Computer Science Education
2001, Vol. 11, No. 3, pp. 247–260

0899-3408/01/1103-0247$16.00

An Empirical Evaluation of Using Constructive Classroom Activities to Teach Introductory Programming

Mark J. Van Gorp and Scott Grissom
Computer Science and Information Systems, Grand Valley State University, Allendale, MI, USA

ABSTRACT

Computer science teaching is often based upon the traditional lecture format. However, this methodology may not be the best way to help many students actively understand underlying concepts. This paper explores an alternative pedagogical approach that emphasizes constructive and collaborative learning in CS1 classrooms. After briefly discussing constructivism and providing examples of constructivist techniques in CS1, empirical research results are provided. These results arise from a study that compares different CS1 sections that utilized the techniques at varying frequencies. A positive correlation was found between frequency and mean final exam scores. However, no pair-wise differences between sections were statistically significant. These outcomes and others are discussed in addition to future research design implications.

INTRODUCTION

Teaching and learning in today's computer science classroom is generally based upon two fundamentally different theoretical models: objectivism and constructivism. Objectivists believe that learning often occurs when students listen to an instructor's explanation, engage in reinforced practice, and respond to external motivation (Fosnot, 1996; Skinner, 1953). Meanwhile, assessment of learning occurs by measuring observed and quantifiable behavioral outcomes on pre-defined tasks (Fosnot, 1996). Objectivism often manifests itself in teacher-centered and teacher-controlled classrooms.

Constructivists, on the other hand, focus more on the learning and experiences of the student. Learning requires the student to actively construct

Correspondence: Scott Grissom, Computer Science Department, Grand Valley State University, 1 Campus Dr, Allendale, MI 49401-9403, USA. Tel: +1-616-895-2088, Fax: +1-616-895-2106, E-mail: grissom@gvsu.edu

personal meaning and understanding while thinking about previous experiences and considering alternative perspectives held by others (Bednar, Cunningham, Duffy, & Perry, 1992). Assessment of that learning evaluates the effective functioning of a learner in a targeted discipline as well as the ability to defend and explain decisions through developed metacognitive skills (Bednar et al., 1992).

Although, constructivism became well-known as a learning theory around the late 1980s and early 1990s, aspects of it are not new as they rest upon the earlier work of Dewey, Piaget, and Vygotsky (Fosnot, 1996; Phillips, 1995; Gadanidis, 1994). Even though presenting an exact definition of constructivism is difficult and likely impossible (Phillips, 1995), constructivist classrooms are often viewed as problem-solving environments manifested through three C's: context, construction, and collaboration. First, students are to be given problems entrenched in authentic and perhaps simplified contexts. This will provide internal motivation for the student as opposed to external motivation that behavioral objectivists would emphasize. Second, students are to construct knowledge based upon meaningful activities: they cannot be given this knowledge. Lastly, students often collaborate with their peers. This aids the knowledge construction process, as students must seek to examine alternative perspectives on problem solutions and thus possibly re-construct their own perspectives and solutions.

CONSTRUCTIVISM IN COMPUTER SCIENCE EDUCATION LITERATURE

Based upon the problem solving nature of computer science, it is compelling to use constructivism in helping students learn the discipline. However, as Ben-Ari (1998) notes, and as we also concluded, there are relatively few empirical articles based upon constructivism in computer science education when compared to another field such as mathematics education. Truly, numerous studies such as Kim, Sharp, and Thompson (1998) have shown constructivist value in mathematics reform.

Further, other positive outcomes are noted when one considers the influence of constructivist elements such as collaboration and cooperation in the computer science education literature. For example, Ramsey, Rada, and Acquah (1994) write that collaborative learning for computer science students can be effective when a well-formulated methodology for working exists.

Keeler and Anson (1995) found that cooperative learning significantly enhanced learning performance and student retention in a computer literacy course. Additionally, authors such as Gorriz and Medina (2000) and McGrath (1990) note that girls in particular excel in those computing environments where learning is collaborative.

