Computers in the CS1 Classroom

William T. Tarimo, Fatima Abu Deeb and Timothy J. Hickey
Computer Science Department, Brandeis University, 415 South Street, Waltham MA 02453, U.S.A.

Keywords: Flipped Classroom, Blended Learning, Computer-Mediated-Communication, Pedagogy Design, Teaching Introductory Computer Science, Educational Technologies, Web-based IDEs.

Abstract: There are two basic approaches to flipping an introduction to programming class (CS1). One involves requiring all students to bring computers to class and to work alone or in groups to solve programming problems. The other approach is to ban computers from the classroom and to require students to solve programming problems on paper. In both approaches the students' attempts are shared with the class and discussed. In this work, we describe an experiment in which we compared these two approaches for a large programming class. We found that the use of computers had no statistically significant effect on the students' learning outcomes, enjoyment of the material, self-assessment of their understanding, use of teaching assistant resources, or self-estimate of how many hours they invested outside of the classroom. We did find that a statistically significant number of students preferred problem solving with friends using computers rather than on paper. We also found that the instructor had much more detailed information about individual student's interaction in class when computers were used, since all student interaction with the coding tools could be logged and analysed. We conclude that, although many faculty are wary of requiring computer use in large classes, there is evidence that students prefer it, it does not negatively affect learning outcomes, and with appropriate tools and pedagogy, it gives the instructor a much deeper and more nuanced view of student performance in the class.

1 INTRODUCTION

There is a growing body of evidence which demonstrates that active learning pedagogies improve learning outcomes in a wide variety of courses, including introductory programming courses (Amresh et al, 2013; Bates et al, 2012; Stone, 2012). It is very natural to allow students to use their laptops in class during active learning sessions of an introductory computer science course. Many faculty, however, are wary of requiring computer use during class sessions since they feel students might become distracted.

In recent years we have seen many new developments in the way teaching and learning are accomplished inside and outside of the classroom. The last decade has seen research, development and adoption of new pedagogies, classroom technology and software applications. One such new pedagogy technique has been the 'inverted' or 'flipped' classroom in which static content is covered outside of class (through readings or videos) and class time is devoted to more interactive and engaging activities. Even though most approaches have leveraged the ubiquity of technology, flipping a classroom does not necessarily require the use of computers or other networked technology.

In this work we present our case study of partly-flipping a large CS1 class. The course was an Introduction to Programming in Java and C in which we used a partly-flipped pedagogy that combines both in-class lectures and in-class programming challenges often using a Think/Pair/Share technique (Kagan, 1989). Since the course was taught in two sections (of about 150 students each), we were able to design an experiment to evaluate the effect of two approaches to partly-flipping the classroom. The first approach is to require all students to bring a laptop or tablet to class and use their computers for various interactions, to answer questions and to solve coding challenges. The second approach is to ban computers from the classroom and to require students to solve problems with pen and paper and to be prepared to present and discuss their solution to the class if called upon.

In the computer-mediated sessions students used two web-based applications, TeachBack (Hickey and Tarimo, 2014) and Spinoza (Abu Deeb and Hickey, 2015), to interact and solve programming

T. Tarimo W., Abu Deeb F. and J. Hickey T..
Computers in the CS1 Classroom.
DOI: 10.5220/0005436600670074
In *Proceedings of the 7th International Conference on Computer Supported Education* (CSEDU-2015), pages 67-74
ISBN: 978-989-758-108-3

problems and share their solutions with the instructor and the class. In the non-computer sessions, we endeavoured to replicate the same pedagogy using pen, paper, blackboards and the instructor's computer projected on a screen. Both sessions covered exactly the same material and used exactly the same pedagogy. Students received nearly identical lectures and were given the same programming challenges. The only difference is that one section was allowed to use their computers to solve the programming problems, while the other section had to use pen and paper only.

In the following sections we present the experimental design that was used to compare the computer and non-computer approaches to the pedagogy. We then proceed to describe the pedagogy used and we compare the way it was implemented using the computer-mediated and pen-and-paper based approaches. Finally we present the results of the experiment and discuss its implications for computer use in the partly flipped introductory programming classroom.

