Decision Sciences Journal of Innovative Education
Volume 4 Number 2
July 2006
Printed in the U.S.A.

TEACHING BRIEF

Peer-to-Peer Instruction in a Programming Course

T. Grandon Gill†
Information System and Decision Sciences Department, College of Business Administration, University of South Florida, 4202 East Fowler Avenue, CIS1040, Tampa, FL 33620, e-mail: ggill@coba.usf.edu

Qing Hu
Decision and Information Systems Department, College of Business, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, e-mail: qhu@fau.edu

INTRODUCTION

The way programming is taught—individual students creating small, self-contained programs—and the way programming is normally accomplished—teams integrating code into complex applications, with emphasis on reuse—differ significantly (Prey, 1995). Compounding the problem, what employers want from management information systems (MIS) graduates appears to be still different. As an example, each of us participated in a different series of focus groups, consisting of dozens of information technology (IT) managers and recruiters, seeking to identify what outcomes were desired from an MIS undergraduate program. The results were strikingly similar, even though separated by nearly a decade (1992 to 2002). Employers demanded fundamental skills—communications, ability to work effectively in teams, general problem solving—skills more general than technological.

The disconnect between curricula, practice, and job skills, summarized in Table 1, presents MIS educators with a serious dilemma. On the one hand, an MIS program is constrained to offer a substantial amount of technical content. On the other, such a program advertises itself as being in tune with the workplace. One possible way to reconcile these diverse agendas is through a course with rigorous technical content employing a pedagogy based on encouraging peer-to-peer learning, a well-accepted principle of undergraduate education (Chickering & Gamson, 1987). This brief summarizes how such peer-to-peer learning has been incorporated into introductory programming courses in two separate undergraduate MIS programs.

COURSE DESIGN

The courses described here are introductory programming courses, required as part of an MIS major at two large state universities. Four distinct design elements were

†Corresponding author.

Table 1: Programming instruction, practice, and employer needs.

Programming Instruction	Programming Practice	Employer Needs (As Reported in Focus Groups)
1. Done alone	1. Done in groups	• Communication skills
2. No reuse of code	2. Code is reused	• Teamwork skills
3. Stand-alone programs	3. Work on pieces of programs	• Problem-solving skills
4. Small programs	4. Large applications	• Able to learn on the job
5. Taught through lecture	5. Learned through samples	

employed, with slight variations between the two sites, to foster both communications and teamwork: (1) use of a fixed grading scale based entirely on assignment performance, (2) encouraging the formation of teams, (3) use of asynchronous discussion groups to provide support, and (4) use of peer teaching assistants (TAs). Each is now described.

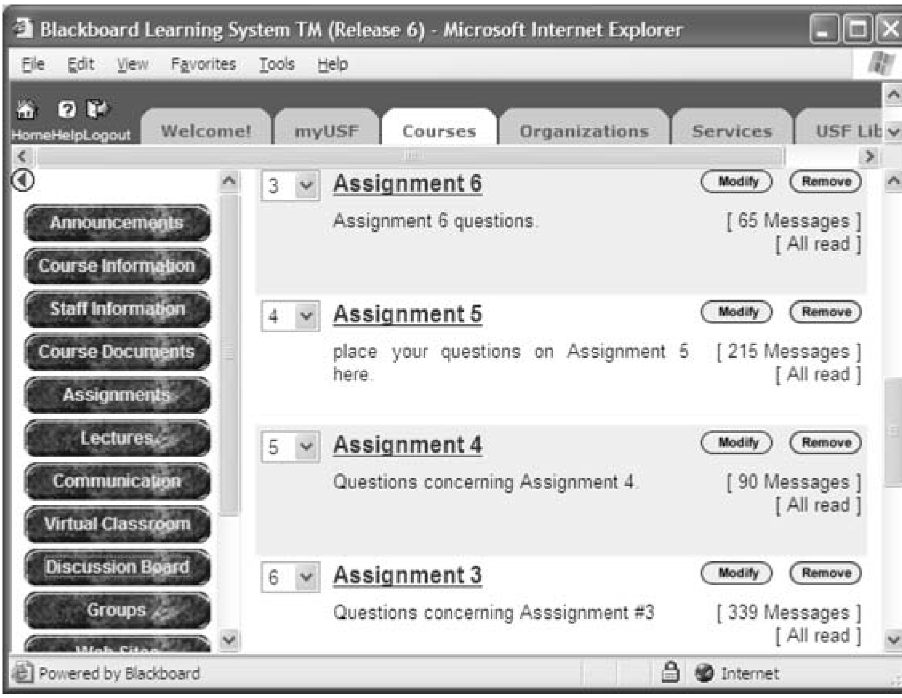
Assignment Focus

If teamwork and peer cooperation are to take place, students must perceive that any contributions they make to the learning of the class as a whole will not work against their own best interests. One way to accomplish this is through a fixed grading policy—specified on the first day of class—that is based entirely on assignment completion. While attractive in principle, such an approach does have an obvious flaw: the efficiency of today's electronic communications could reasonably be expected to produce near-instantaneous distribution of an assignment throughout the entire class as soon as one student completed it. Ironically, the problem is exacerbated when the spirit of cooperation has been instilled.

