS and what does this have to do with collaborative teaching? More specifically, how does requiring students to go outside the traditional course instructor-student interaction relate to CS/CIS? First, it is necessary to operationally define what is meant by a constructionistic approach and how collaborative teaching enhances this process. Then, we will outline two examples of how this strategy is used in CS/CIS courses and tie these examples back to our objectives.

Constructionism asserts that learning is an active process, in which people actively construct knowledge from their experiences in the world. Add to this the idea that people construct new knowledge with particular effectiveness when they are engaged in constructing personally meaningful artifacts. Though constructionism shares certain ideas with 'hands-on' approaches to education, it goes beyond hands-on in several important ways. In many hands-on activities, students simply follow a 'recipe' of what to do, such as creating a required computer program. Students are limited in how far they can improvise and explore. A constructionist alternative not only expands

the possible range of explorations, it also makes those explorations more personally meaningful. One of best ways to understand something is to create it, construct it, build it (13).

There are various terms regarding the shared responsibility and delivery on instruction in classrooms, many of which have been used synonymously such as team teaching, co-teaching, cooperative teaching and collaborative teaching (14). Team teaching is characterized as taking turns in leading a discussion or having the two teachers play roles in a demonstration (14). Cook and Friend (5) define four key components of co-teaching: two educators, delivery of meaningful instruction, diverse group of students, and common settings. Bauwens and Hourcade (3) define cooperative teaching as "…two or more educators possessing distinct skills working in a co-active and coordinated fashion to jointly teach academically… in educationally integrated settings…" For the purposes of this paper, we define collaborative teaching in a much broader sense by incorporating portions common to all of these. That is, collaborative teaching is two or more educators, working together in a joint intellectual effort, delivering meaningful instruction to groups of students in an academic environment. Most team teaching approaches are often teachers in the same subject (16); the collaborative approach allows for teachers representing different disciplines. Further, the collaborative approach does not mean that both educators must be together in the same classroom, or even teach the same group of students or even the same project. In this case, collaborative teaching simply means multiple teachers directing class projects between separate classes.

Collaborative teaching in higher education (especially across disciplines and between academic institutions) is the exception (2) even though research has demonstrated this as a best practice for improving student outcomes (10). This is mainly due to the problems encountered in team teaching such as the use of multiple faculty resources to teach one class, conflicts in teaching style, course presentation and grading, and academic time and energy constraints (7). In both collaborative teaching experiments, we taught and ran our specific classes separately. This alleviated the problems associated with team teaching, while at the same time allowing for interdisciplinary integration of material.

So how does collaborative teaching help us achieve our objectives? Bloom's taxonomy (4) is a useful starting place for discussing learning objectives. Bloom maintains that learning occurs in a hierarchy of increasing cognitive complexity. This hierarchy progresses from the simplest form in which students passively receive knowledge to comprehension, application, analysis, synthesis and ultimately, evaluation. As this transition occurs, the students must become increasingly active in the learning process. Collaborative teaching can help students to better 'understand' the CIS/CS process through experiential manipulation. To understand the process of creating CIS/CS projects, you need a basic knowledge of theory, programming, and systems analysis and design. But there is more to CIS/CS projects than this.

In many CIS/CS projects, students solve problems that exist in a simulated environment, either a structured program or computer simulation model. Further, even if students work on an actual CIS/CS project, they are usually confined to a specific functional area of expertise. But instead of solving problems in a static environment, what if students could construct actual systems for real organizations and devices, and what if students solved these problems on their own asking for expertise from multiple instructors and not just the course instructor? These activities could

fundamentally change how students think about CIS/CS and learning. This is where constructionism and collaborative teaching become intertwined. Students must 'create it, construct it and build it' in collaboration with their fellow students and teachers. In our collaborative teaching approaches, we created learning environments where students were not only encouraged, but also required to seek expertise from instructors outside the course. As suggested by Langenberg (9), the ultimate responsibility for effectively delivering all forms of learning is shifted from the individual faculty member to the department.