2 THE EXPERIMENT

Introduction to Programming in Java and C is the first course in the Computer Science major in our department. Students who performed well in an equivalent CS1 course in high school may skip the course, but all other potential majors are required to take it. It was taught in two sections (self-selected by the students). One section had 136 students and the other had 148. Both sections had the same instructor, exams, homeworks, teaching assistants, and daily lesson plans. For both sections, we provided screen recordings of each class that students could review at their leisure.

The course was divided into 4 units, each lasting about 3 weeks. Each unit culminated in a 90-minute exam that provided a summative assessment of student mastery of the material for that unit. In the first two units students were required to bring their computers to class and to interact with the instructor using TeachBack and Spinoza. Ten percent of their final grade was based on the number of TeachBack formative assessment questions they answered (whether the answers were correct or not). During Units 1 and 2, students were required to bring computers to class and use them to interact with the instructor and their peers. During Unit 3, computers were banned from section 1 while still being required in section 2. During Unit 4, the protocol was reversed. Computer were required in section 1 and banned in section 2. This provided us with two units of control in which both sections used computers, and two experimental units where one section required computer use and the other banned its use.

3 THE ACTIVE-LEARNING PEDAGOGY

Before each week of classes students were assigned topics or subtopics to read and as a weekly homework - submit a short reflection on what they learned and any confusing ideas in the reading. Each class had lectures intermixed with class-wide interactive activities. The lectures involved PowerPoint slides, notes from the class website, live coding demonstrations by the instructor, and visits to various websites. The interactive activities included short answer questions as well as programming challenges.

In this section we discuss the main pedagogical techniques used in the two versions of the class and along the way we introduce the TeachBack and Spinoza tools. TeachBack (Hickey and Tarimo, 2014) is a web-application with three main features: a supervised back-channel forum (called the Forum) where students can ask and answer questions with each other and with TAs who are always present during classes, a pie chart and timeline plot (called the Feedback) where students can indicate if they are confused, engaged, or bored and include a 50 character explanation of their affect and cognition (i.e. emotional and comprehension) states, and a clicker-type application (called the iResponder) which allows the instructor and TAs to collect and grade student answers to formative assessment questions during the class. Spinoza (Abu Deeb and Hickey, 2015) is a web-based Java IDE that allows students to solve simple programming problems online and provides the instructor with a real-time view of the progress of the class with similar solutions grouped together.

3.1 PowerPoint Lecture Activity

Although the students were required to read the text before class, we often began a class with a PowerPoint overview of the main ideas presented in the readings. In the computer-based version of the class, students could view the PowerPoint slides on their computers and ask questions of the teaching assistants using the TeachBack Forum. In the pen-

and-paper version they could print out the slides on paper before class and ask questions by raising their hands and interrupting the class flow.

3.2 Live-Coding Activity

Another lecture-style activity is when the instructor solves or demonstrates a programming problem using a Java IDE and the class watches (or in the computer-mediated version, follows along). This can be made interactive by asking students to provide suggestions for how to solve the problem. In the computer-mediated version when students are following along with the coding using Spinoza and they encounter syntax errors they can interact with the TAs using the TeachBack Forum without interrupting the class.

3.3 Answering Student Questions during Class

In both versions of the class, students were encouraged to ask questions if they were confused. In the pen-and-paper version, students would raise their hands and engage with the instructor while the class paused. In the computer-mediated version, students used the Forum feature of TeachBack to ask questions online, and have their questions answered by TAs assigned to the course, or sometimes by other students who were monitoring the forum. The instructor would briefly review the forum with the class at the end of most activities.

3.4 Posing Questions for Students to Discuss and Answer

After a lecture activity, we would usually pose a series of questions and ask the students to think for a minute about a solution, then to talk with their neighbors about their solution, and finally to share their solutions with the class. Typical examples would be predicting the result of evaluating a snippet of code, or finding a bug in a piece of code shown on the projector. In the computer-based version, we used the iResponder feature of TeachBack. Figure 1 shows a typical activity in which the instructor projected a method on the screen and asked students to predict the return value for various calls. iResponder allows the instructor and TAs to not only see the solutions (grouped) but to grade them and assign points and comments. Once a sufficiently large number of students have submitted an answer, the instructor reviews the most common solutions and leads a short class discussion on the different approaches and the different kinds of errors. In the pen-and-paper version, it is difficult