CREATING COLLABORATIVE AND CONSTRUCTIVE ENVIRONMENTS

When thinking about the value of constructive and collaborative learning environments in computer science, we decided to employ constructive group work in our CS1 classes. The goal is for students to actively think, collaborate, and construct problem solutions instead of remaining recipients of purely lectured information. Many of our activities seek to mimic what a computer scientist might do and include the following.

Code Walkthroughs

In this activity, students step through existing code and predict the output. This helps students practice and better understand flow of control. Responsibilities can be assigned based on the code that is being reviewed. For example, consider a class session on parameter passing that distinguishes between passing by value and passing by reference. We provide a number of methods that have four parameters. Some are passed by reference and some are passed by value. Each group member is responsible for tracing through the code for one parameter. This division of labor helps to keep everyone involved instead of the strongest student taking control of the group.

Writing Code

Another activity is to have groups write code to solve a small problem. For example, ''write pseudo code to simulate 500 coin tosses and print the number of heads.'' It is important that we walk around the room and observe the groups working together. We provide guidance when appropriate but prefer to remain quiet and let group members answer their own questions. Groups are motivated to write a complete and accurate solution since they may be called to share their solution with the rest of the class.

A variation of writing code during class is to assign one or two small problems for students to solve before coming to class. They bring their typed

solutions to class and break into their groups. Group members walk through each other's code to compare solutions and then collaborate on a group solution. Individual solutions as well as the group solution are then handed in. With this technique, all students must think about the problem on their own before coming to class and also understand the ways others have solved the problem.

Scaffolding

This educational concept recognizes that novices need additional support to solve a problem. The idea is simple but important to computer science pedagogy as learning language syntax, code flow, data representation, and appropriate design are daunting tasks for novices.

We use scaffolding in various ways. For example, we split students into groups of two and give them code that solves a given problem. The code in this example is the scaffolding that students will build upon. We then have them insert comments to describe the semantics of the code. Alternatively, we may give comments (the scaffolding) to the groups that describe an algorithm. Students then write code that corresponds to the comments. Of course, the ultimate goal is for students to generate all the code and think about efficient solutions, but novices (especially CS1 students) need support in their initial programming endeavors.

Code Debugging

A fifth activity is to give groups of students syntactically and logically buggy code. In this way all students can contribute to finding errors. For example, some students, who may be struggling with the course, will find the more obvious syntax and logic errors, while others are challenged to find more subtle errors that are not easily recognized. Sometimes students are asked to do this individually and then split up into groups and share results. At times, constructive thinking is promoted further when students disagree on what is or is not an error in the code.

Lecture Note Reconstruction

We have used this technique in class sessions of 50 min as well as 75 min. Here, we ask students to not take notes during a mini lecture but instead to pay close attention. After 15 min of lecture they have a few minutes to reconstruct an outline of the lecture from memory. They then meet in their groups and refine their notes further. This is a great exercise to help students improve their listening skills (Bonwell, 1996).

For example, we deliver a lecture on sorting algorithms that compares insertion sort, selection sort, and bubble sort. Students are instructed to not take any notes and simply observe us demonstrate each algorithm. We have large cardboard cards with numbers on them that we manipulate as if they are integers in an array. After 'seeing' the algorithms in action, students write descriptions in their own words.

Even though these activities seem to promote a pleasant pedagogical variety in a CS1 classroom, we could not conclude if these activities were actually making a difference in the learning of our CS1 students. Thus, we decided to compare the cognitive and affective differences of students whose instructors used activities such as these with students whose instructors maintained more traditional lecture environments.

METHOD

Design of the Investigation

Six sections of a CS1 course participated. Each section started with approximately 32 students and was taught by a different instructor. Instructors used their natural teaching style with respect to the use of constructive activities in the classroom. Neither the students nor instructors were aware of the study until the end of the term.

The 15-week course met four times per week including three 1 hr lectures and a 2 hr closed laboratory. We used Java to introduce the standard CS1 contents such as variables, selection statements, repetition statements, methods, and objects. Students completed six programming projects, two midterm exams, and one final exam.