PROJECT OVERVIEW AND METHODOLOGY

The main objective of collaborative teaching in CIS/CS courses is to develop teaching methods and approaches that are more meaningful and interesting for the student. The teachers' role in these projects changed from an instructor to the role of a facilitator of learning and coordinator of learning environments. The first courses taken by CIS/CS students allow the instructors to introduce and develop the theory of their particular disciplines including basic programming and theory (incorporating passive knowledge and comprehension cognitive processes). The second and third year courses taken include team projects using software engineering, database and system analysis and design approaches (incorporating application and analysis cognitive processes). The final portion of the courses taken includes a capstone course. For the CS student, the course involves an individual project that integrates several areas within computer science. For the CIS student, this is a systems projects class in which the students must complete a real world project integrating the computer science concepts with the different business functional areas of an organization (incorporating synthesis and evaluation cognitive processes). The aim of these courses is to allow students to apply the knowledge and skills they have acquired and to extend their knowledge. The goals of course projects are to "prepare students for the working life, making them familiar with the work place by practicing their skills on real-world problems"(11) as well as to "provide students with an insight and a flavour of research methodology that should be useful for those students who continue to study."(11) Jarvinen (8) states that one of the most important things in education is to adjust the teaching methods to the nature of the content.

Our first example of collaborative teaching using constructionism in CIS/CS courses occurs in the CS senior capstone experience course in which students are assigned individual projects. These projects are designed to help the student integrate concepts from a variety of areas within CIS/CS as well as to engage the student in research to extend their knowledge. At the beginning of the semester, students are given a project description and requirements. The first requirement is to research methods to solve the problem and then to determine all necessary hardware to implement its solution. When necessary, students are expected to acquire any additional skills to complete their projects. All CIS/CS faculty collaborate with the students, serving as resources for information and feedback. In addition, students often collaborate among themselves since their projects may be related through a larger problem. Two such projects were to develop software systems for the CCR (Computer Controlled Railroad). One student was to develop an operating system for the CCR that prevented collisions yet also provided for free movement of the trains. A second student was to develop a system by which the actual speed of the train could be calculated and to detect and identify a "lost" train. As the semester progressed, the student designing the operating system for the CCR wanted confirmation of his object-oriented design and use of objects. Since the instructor of the course specialized in architecture and operating systems, the student sought assistance from

another faculty member within CIS/CS. He presented his design, was given feedback and made some changes. Together, the instructors and the student verified that changes to the underlying structures would maintain the integrity of the operating system. Much later in the semester, both students developing software systems for the CCR discovered they had a common problem. Without building a human interface, it would be difficult for anyone not keenly involved in the projects to believe that the occurrences on the train track were not staged and that allocation of resources and speed calculations were being performed in real-time. The students clearly wanted others to understand what they had designed and constructed, but time was becoming a crucial factor. Fortunately, these two students were taking another course in which they were learning how to design and implement graphical user interfaces. The instructors of the capstone course and GUI/HCI course met with the students and agreed that the development of a human interface for the train project was appropriate and that because of its numerous requirements, could substitute for the final two projects in the GUI/HCI course. The requirements for the CCR's interface were substantial. Ease of use was paramount as was efficient redrawing of the images and updating of the controls. The ability to resize and maintain aspect ratios was essential because the test and production displays were very different. Finally, the interface needed to be fully documented for extensions would inevitably be added. In short, the development of the interface provided an excellent opportunity to integrate software engineering concepts. Using the instructor of the GUI/HCI course as a resource, these students together developed an interface that that not only verified their knowledge of the concepts in the GUI/HCI course but also extended them. During their senior capstone presentations to faculty and students, the interface that they created together enabled those present to understand how their software systems solved the problems that they were originally assigned. As a result, many of those present actively participated in the Q&A portion of the presentations, discussing choices made during design of the systems as well as possible extensions and improvements. Through this unusual collaboration among the faculty and the students, the students pulled together concepts from different areas of CIS/CS to produce a software system that met the requirements of their capstone projects, met the requirements of their GUI/HCI course assignments, and provided other users the ability to control the CCR and gain feedback. Members of the CIS/CS faculty were stimulated by the success of this interaction for it provided not only an opportunity for students to share ideas and grow in knowledge but also provided similar opportunities for the faculty. As all learned a great deal from this unusual but valuable collaboration, faculty now actively seek to provide similar opportunities for our students.