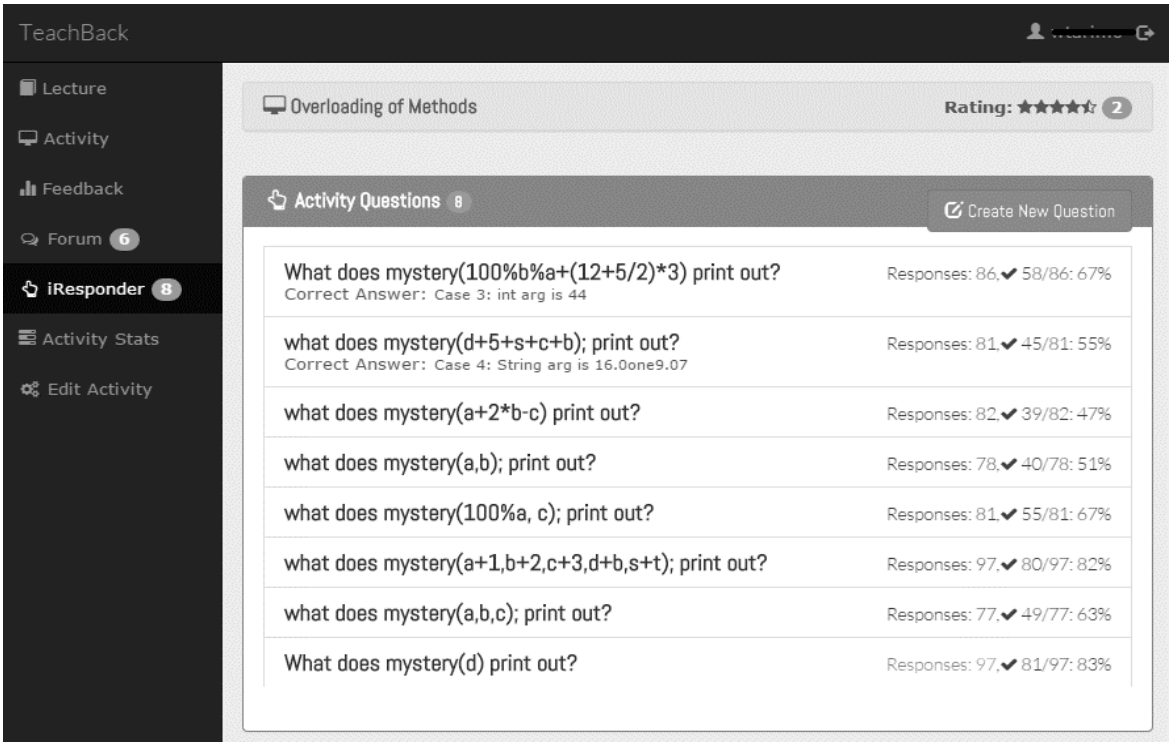

Figure 1: A typical iResponder screen.

to determine how many students have completed the activity and it is hard to tell what the most common solutions and errors were. Students were motivated to solve problems in the pen-and-paper class by randomly selecting students to describe their solutions (possibly on the board or typing into the instructor's computer).

3.5 Programming Problems

In this activity, students are given a programming problem and asked to think about how they would solve it and then work with their neighbours to come up with a solution. For example, students could be asked to write a method with three integer parameters that returns true if the parameters all have different values.

In the computer-mediated version of the class, we used a web-based Integrated Development Environment (IDE) called Spinoza that allows instructors to quickly create a programming problem. Figure 2 shows the student view of a Spinoza programming problem which provides a description of the problem on the left, some initial scaffolding code in the centre, a "Run" button below, space for the output on the right, and the results of an instructor supplied set of unit tests at the bottom. Students can then write, run, and debug the problem using the web-based IDE. Spinoza has an instructor's view which shows the number of students that have hit the "Run" button and it groups the programs together based on a similarity function (ignoring white space, variables names, etc.). The instructor can see in real-time the most popular proposed solutions to the problem and can view and debug those solutions in front of the class. The debugging process itself can be formulated as a Think/Pair/Share model (Kagan, 1989), where students try to find and discuss the bugs (both syntactic and logical) in small groups before sharing with the class.

In the pen-and-paper version of the class, programming problems are displayed on the screen and students are asked to write their solutions on paper. The instructor then randomly selects students to share their solutions. The disadvantage of this approach is that the instructor doesn't know what the most common solutions or errors are and the process of sharing a solution with the class is more time consuming.