Individual validation exams for each assignment serve as a remedy for such nonproductive cooperation. These exams test the student's understanding of the assignment that he or she handed in. Although online exams are used for some assignments at one site, the largest assignments are validated by oral exams at both sites. In these exams, administered individually, the instructor or TA asks the student questions about the assignment. The objective is to determine if the student *understands* what was handed in. The student found wanting in such understanding remains in a "not yet passed" status—meaning that no credit is given for the assignment. The exam may be retaken as many times as required within a specified time period.

The original intent of the oral exam process was limited to ensuring that rigor in the course was not lost as conventional tests were discontinued. Two other benefits of administering these exams quickly emerged, however. First, they provided students with practice at being effective communicators. Indeed, countless students have commented—to both instructors—that such exams provide excellent training for job interviews. Second, the exams have served to motivate peer learning. A common sight at both locations is clusters of students gathering to brief each other on the assignments handed in by the team and to tutor teammates retaking oral exams.

Figure 1: Assignment discussion groups.



Encouraging Teamwork and Peer-to-Peer Learning

Team formation is encouraged, but not required, at both sites. At one site, students may submit an assignment as a group, receiving precisely the same credit as those submitting individually. At the other site, each student must submit the assignment individually, but may prepare the work in groups. Teams are also allowed to cross-pollinate—all made possible by the rigor introduced by the assignment validation process. At the site where group project submission (as well as preparation) is allowed, it is estimated that roughly 80% of all assignments are submitted as part of a team.

Use of Asynchronous Discussion Groups

Asynchronous discussion groups, hosted on the Blackboard, are central to the peer-oriented approach at both sites (see Figure 1). They provide an environment where students can share information about the assignments. They also replicate the approach to technical support used by many vendors, making them realistic approximations of commercial settings.

As with group formation, use of the assignment discussion groups is entirely voluntary. The assignment discussion groups are highly active, with traffic for a single assignment sometimes exceeding 200, or even 300, posts at one site, where traffic estimates suggest that an average student accesses the board every other day. Students are encouraged to reply to questions posed by their peers and do so often. Sometimes they even post their own learning aids. For example, a student at one site posted a spreadsheet that greatly clarified a particular exercise—and the assignment completion rate more than doubled from the previous semester.

Undergraduate TAs

The final element of course design stimulating peer-to-peer interaction is the use of undergraduate TAs. Although the success of this approach had been previously reported in the computer science literature (e.g., Roberts, Lilly, & Rollins, 1995; Reges, 2003), MIS courses had traditionally used graduate students for such roles. There were, however, a number of advantages to using undergraduates who had recently completed the course. Recent students understood the course procedures and required minimal training—especially because students interested in becoming TAs had to take all of their oral exams with the instructor, providing the opportunity to assess their knowledge, communications skills, and motivation. As peers to students taking the course—more often than not, a TA was enrolled in one or more other classes with any given student taking the course—they were less intimidating than graduate students (or instructors for that matter). The opportunity to become a TA could also serve as a motivator for achievement-oriented students enrolled in the class. It was often the case that the students who served as TAs found jobs more rapidly than the others upon graduation, largely because they had been TAs for a rigorous programming class. Finally, undergraduates were 20–30% less expensive than graduate students, allowing the total hours of assistant time to be expanded.

The duties of TAs included: (1) running weekly structured lab sessions, as well as covering office hours and open labs; (2) replaying taped lectures for sections where the instructor did not lecture live (at one site); (3) grading assignments; (4) administering lab-based validation exams; (5) administering most first-try oral exams; and (6) monitoring the online discussion groups (along with the instructor), the goal being to keep response times as short as possible. For most students, TAs served as the primary point of personal contact throughout the course—because a relatively small percentage of students typically chose to visit during the instructor's office hours.

OUTCOMES

The pedagogy described has been implemented since spring 2003 at one site and fall 2003 at the other. Students at the earlier site were surveyed at the end of the course for all 2003 sections, with 60–70% electing to fill in a 300+ item instrument for extra credit. Selected results, illustrating the perceived effectiveness of the peer-oriented techniques employed, are presented in Table 2. The first group of items (value of 1 in column 1) shows a high degree of satisfaction with the assignment and validation process and a reluctance to change back to tests—in spite of the heavy time demands of the course relative to other courses. The second group shows a high degree of satisfaction with group processes and the perception that the course was helpful in learning to work effectively with others. The third group shows high levels of satisfaction with the Blackboard discussion groups and the perception that it contributed to learning. Finally, the fourth group shows their high level of

Table 2: Measures of perceived peer-to-peer effectiveness in site 1 course.