The second example of using constructionism in collaborative teaching is the CIS student's capstone course, systems projects. In this course the students are divided into groups of 3-4 students per team. The teams are then given a real-world computer information systems project to complete by the end of the 15-week semester. It is completely up to the students how the task is to be completed and the workload allocated. The students must complete all systems analysis and design steps, determine the appropriate programming language, information technology, database design (if applicable), interface and input and outputs necessary to complete the project on time and to the client's satisfaction. The students must acquire the necessary skills needed to complete the task. In most cases, the students may not even be familiar with the programming language or information architecture required to complete the task, so they must also be able to acquire the necessary knowledge and skills to complete the project. The students are also required to approach members of the faculty to validate their design and help them acquire what is necessary. For example, students who require a database in their projects must present the ERD and database design to the

instructor who teaches the database course for critique and suggestions. Students must also have their coding checked with the instructors who teach software engineering. These instructors are not the same people who teach the systems projects course. In addition, the students must present their projects to the entire CIS committee as well as the client at the completion of the project. These experiences have proved thought provoking and enlightening to the entire group of instructors who teach CIS courses. In many instances, the IS faculty disagrees with the CS faculty particularly in terms of design and process. These disagreements have led to very lively discussions involving not only the faculty but also, the students. The students must not only resolve the issues, but they must also justify their solutions to the faculty involved. These disagreements have given rise to one of the most powerful learning environments that our students have encountered. The students see that there is more than one way to solve a problem, that there will be disagreements over which way is best and that they must decide which course of action to take. There is no 'cook-book' answer to their problems. The students must learn to construct new knowledge from these encounters and synthesize and apply this knowledge to form an integrated organizational perspective to solving problems.

SUMMARY AND CONCLUSION

Papert (12) states that there are two views of education,

> "In one, the goal of education is to foster individual development. The other focuses on the information that the individual will acquire…What you ought to be learning in school is that you don't need to be taught in order to learn. This is the fundamental cleavage between theories of education: empowerment of the individual versus instruction and being taught... Do we want empowered individuals who will feel the power to make their own decisions and to shape their lives? Or do we prefer citizens who will accept the discipline of following the instructions and the programs that are set up for them by others? Knowledge is not transmitted it is constructed."

Perhaps the most important objective was to encourage applications of student learning in experiential collaborative settings involving integration of computer science and business concepts with IS technology. An ancillary benefit to students involved enhancing students' personal skills and self-confidence by working on self-directed team projects. As a result of these projects, students better understand organizational concepts, information system integration possibilities, teamwork and the difficulty of managing projects. Further, the experience is helpful to the students in job-hunting after graduation. For ourselves, the synergy of collaborative teaching has opened to door to improving our teaching. Collaboration also reduces isolation and allows for another perspective on course design and pedagogy. But most importantly we were able to meet our mutual educational goals with interdisciplinary and interacademic approaches to teaching. In the final analysis we found that all these goals were indeed achieved.

Student feedback indicated that these learning exercises succeeded in achieving the following learning outcomes for the students:

- Engaging the students in discussions of multiple and different disciplinary perspectives broaden students understanding of knowledge;
- Providing students with the opportunity to experience working in and managing groups;
- Providing students the opportunity to see themselves and their peers as constructors of new

knowledge;
- Providing students with the opportunity to offer their opinions and have a greater connection to the faculty;
- Providing students with real-life experience in creating new knowledge;
- Brainstorming a real business problem using information technology;
- Using their ideas to exploit the available technology to gain competitive advantage;
- Understanding the limitations of technology in implementing creative ideas;
- Enhancing student's personal skills and self-confidence.