The course was coordinated to keep the content consistent between each section. All sections used a common syllabus and instructors kept in contact with each other via e-mail and impromptu conversations in the hallway. The intent was to have all sections cover the same material at about the same pace. Programming projects were identical for all sections. The midterm exams covered the same material but were not identical. Instructors wrote their own midterm exams. For consistency, all sections took an identical final exam at the same time on a Saturday morning.

Materials

We developed a survey to measure the student's perceived frequency of constructive activities performed in the classroom (Appendix). The brief

instructions included examples of activities such as hands-on exercises, writing code, reading code and other group activities. There were only two questions.

The first question was "Which of the following best describes your classroom this semester (during lecture, not lab time)?" The five choices for responses were: (1) no classroom activities at all, (2) a few activities throughout the course, (3) activities were performed about once per week, (4) almost every day, and (5) the course was entirely activity based with no traditional lecturing.

The second question was "If you completed any of these activities, how well do you think they helped you learn the subject?" The five choices for responses were: (1) poor, (2) fair, (3) OK, (4) good, and (5) excellent.

Procedure

Students completed the survey 2 weeks before the course concluded. Instructors also completed the survey. The instructors' perceptions of classroom activities generally matched their students' average response. Each section was assigned to one of four groups based upon the average student response to the first question. This was not difficult since responses clustered around one or two categories within each section. We identified two sections that had no constructive activities, one that rarely had activities, two sections that had activities about once per week, and one section that had activities almost every day.

Students completed the final exam on a Saturday morning. Instructors worked together afterwards to grade all of the exams. Each instructor was responsible for grading one page of the exam. Student names were hidden so instructors were grading an anonymous exam. This process reduces instructor bias during grading and insures that exams are graded consistently.

We recorded final exam grades for each student. These grades were the most reliable data since all students completed the same exam. This was not the case with the two earlier midterm exams. Programming project grades were not considered reliable because instructors graded their own projects and perhaps used different grading criteria.

Students completed a campus-mandated evaluation form at the end of the course. Three questions were selected for this study that best represented their attitude toward the course and instructor. Student evaluation responses and exam grades were collected for each student and recorded in a spreadsheet. Random letters were assigned to each section regardless of which category

they were assigned to with respect to constructive activities. During initial analysis we were looking for differences between any sections. This strategy helped to keep the analysis objective.

One measure of class performance is to consider the WDF rate (Chase & Okie, 2000). This is a measure of the number of students that withdraw from the course (W) or earn a low grade (D or F). These students will not be able to continue to CS2 without repeating the current course. We calculated the WDF rate for each section based on the original enrollment. Final grade reports have a W grade for students who withdraw from the course after the first week of the term.

RESULTS

Final Exam Grades

There was a noticeable linear improvement of average final exam grades (Figure 1). The highest possible grade was 100. Average grades increased as the frequency of constructivist activities increases. Upon further analysis, we found that a slightly positive and significant correlation did exist ($p < .05$, $N = 147$). Spearman's rho was used to test for correlation since we did not assume normality of underlying data (the scores of a few sections were slightly skewed). Thus, we chose the more reliable non-parametric test.

To determine if a significant difference occurred between any of the sections, we used the non-parametric Kruskall–Wallis test. This test is similar to a one-way Analysis of Variance, but instead performs its calculations by

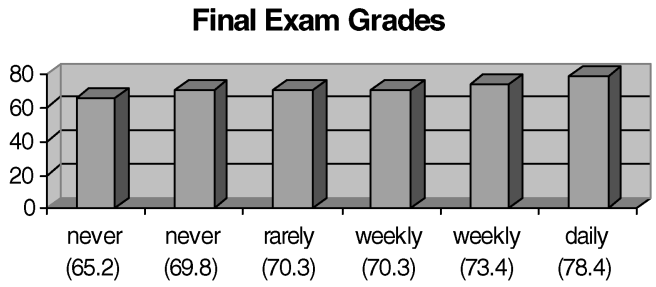

Fig. 1. Final exam grades by frequency of constructive activities.

