TITLE: Rethinking the Gateway Computing Curriculum Across Engineering Disciplines

Dr. Michael Joseph Reese Jr., Johns Hopkins University

Michael Reese is the Associate Dean & Director at the Johns Hopkins Center for Educational Resources. Dr. Reese previously worked as an Educational Technologist at Caliber Learning and Booz-Allen and Hamilton. He also consulted with the University of Maryland School of Nursing on the launch of their first online degree program. He earned a Ph.D. in sociology at Johns Hopkins. His dissertation modeled how educational innovations diffused in higher education. He also earned an M.Ed. in educational technology from the University of Virginia and a B.S. in electrical engineering at Virginia Tech, where he was named the Paul E. Torgersen Leadership Scholar.

Prof. Michael L. Falk, Johns Hopkins University

Michael Falk is Vice Dean for Undergraduate Education and a Professor in the Department of Materials Science and Engineering at Johns Hopkins University's Whiting School of Engineering where he has served on the faculty since 2008 with secondary appointments in Mechanical Engineering and in Physics and Astronomy. He holds a B.A. in Physics (1990) and a M.S.E. in Computer Science (1991) from Johns Hopkins University and a Ph.D. in Physics (1998) from the University of California, Santa Barbara. He has received several awards for his educational accomplishments, and in 2018 received the Materials Research Society's Impact Award for his work on broadening participation in STEM and computing education. His education research focuses on integrating computation into the undergraduate core curriculum. Falk also served as the lead investigator for STEM Achievement in Baltimore Elementary Schools (SABES) an NSF funded Community Enterprise for STEM Learning partnership between JHU and Baltimore City Schools.

Dr. Joanne F. Selinski, Johns Hopkins University

Joanne Selinski is an Associate Teaching Professor and the Director of Undergraduate Studies in the Computer Science Department at Johns Hopkins University. She has MSE and PhD degrees in Computer Science from JHU, along with a BS in Mathematics from Chestnut Hill College. Her primary research interests are computer science education.

Dr. Sara Miner More, Johns Hopkins University
Dr. Ali Darvish, Johns Hopkins University
Ivan Sekyonda, Johns Hopkins University
Amy Brusini, Johns Hopkins University
Dr. Alejandra J. Magana, Purdue University-Main Campus, West Lafayette (College of Engineering)

Alejandra Magana is an Associate Professor in the Department of Computer and Information Technology and an affiliated faculty at the School of Engineering Education at Purdue University. She holds a B.E. in Information Systems, a M.S. in Technology, both from Tec de Monterrey; and a M.S. in Educational Technology and a Ph.D. in Engineering Education from Purdue University. Her research is focused on identifying how model-based cognition in STEM can be better supported by means of expert technological and computing tools such as cyber-physical systems,visualizations and modeling and simulation tools.

Dr. Ahmed Ibrahim, Johns Hopkins University

Ahmed Ibrahim is the Senior Education Research Consultant at the Johns Hopkins Center for Educational Resources. Dr. Ibrahim leads the initiatives of scholarship of teaching and learning (SoTL) at both the Krieger School of Arts and Sciences and the Whiting School of Engineering. Dr. Ibrahim earned a PhD in educational psychology (learning sciences) from McGill University and completed his postdoctoral training in developmental psychology at the University of California at Riverside (UCR). He earned a B.S. in electrical engineering from Cairo University, and a MSc. in computer engineering from New York University (NYU) before working in the engineering field for several years. Dr. Ibrahim's research interests include: (1) practices of science, (2) engineering design, (3) computer-human interaction, (4) pedagogical training, (5) educational research, and (6) assessment and program evaluation. Dr. Ibrahim has a number of publications in peer-reviewed journals such as the International Journal of Science Education (IJSE) and the International Journal of Science and Mathematics Education (IJSME). Dr. Ibrahim teaches graduate and undergraduate courses in research methods, assessment and evaluation, statistics, and proposal writing.

Nathan Graham, Johns Hopkins University

Nathan Graham is the Director of the Center for Digital and Media Initiatives at the Whiting School of Engineering at Johns Hopkins University. He is also a PhD candidate at Rutgers University.

Mr. Paul Huckett, Johns Hopkins University

Rethinking the Gateway Computing Curriculum Across Engineering Disciplines

Introduction

Engineering across all disciplines now requires the ability to deploy computing within engineering practice. New computational tools and computational analysis advances discovery and deepens fundamental understanding. Not surprisingly, there has been a surge of interest in computer science by undergraduates straining many institutions. Traditionally students of various disciplines have been introduced to computing through courses taught within the discipline of computer science (CS). There is a growing perception within the various engineering disciplines outside of CS that being introduced to computing within a disciplinary context is helpful for the later integration of computing into advanced coursework. It also helps students' careers [2], [3]. This shift has led to a diversification of the coursework by which engineering students are introduced to computing. These courses are taught in a variety of programming languages, chosen often due to their degree of application and perceived utility in the host discipline.