Faculty feedback indicated that these learning exercises succeeded in achieving the following scholarship outcomes for the faculty:
- Providing faculty with a sense of community in having their opinions valued and being a part of the final project;
- Sharing of roles, attitudes and expertise especially in regards to curriculum and instruction, fostered a greater synergy within the department;
- Assisting the students at all levels increased the professional development of the instructors;
- Hearing and respecting each others differences of opinions provided a valid learning opportunity to the students and faculty;
- Increasing the rapport among students and instructors, instructors and instructors, students and students;
- Increasing professional development by learning from each other about respective fields and pedagogical techniques;
- Supporting each other in teaching experiments and in conducting classroom research;
- Fostering pedagogy of team teaching and learning approaches;
- Helping in program assessment and development in both CS and CIS programs.

According to Eisen and Tisdell (6), "Our job is ultimately to enable students to integrate new information from a variety of disciplines so they can become ongoing constructors of new knowledge, both on an individual level and with others in a social context." We need to empower our students and collaborative teaching using a constructionistic approach is a pedagogical way to teach our students to teach themselves.

> "The challenge for teachers lies in making the scholarship more relevant to students so that they may approach their chosen profession with enhanced understanding and a greater capacity for imagination and creativity" (1).

REFERENCES

1. Adams, E. and Pugh, D. (1994). The Humanities and Professional Studies, *College Teaching*, **42** (2), 63-65.
2. Bakken, L., Clark, F. and Thompson, J. (1998). Collaborative Teaching, *College Teaching*, **46** (4), 154-157.
3. Bauwens, J. and Hourcade, J. (1995). *Cooperative Teaching: Rebuilding the Schoolhouse for all Students*. Austin, TX: PRO-ED.

4. Bloom, B., Hastings, J. and Madous, G. (1971). *Handbook on Formative and Summative Evaluation of Student Learning*. New York: McGraw-Hill.
5. Cook, L. and Friend, M. (1996). Co-Teaching: GUI/HCIdelines for Creating Effective Practices in E. L. Meyers, G. A. Vergason, and R. J. Whelan (Eds.), *Strategies for Teaching Exceptional Children in Inclusive Settings*, Denver: Love, 155-182.
6. Eisen, M. J. and Tisdell, E. J. (2003). Team Teaching: The Learning Side of the Teaching-Learning Equation, *Teaching Excellence*, **14** (6), 1-2.
7. Forcey, L. and Rainforth, B. (1998). Team Teaching "Conflict Resolution in Educational and Community Settings": An Experiment in Collaboration and Conflict Resolution, *Peace and Change*, **23** (3), 373-385.
8. Jarvinen, Esa-Matti (1998). The Lego/Logo Learning Environment in Technology Education: An Experiment in Finnish Context, *Journal of Technology Education*, **9** (2), 1-14.
9. Langenberg, D. N. (1992). Team Scholarship Could Help Strengthen Scholarly Traditions, *The Chronicle of Higher Education*, 2 September, A64.
10. Murata, Roberta (2002). What Does Team Teaching Mean? A Case Study of Interdisciplinary Teaming, *The Journal of Educational Research*, **96** (2), 67-77.
11. Olsson, B., Berndtsson, M., Lundell, B., and Hansson, J. (2003). Running Research-Oriented Final Year Projects for CS and IS Students, *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, 79-83.
12. Papert, Seymour (1990). A Critique of Technocentrism in Thinking About the School of the Future, *Epistemology and Learning Group, Massachusetts Institute of Technology Media Laboratory,* E&L Memo No. 2, 1-10.
13. Resnick, Mitchel (1994). Learning About Life, *Artifical Life*, **1** (1-2), 1-10.
14. Welch, M., Brownell, K. and Sheridan, S. (1999). What's the Score and Game Plan on Teaming in Schools? A Review of the Literature on Team Teaching and School-Based Problem-Solving Teams, *Remedial & Special Education*, **20** (1), 36-49.
15. Wenger, M. (1999). Team Teaching for Higher Level Learning: A Framework of Professional Collaboration, *Journal of Management Education*, **23** (3), 311-328.
16. Winn, J. A. and Messenheimer-Young, T. (1995). Team Teaching at the University Level: What We Have Learned, *Teacher Education and Special Education*, **18** (4), 223-229.