Table 1. Test for Significant Differences Among Exam Scores Based Upon Section.

Kruskall–Wallis mean rank scores			Kruskall–Wallis significance	
Frequency of activities	*N*	Final exam Mean rank		Final exam
Never	27	60.44	Chi-Square	8.188
Never	26	70.58	df	5
Rarely	28	72.73	Asymp.	.146
Weekly	22	71.48	significance (*p*)	
Weekly	22	78.70		
Daily	22	94.11		
Total	147			

ranking each student's score. Thus, a mean rank score is shown in Table 1 rather than a mean score. As with Spearman's rho, this test does not assume underlying normality of data.

There was a large difference between a Never section (60.44) and the Daily section (94.11), however, this difference was not statistically significant ($p = .146$). Thus, we could not statistically conclude that frequency of constructive activities did account for a difference in exam grades.

Withdrawal Rate

The WDF rate for each section is shown in Fig. 2. There is no apparent relationship between the WDF rate and the frequency of constructive activities.

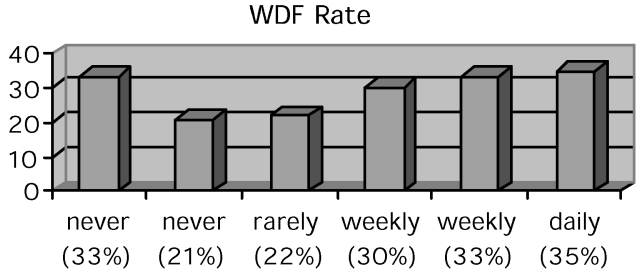

Fig. 2. WDF rate by frequency of constructive activities.

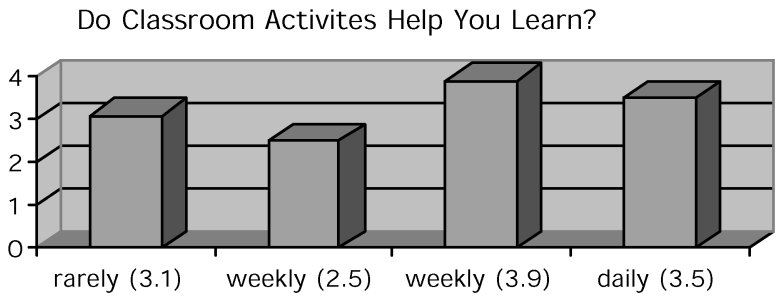

Fig. 3. Average student responses.

Subjective Assessment

Students were asked how well they thought classroom activities helped them learn the material (1: poor and 5: excellent). Average responses for sections that had some level of activity are shown in Fig. 3. There is no apparent correlation with frequency of activities and average student response.

Student Satisfaction

Responses to the student survey at the end of the term were consistent within each section (Figure 4). However, there was no apparent correlation with student satisfaction and frequency of classroom activity.

We selected the following three questions as a representative subset of the survey.

(1) The course was taught well (1: strongly disagree, 5: strongly agree).
(2) I enjoyed taking the course.
(3) I have benefited by having this instructor.

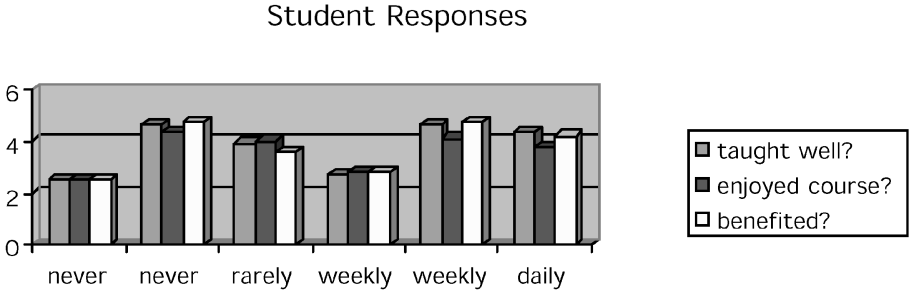

Fig. 4. Student satisfaction with the course and instructor.