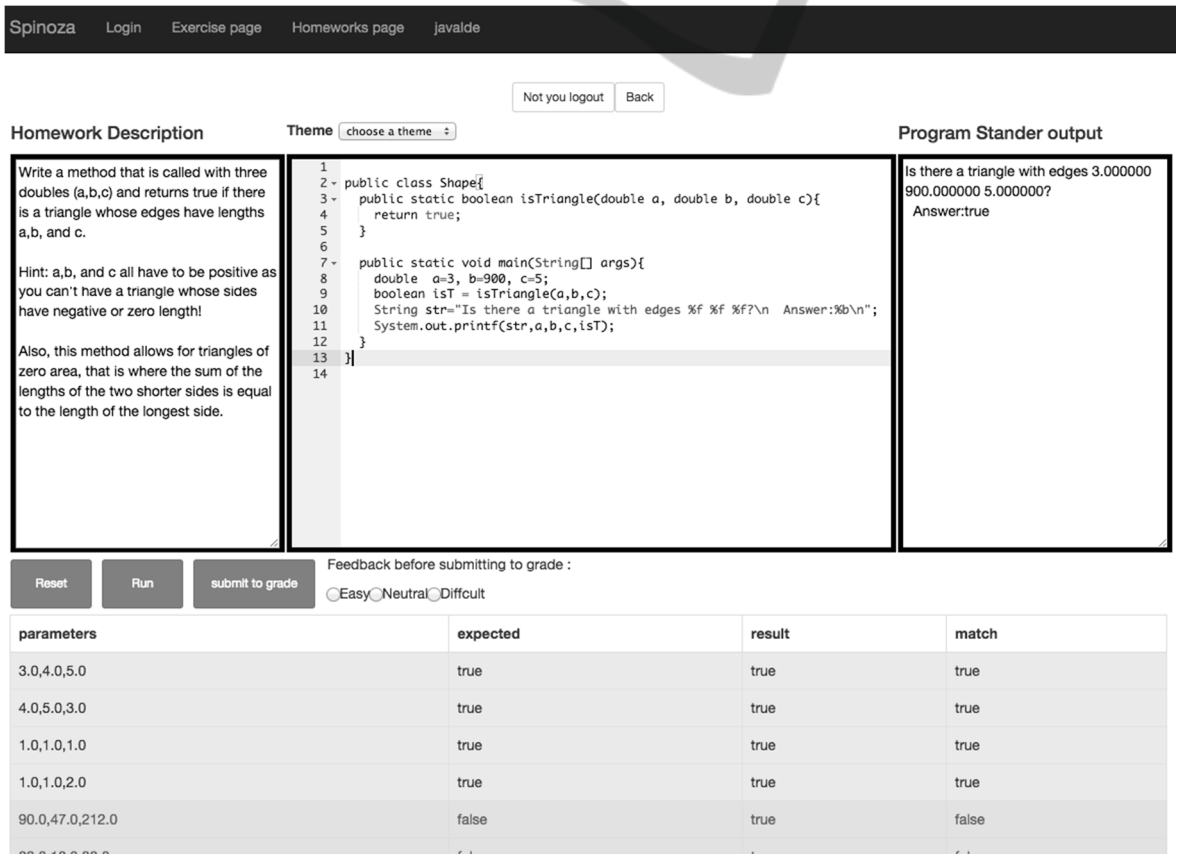

Figure 2: The student view of a Spinoza problem.

3.6 Feedback

After new material has been introduced we often ask the students for feedback, typically at the end of an activity or class. We ask whether they are confused, bored, or engaged by the material and also ask for a short comment. In the computer-mediated version, this is done using the TeachBack Feedback feature, which displays a pie chart showing the three responses. Hovering over one of the pie slices reveals a list of the comments students provided. We often find 20%-50% of students report feeling confused when a class introduces new material (e.g. arrays or the *for-each* loop). This provides an excellent opportunity to reassure them that it is natural to feel confused when learning new material. The comments also show what confused them or expand on their affect. At this point the instructor also clarifies the various confusion issues. Since it is so easy to get and analyse feedback from students using TeachBack, we often get feedback after each activity in a single class. TeachBack also provides an instructor/TA view of the daily progress of individual students using performance and participation statistics at an activity, lecture and course levels.