Goal	Item	Scale (Count)	Mean
1	The oral exam on Assignment 3 provided a fair assessment of my knowledge at the time	A (110)	4.4
1	The oral exam on Assignment 5 provided a fair assessment of my knowledge at the time	A (92)	4.1
1	The oral exam on Assignments 6 and 7 provided a fair assessment of my knowledge at the time	A (62)	4.0
1	Time per week spent on the class	T (110)	17.3
1	Time spent per week on other MIS classes	T (105)	7.8
1	Time spent per week on other business classes	T (106)	5.4
1	Time spent per week on other classes in the university, outside the College of Business	T (89)	4.2
1	The course should put greater emphasis on tests and less on assignments	A (112)	1.9
1	The grading system we used was helpful to learning	H (112)	4.2
2	Satisfaction with group work (collaboration) between students	S (103)	4.2
2	Teamwork in labs was helpful to learning	H (86)	4.2
2	Working with peers outside of class was helpful to learning	H (101)	4.1
2	How much has this class added to your skills in working effectively with others?	G (104)	3.8
3	Satisfaction with Blackboard	S (113)	4.6
3	Blackboard discussion groups were helpful to learning	H (97)	3.6
4	Median score on TA evaluations	S (N/A)	4.1
4	Satisfaction with TA availability	S (110)	4.4
4	TAs were helpful to learning	H (106)	4.1

Scale A (Agreement)
1-Strongly disagree, 2-Mildly disagree, 3-Neutral, 4-Mildly agree, 5-Strongly agree

Scale T (Time)
Estimated hours per week

Scale S (Satisfaction)
1-Very dissatisfied, 2-Somewhat dissatisfied, 3-Neutral, 4-Somewhat satisfied, 5-Very satisfied

Scale H (Helpful)
1-No help, 2-A little help, 3-Moderate help, 4-Much help, 5-Very much help

Scale G (Gain)
1-Not at all, 2-A little, 3-Somewhat, 4-A lot, 5-A great deal

satisfaction with the TAs (all scored higher than the instructor) and the perception that TAs were instrumental in the learning process.

Referring back to Table 1, Table 3 identifies some of the ways that peer-centered techniques address inconsistencies between programming courses and commercial practice (and employer requirements). In interpreting the table, it is important to recognize that the course's peer focus represents an ingredient

Table 3: How peer-to-peer techniques address programming practice and employer needs.

Programming Practice	Employer Needs (As Reported in Focus Groups)
1. *Done in groups*: The vast majority of students worked in groups and satisfaction was high (see Table 2) 2. *Code is reused*: The peer-centered validation process, modeled after submarine qualification (Gill, 2005b), allowed materials to be reused without loss of rigor and also made it possible for students to share and discuss code without loss of rigor 3. *Work on pieces of programs*: Although this objective was addressed directly through assignment design (see Gill, 2005c) rather than the peer-centric aspects of the course, use of undergraduate peer TAs who were already familiar with the course processes and assignments (and therefore requiring minimal training and supervision) made incorporating complex, multipart assignments based on modifying sample code more practical, from an implementation standpoint 4. *Large applications*: Allowing students to work with each other without penalty and consult TAs freely allowed expectations to be raised. Thus, assignments were of substantially greater size (e.g., Gill, 2004; Gill, 2005a) than would normally be seen in an introductory course 5. *Learned through samples*: [See justification for (3) above]	• *Communication skills*: The oral exams used for validation purposes underscored the importance of both knowing the material and communicating it effectively. The heavy reliance on discussion boards placed a similar emphasis on developing effective written communications skills—because poorly framed questions or responses tended to go unheeded • *Teamwork skills*: With no penalty for group completion of assignments—and no instructor intervention where teams fell apart—students generally reported increases in their ability to work effectively with others (see Table 2) • *Problem-solving skills*: The strict assignment focus of the course, enabled by its reliance on peer-validations, meant that all student efforts were directed toward solving problems (i.e., assignments) • *Able to learn on the job*: The self-paced nature of the course—made possible by the widespread availability of peer support—forces students to establish their own deadlines and to seek out both course materials and the support they need. Reliance on asynchronous discussion groups for support also acquaints students with a technique widely used in IT for posing questions and getting support

that enables the overall self-paced, assignment-centric design (Gill, 2005c). Thus, while peerfocus obviously contributes directly to some outcomes (e.g., teamwork, communications skills), the role it plays in enabling other aspects of the design—such as the complexity of the assignments—is more indirect (though no less important).