DISCUSSION

In this Section we discuss the results, consider the experimental design, and make recommendations for further investigation.

Results

The slightly positive correlation among final exam grades and frequency of constructive activities is interesting. For example, this could suggest that a variety of constructive activities help students understand the material better and may hold their attention to the task longer. Bonwell and Eisen (1991) suggest that student attention wanes after 10–20 min. Providing scheduled activities throughout a lecture may increase attention and therefore increase comprehension.

However, correlation does not imply cause and effect, and results from our study do not allow us to conclude any dependence of final exam scores upon frequency of collaborative and constructive activities. This could be due to the fact that we need to better structure some of our group activities (Slavin, 1990). One must also consider the other factors involved in constructivism such as promoting authentic contexts. Because we gave the same simplified authentic projects to all students, this factor may have negated some of the effects of classroom collaboration.

The WDF rate did not provide much insight for this study. One concern is that the WDF rate varied considerably even with the same instructor. For example, one instructor taught three sections of this course (only one was randomly chosen to be included in this study). The WDF rate for his three sections were 22, 33, and 38%. A second concern with the WDF rate is that with small classes only a few students will greatly affect results. Sections started with approximately 32 students. The two extremes in our study were 7 (21%) and 11 students (35%) who withdrew or earned a poor grade. This is only a difference of four students, which can be contributed to a number of factors such as student workload, time of day the course is offered, and any number of other factors not controlled by this study.

Experimental Design: Present and Future

It is challenging to implement a sound experimental design in a college teaching environment. There are almost always differences that are difficult to control such as teaching style, grading criteria, and size of the class. The strength of this study was that six sections taught by different instructors

covered the same material, students completed the same final exam, and care was taken to consistently grade the exams across all sections. Many institutions do not have the luxury of offering six sections of the same course with different instructors.

One potential bias of the data is that all sections had a weekly lab session, which involved programming activities. This level of classroom activity, even though it was during lab time, may have affected the outcomes. An alternative design could compare the effect of constructive activities by having control sections with no lab time. Another option could include a group whose classroom work is completely based upon developing programs solely in the lab with no lecture. This would more authentically simulate the work of a computer scientist and consequently align more with constructivist beliefs. Our students often comment that they learn the most in a lab. This is not surprising. Not only they are getting feedback on their thinking from their peers and the instructor, but they are also getting direct feedback from computer-generated errors. These are their own errors and not ones made up by the instructor.

Our study did not account for programming knowledge of incoming students. Future research designs will need to control for variables such as these. Other factors to consider are gender, overall grade point average, and age level of the student. One could perform a multiple factor analysis of variance that investigates how grouped variables such as gender and frequency of collaborative activities may affect final exam outcomes.

In our study, the six instructors collectively wrote the final exam early in the term. One might argue that exam grades were affected by how closely each instructor ''taught to the exam''. One improvement would be for a third party to write the exam. Instructors would have to be informed of the general content but would not be familiar with the specific questions.

Lastly, one could consider a qualitative research design. This form of inquiry emphasizes the understanding of collected data rather than the reduction of that data to mere statistics. Qualitative researchers believe that a reductionist approach may miss important meaning generated by underlying data. A researcher using this design could interview students in a constructivist-based CS1 classroom, interview the instructor, observe lab and classroom activities, analyze student learning on open-ended project or exam questions, and subsequently infer themes that may otherwise not be accounted for in a quantitative design. Because classroom dynamics are extremely complex, the interaction of those dynamics may be difficult to

control in a concise quantitative design. Indeed, the creation of the Logo programming language was based upon Seymour Papert's observation of children learning with computers. It was not based solely upon careful analysis of quantitative outcomes.

CONCLUSION

The outcome of this study supports the notion that collaboration and constructive activities can aid learning, whether statistical or not, in a computer science classroom. However, measures such as WDF rates may not be affected at all. Acknowledging that it takes time to adjust pedagogy, it is compelling to refine these techniques and perform studies based upon the other research designs presented. Just as aspects of the computer science discipline frequently undergo change, computer science educators must also be willing to explore change in their teaching and philosophy of learning.