In the pen-and-paper version, we ask students to put this information on a small card or piece of paper, which is then reviewed by the instructor after the class. One disadvantage of this approach is that we can't report the results until the following day and it can take 30 minutes to an hour to read through a few hundred separate comments.

4 DATA COLLECTION

After each unit, students were asked to complete a survey where they self-assessed their level of understanding of the material in that unit as well as their level of enjoyment of the material in that unit. In units 3 and 4 they were also asked to rate each of the different styles of pedagogy employed in terms of its effectiveness for their own learning.

We kept track of the number of students from each section that visited TAs during each of the units and asked students to estimate how many hours they spent working on the course outside of class. We also kept track of each student's participation in various components of TeachBack during each class, each unit, and the semester. Finally, grades on the four unit quizzes as well as course grades were used to measure mastery of the material by unit and over the entire course.

5 RESULTS

We found four main results from our analysis of the data which we summarize below:

5.1 The Use of TeachBack/Spinoza in Class Does Not Harm Learning Outcomes

In Unit 3, computers were banned in section 1 and required in section 2. In Unit 4, the reverse policy held, computers were required in section 1 and banned in section 2. We found that there was no statistical differences between the two sections during those units in terms of quiz scores, student satisfaction, student self-assessment of understanding, or student use of teaching assistants. From the surveys at the end of each unit, students self-reported their levels of learning and satisfaction in the range [1-5]. As seen in figures 3 and 4, the averages on each section do not indicate any significant influence from the changes of pedagogies in units 3 and 4.

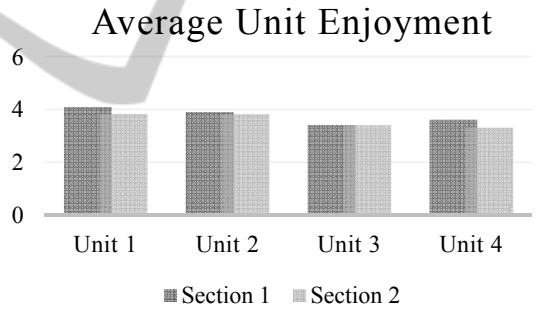

Figure 3: Average perceived enjoyment.

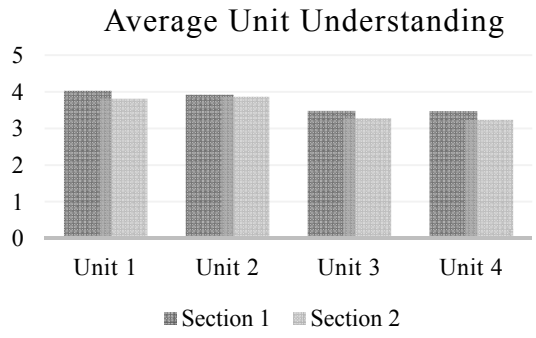

Figure 4: Average perceived understanding.

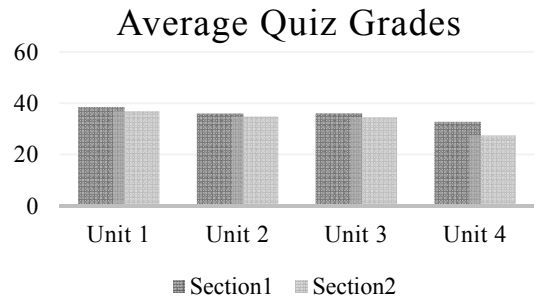

Figure 5: Average end-of-unit quiz grades.

Section 1 generally indicated a higher level of enjoyment, understanding, and mastery than section 2, for all units, but that increased level of understanding was not statistically significant.

For example, in Figure 4, the difference between the average understanding in unit 3 between sections 1 and section 2 was 0.17 but the p-value for the two-tailed unpaired T-test for those means was .20 which is not significant. Likewise, in Figure 3 the difference of average enjoyment for unit 4 between sections 1 and 2 was 0.23 but the p-value was .12, again indicating no significant difference. None of the apparent differences in section 1 and section 2 shown in these three figures was significant at the .10 level.

If use of computers was especially distracting, we would expect to see Section 1 outperform Section 2 in Unit 3, and the opposite occur in Unit 4. No such effect was found.