At the second site, regular end-of-the-semester student evaluations were conducted and compared with those of a previous semester (where the same course was taught by the same instructor using a traditional test-based pedagogy). The results were consistent with the first site's survey findings. Student satisfaction scores increased along with the perceived amount learned. Interestingly, difficulty and effort-expended perceptions were virtually unchanged (see Table 4).

Table 4: Student perception of the traditional versus the new pedagogy in site 2 course.

Evaluation Question	New Design (Fall 2003—14 Responses)	Traditional Design (Spring 2003—12 + Responses)
Overall rating of the instructor (Mean: 1-Excellent, 5-Poor)	• Excellent (43%) • Very good (29%) • Good (21%) • Mean (*SD*): 1.71 (0.83)	• Excellent (23%) • Very good (46%) • Good (8%) • Mean (*SD*): 2.46 (1.39)
How difficult was this course for you? (Mean: 1-Very difficult, 5-Very easy)	• Very difficult (14%) • Somewhat difficult (71%) • About right (14%) • Mean (*SD*): 2.00 (0.55)	• Very difficult (42%) • Somewhat difficult (17%) • About right (33%) • Mean (*SD*): 2.08 (1.08)
How much effort have you put into this course? (Mean: 1-Exceptional amount, 5-Almost none)	• An exceptional amount (21%) • More than usual (64%) • About as much as usual (14%) • Mean (*SD*): 1.93 (0.62)	• An exceptional amount (33%) • More than usual (42%) • About as much as usual (25%) • Mean (*SD*): 1.92 (0.79)
How much do you think you have learned in this course? (Mean: 1-An exceptional amount, 5-Almost nothing)	• An exceptional amount (36%) • More than usual (29%) • About as much as usual (36%) • Mean (*SD*): 2.00 (0.88)	• An exceptional amount (0%) • More than usual (58%) • About as much as usual (25%) • Mean (*SD*): 2.58 (0.79)

The qualitative analysis of the techniques (augmented by the survey responses at both sites) suggests that the techniques being employed are serving to make the courses both more realistic and effective. We are currently working to identify and obtain more objective sources of confirmation—necessarily a long-term project, because assessing the true success of a business course virtually demands a measurement of its impact in the workplace. In the short term, however, we believe our pilot studies to be extremely promising and definitely warrant continuing to use and refine the design.

REFERENCES

Chickering, A. W., & Gamson, Z. F. (1987). Seven principles for good practice in undergraduate education. *AAHE Bulletin, 39*(7), 3–7.

Gill, T. G. (2004). Teaching flowcharting using FlowC. *Journal of Information Systems Education, 15*(1), 65–78.

Gill, T. G. (2005a). Engaging introductory programming students with CGI. *Decision Sciences Journal of Innovative Education, 3*(1), 177–181.

Gill, T. G. (2005b). Teaching C++ submarine style. *IEEE Transactions on Education, 48*(1), 150–156.

Gill, T. G. (2005c). Assignment-centric design: Testing the assignments not the lectures. *Decision Sciences Journal of Innovative Education, 3*(2), 339–346.

Prey, J. C. (1995). Cooperative learning in an undergraduate computer science curriculum. *IEEE Frontiers in Education 1995*, *Session 3c2*, 11–14.

Reges, S. (2003). Using undergraduates as teaching assistants at a state university. *SIGCSE'03,* February 19–23, 2003, Reno, NV, 103–107.

Roberts, E., Lilly, J., & Rollins, B. (1995). Using undergraduate teaching assistants in introductory programming courses: An update on the stanford experience. *SIGCSE '95*. 3/95, Nashville, TN, 48–52.

T. Grandon Gill is Associate Professor at the University of South Florida. His educational background includes an undergraduate degree in Applied Mathematics from Harvard College and MBA and Doctor of Business Administration in the Management of Information Systems degrees from Harvard Business School. His teaching areas have included programming, management of information systems, database design, the Internet, and case method research. He has received numerous teaching awards, including the Florida Atlantic University award for excellence in undergraduate teaching. His research interests include expert systems, organizational learning, and MIS education. He has published in numerous journals such as *MIS Quarterly*.

Qing Hu is Associate Professor in the Department of Information Technology & Operations Management in the College of Business at Florida Atlantic University. He received his PhD in Computer Information Systems from the University of Miami, Florida. He is also a Microsoft Certified Trainer and Microsoft Certified Solution Developer. He has been teaching college-level programming courses and conducting programming training for professional developers for more than a decade. His research interests include the economics of information technology, especially in the areas of the impact of information technology on organizational structure and performance. His work has been published in academic journals including *Information Systems Research, Journal of Management Information Systems, California Management Review, IEEE Transactions on Software Engineering,* and *Communications of the AIS*.