ACKNOWLEDGEMENTS

We appreciate the willingness of our colleagues to provide access to their classrooms: Jerry Scripps, Scott Lis, Ken Johnson, and Roger Ferguson. We also want to thank Dr. Paul Stephenson for his assistance with the statistical analysis. Finally, we acknowledge the contributions of the students who made this study possible.

REFERENCES

Bednar, A.K., Cunningham, D., Duffy, T.M., & Perry, J.D. (1992). Theory into practice: How do we link? In T.M. Duffy & D.H. Jonassen (Eds.), *Constructivism and the technology of instruction: A conversation* (pp. 17–34). Hillsdale, NJ: Lawrence Erlbaum Associates.

Ben-Ari, M. (1998). Constructivism in computer science education. *The Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Science Education* (pp. 257–261).

Bonwell, C.C. (1996). Enhancing the lecture: Revitalizing a traditional format. In T.E. Sutherland & C.C. Bonwell (Eds.), *Using active learning in college class: A range of options for faculty.* San Francisco: Jossey-Bass.

Bonwell, C.C., & Eison, J.A. (1991). Active learning: Creating excitement in the classroom. *ASHE-ERIC Higher Education Report No. 1.* Washington, DC: The George Washington University, 1991.

Chase, J.D., & Okie, E.G. (2000). Combining cooperative learning and peer instruction in introductory computer science. *SIGCSE Bulletin*, *32*, 372–376.
Fosnot, C.T. (Ed.). (1996). *Constructivism: Theory, perspectives, and practice.* New York, NY: Teachers College Press.
Gadanidis, G. (1994). Deconstructing constructivism. *The Mathematics Teacher*, *87*, 91–97.
Gorriz, C.M., & Medina, C. (2000). Engaging girls with computers through software games. *Communications of the ACM, 43*, 42–49.
Keeler, C.M., & Anson, R. (1995). An assessment of cooperative learning used for basic computer skills instruction in the college classroom. *Journal of Educational Computing Research, 12*, 379–393.
Kim, M.K., Sharp, J.M., & Thompson, A.D. (1998). Effects of integrating problem solving, interactive multimedia, and constructivism in teacher education. *Journal of Educational Computing Research, 19*, 83–108.
McGrath, D. (1990). Eight ways to get beginners involved in programming. *The Computing Teacher, 18*, 19–21.
Phillips, D.C. (1995). The good, the bad, and the ugly: The many faces of constructivism. *Educational Researcher, 24*, 5–12.
Ramsey, P., Rada, R., & Acquah, S. (1994). Collaborative learning for computer science students. *Journal of Computers in Mathematics and Science Teaching, 13*, 377–389.
Skinner, B.F. (1953). *Science and human behavior.* New York, NY: Free Press.
Slavin, R.E. (1990). *Cooperative learning: Theory, research and practice.* Englewood Cliffs, NJ: Prentice-Hall.

Appendix

CS 162 Survey

We are conducting a survey of teaching methods used in CS 162. For this survey, consider the definition of 'classroom activities' to be working alone or in small groups to solve a problem during class. The instructor is not lecturing during these periods. Which of the following best describes your CS 162 classroom this semester (during lecture, not lab time). Your response should be anonymous.

Pick one:

_____ The instructor lectures and responds to student questions. There are no 'classroom activities'.

_____ In addition to the instructor's lecture, we completed 'classroom activities' (alone or in groups) on a few occasions during the semester.

_____ In addition to the instructor's lecture, we completed 'classroom activities' (alone or in groups) about once per week.

_____ In addition to the instructor's lecture, we completed 'classroom activities' (alone or in groups) on most days.

_____ The instructor never lectures. We completed 'classroom activities' (alone or in groups) every day and all day.

If you completed any of these 'classroom activities' (not during lab), how well do you think they helped you learn the subject?

poor _____ fair _____ OK _____ good _____ excellent _____