5.2 Most Students Prefer using Computers in Class

When asked about the two different styles of active learning - writing programs with your neighbours on paper versus writing programs on your computer while talking with your neighbours, the use of computers was thought to be more effective and the results are statistically significant. Students used a five point scale to rank effectiveness of learning from 1 = not effective to 5 = very effective. Solving programming problems with friends using pen-and-paper was ranked at 2.96/5 and solving programs using Spinoza with friends at 3.65/5 with a difference of 0.69. This is significant at the 0.001 level using a two-tailed paired T-test. The 95% confidence interval of the difference is 0.5 to 0.88.

Below are some typical comments from students after unit 4. Here is a section 1 student, happy to be able to use his computer again in class:

-*"I really enjoyed when we got to live code in class. It was helpful to either follow along with what [professor] was typing or work on building up the program with the people around us. It allowed me to see what thought process has to go into building up a program."*

And here are comments from students in section 2 explaining why they were disappointed about not being able to use computers in class:

-*"The lack of computers makes following along a lot less interesting and understanding class material becomes much more difficult."*

-*"Taking notes on paper and not being able to practice coding in class slowed down acquisition of the material greatly. It [took] much longer for this unit than others to master the material. I also disliked being asked to work in teams or to talk to people in class, but that's because I'm shy ..."*

-*"We can't use computer[s] to do real-time programing in class. To make it up, I have to go back home and watch the class recordings to brush my memory on what programing topics we went through in class that day. It is really time consuming."*

5.3 Some Students Were Distracted by Computers in Class

A close examination of the student comments about each unit demonstrated that there was a group of students who did not feel they learned as well with computers as they did without. Indeed there were a few students who would attend the lectures from the other section when the pedagogy was switched because they felt they could not learn well when required to interact with a computer in class. These were mostly students who reported being easily distracted in general. Below are some comments from students indicating what they liked about unit 4 when computers were not allowed.

-*"Not using a computer, it lead me to better concentrate."*

-*"Not being allowed to use our computers helped for concentration and focus."*

Most students, however, didn't report being distracted by the use of computers in the class, contrary to the worries of many instructors. This observation is largely due to the nature of the pedagogy. The division of the class time into short interactive activities allowed students to always be engaged with the material, their peers or the instructor. There was no time for students to get side-tracked into distraction with non-class related endeavours.

5.4 Students Generally Approved of the Active Learning Approach

In general, students appreciated the pedagogy used in the class, whether or not we were using computers. Here are some illustrative comments.

-*"The class was very lenient towards our learning and it's a great feeling to know that the teaching staff is very forgiving for us 'newbies'. Learning is the number one goal."*

-*"I was forced to try to learn the material to the best of my ability beforehand to be as prepared as possible whether or not I was using my computer or notebook."*

6 RELATED WORK

A recent study involving flipping an introductory computer science class was performed by (Amresh et al, 2013), where students would watch prepared lecture videos before classes, and have interactive discussions in class. Through summative assessments, this flipped model was found to produce higher average test scores. However, due to many years of traditional classrooms, students found this new approach to be overwhelming at times, especially as the videos and reading became boring and less engaging. In regard to this, (Bates et al, 2012) point out that successful flipped classes require the acceptance and embracing of this new unstructured and contingent lecture approach where the instructor is a coach of learning. In this case study in an introductory physics class, students were assigned pre-class readings and quizzes, and class meetings involved discussions driven by clicker questions. An important factor for success is to have access to or create sufficient clicker questions for good discussions. If students can be motivated to complete the work outside of class, flipped classrooms can enable more and deeper understanding without necessarily covering less content. Since students are more exposed to the materials in pre-class and in-class activities, the flipped pedagogy has the advantages of developing life-long learners, increasing engagement during classes, and increasing interactions among students and the instructors (Stone, 2012).

Systems similar to the Spinoza system used in this study have been developed to facilitate teaching introductory programming classes. JavaBat (Parlante, 2007) is a web application that helps students to build coding skills by providing immediate feedback to small problems in which they write code for the bodies of single methods. The system generates several tests (handwritten by the instructor) and shows students the results of those automatic tests. Students can specify a teacher who can then see their work and follow their progress, but the teacher cannot write comments or otherwise communicate with the students through the tool. Another system is Informa (Hauswirth and Adamoli, 2009), a clicker software system for teaching introductory programming with Java. Informa has been used in flipped classrooms as a way to support active learning of programming skills. It supports several different types of questions, including problems requiring students to write Java code, but it does not run the students' code and it is not web-based, it requires a Java app to be downloaded and installed. It also allows students to download and comment on other students' solutions. Spinoza allows instructors and TAs to view and comment on student programs, but does not currently allow students to comment on other students' code.

7 CONCLUSIONS

The results of our study demonstrate that the use of computers did not affect learning outcomes in any statistically significant way. One explanation for this surprising finding is that the key factor in student learning is the pedagogy itself, not whether the students had computers in class or not. The thought process involved in trying to solve programming problems can be pursued just as effectively using pen-and-paper as using computers. The highly interactive pedagogy itself encouraged students to maintain high levels of interaction, engagement and motivation with the material whether they used computers or not.

We know from previous studies that active learning in flipped classes is a more effective pedagogy than straight PowerPoint lectures (Amresh et al, 2013) and the results from this paper suggest that this pedagogy can be delivered either with or without a computer.

The various avenues of interaction offered by tools like TeachBack and Spinoza offer increased participation and involvement rates. But that is not all, like most computer-mediated communication tools, TeachBack and Spinoza allow content and conversations to be stored and accessed at later times. Moreover, participants don't have to be in the same physical locations, and users can engage in multiple conversations at once. In a way, these tools

liberate learning and teaching from constraints of time and distance (Hickey and Tarimo, 2014; Reed, 2000) where barriers such as distance, disabilities, shyness and cultural difficulties are overcome. Our proposed computer-mediated pedagogy features various interactive and engaging activities that do not give students the opportunities to get distracted. However, as we have discovered in this study, there are a few students who are ill equipped to handle computer-mediated interactions and online environments. Our results suggest that it might be worthwhile to offer two versions of the CS1 class, one which is fully computer-mediated providing the instructor with high quality and timely information about student performance, and one that is not computer-mediated to accommodate those students who are prone to distraction when given access to a computer in class. Another alternative approach is to teach a hybrid class, where computers are only allowed during certain in-class activities and are banned at other times.

From the instructors' point of view, the use of computer-mediated pedagogy does have many benefits. As mentioned above, it provides a detailed record of the activity of each student in the class including which questions they answered, whether their answers were correct, how they tried to solve a programming problem, what their level of confusion was after each activity, etc. In this experiment, we did not try to use this additional data to customize our support for individual students in the class. We strongly suspect that this detailed information about individual students could be used to provide individualized support for at-risk students in a way that would make a statistically significant difference in learning outcomes. We plan to test this hypothesis in future experiments.

REFERENCES

Abu Deeb, Fatima and Hickey, Timothy J. "Spinoza: The Code Tutor" Proceedings of the International Conference on Computer and Information Science and Technology, May 11-12th 2015, Ottawa, Canada.

Amresh, Ashish, Adam R. Carberry, and John Femiani. "Evaluating the effectiveness of flipped classrooms for teaching CS1." Frontiers in Education Conference, 2013 IEEE. IEEE, 2013.

Bates, Simon, and Galloway, Ross. "The inverted classroom in a large enrolment introductory physics course: a case study." Proceedings of the HEA STEM Learning and Teaching Conference (2012). April 2012.

Erlich, Zippy, Iris Erlich-Philip, and Judith Gal-Ezer. "Skills required for participating in CMC courses: An empirical study." Computers & Education44.4 (2005): 477-487.

Hauswirth, Matthias and Adamoli, Andrea. 2009. "Solve & evaluate with Informa: a Java-based classroom response system for teaching Java." Proceedings of the 7th International Conference on Principles and Practice of Programming in Java (PPPJ '09).

Hickey, Timothy J. and Tarimo, William T. "The Affective Tutor." *Journal of Computing Sciences in Colleges* 29.6 (2014): 50-56.

Kagan, Spencer. "The structural approach to cooperative learning." *Educational Leadership* 47.4 (1989): 12-15.

Parlante, Nick. Nifty reflections. SIGCSE Bull. 39, 2 (June 2007), 25-26.

Reed, April. "Computer-mediated communication (CMC) and the traditional classroom." *Teaching with Technology Today* 5.6 (2000).

Stone, Bethany B. "Flip your classroom to increase active learning and student engagement." Proceedings from 28th Annual Conference on Distance Teaching & Learning, Madison, Wisconsin, USA. 2012.