A major research university in North America undertook a redesign of its freshmen-year computing curriculum after seeing a proliferation of computing courses across engineering disciplines. The primary goal was to better align the learning outcomes across computing courses offered by various engineering disciplines. *This curriculum redesign also provided an opportunity to encourage faculty adoption of evidence-based teaching practices*. Research suggests that student-centered, active-learning strategies lead to deeper student learning and longer retention compared to lecture-based methods [4], [5]. These strategies also increase students' confidence in using computational methods and their recognition of the value of computational skills to help them succeed in future courses and careers [6], [7], [8]. There is evidence that students learn more in flipped courses [9]. The curriculum redesign team – comprised of faculty and instructional support staff - committed to employing a flipped course approach to the gateway computing curriculum. For this paper, a flipped course is defined as students watching lectures and reading content before they attend class sessions. Students then complete challenge problems under the supervision of faculty and teaching assistants during class time [10].

While evidence exists that students learn more in a flipped course, there are limitations in the design of these studies. For example, many employ single-group study designs [9]. This paper presents the results of a comparative analysis of two introductory JAVA computing courses taught with different pedagogical strategies – a flipped course approach and a traditional lecture course – during the same semester. The study will go beyond collecting and analyzing student perceptions, and also compare measures of student learning in the courses to identify if there is a difference in student outcomes between the pedagogical approaches. Evidence shows that flipped classrooms are more effective [10]. The team will also study how students' computational beliefs differ in the courses. The Technology Acceptance Model suggests that students' acceptance of technology like computational skills can impact their mastery [11]. This acceptance is dependent on their beliefs about the utility of computation, the perceived ease of use, and future intentions to use computational skills in subsequent courses and careers [12]. The authors will summarize how the data informed changes to future implementations of the course. The authors believe the findings and suggestions for improvement will benefit instructors teaching any first-year course, not just those teaching computing.

Research Design

The team conducted a comparative analysis of a flipped course and a traditional lecture approach for an introductory JAVA programming course. The two courses were both taught during the fall 2018 semester. The learning objectives for each course were similar. The primary difference was how the courses were taught. One instructor taught the traditional lecture course and three different faculty taught each taught multiple sections of the flipped course.

Course Details

In the traditional lecture course, 90 students met in a common classroom twice per week for 75 minutes. Students would generally follow the instructor through coding examples by mimicking the code on their laptops. Occasionally, the instructor would organize students into small groups to complete challenges that lasted 10-15 minutes. Students were evaluated in the class through projects and exams.

The flipped course was taught on the same topic – introductory JAVA programming – with similar learning objectives. In the flipped course, students attended one of eight sections three times per week for 50 minutes. No section was larger than 20 students. Students were assigned weekly readings from an interactive textbook that also included short comprehension questions. Three different faculty taught the flipped course (2-4 sections each). Faculty recorded weekly video lectures to supplement the textbook content. Students completed the readings and watched the lectures before class. The average video lecture was nine minutes long (median = eight minutes, min = < one minute, max = 26 minutes). During class, students worked on challenge problems to apply concepts learned from the interactive textbook and online lectures. Students worked in groups, requesting help from the instructor as needed. Two class meetings each week were facilitated by a faculty instructor and a lead course assistant. The Friday sessions were facilitated by a lead course assistant and two other course assistants. Students completed weekly quizzes and auto-graded coding challenges to assess their learning. Students self-scheduled proctored quizzes outside of class time in a computer lab. Students also completed four computing projects contextualized by an engineering problem throughout the semester. For example, students wrote a program to simulate and visualize heat distribution across a metal plate. Students took a final exam at the end of the semester.

During enrollment, students did not know the two JAVA courses would be taught differently. Upperclassman enrolled in the traditional course and only freshmen were permitted to enroll in the flipped course. Both groups, however, included students with diverse computing experiences (e.g., engineering majors, non-engineering majors, some computing experience, no computing experience).

Data Collection Strategies

The team employed several data collection strategies (See Table 1) to assess students' perspectives on the course, students' beliefs about the utility of computing in their future careers, and differences in students learning outcomes. Faculty used the CS1 concept inventory to measure learning differences. The CS1 concept inventory is a validated instrument that assesses students' mastery of computing concepts typically taught in introductory courses [13]. Students in each course took the concept inventory during the last week of the semester. The team measured students' self-beliefs about computing utility and intention through an end-of-semester survey several of the authors used in previous educational research (See Appendix 1). Student perspectives on the course were captured through the end-of-semester survey and standard course evaluations. Students in both courses were recruited to participate in end-of-semester focus groups to explore students' perspectives in more detail. The focus group recruitment was done purposefully to capture perspectives from students who performed well in each course and those who struggled. Focus groups were conducted by an education researcher from the institution's teaching and learning center. This allowed students to share their opinions while remaining anonymous from the instructor. Results were summarized for the faculty to ensure individual students were not identified.

The qualitative data – course evaluations and focus group transcripts - underwent close readings by an analyst to understand the intricacies of the students' feedback. Students comments were interpreted using an open-coding protocol motivated by an analytical grounded-theory-guided process [14].

Table 1: Data Collection Strategies

Construct Categories	**Instrument**
Student Learning of Core Programming Concepts	CS1 Concept Inventory
Student Perceptions of the Course	• End-of-Semester Survey • Standard Course Evaluations • End-of-Semester Focus Groups
Computational Beliefs: Utility, Confidence, Future Intentions	End-of-Semester Survey

Results

Student Learning

Students in both courses took the CS1 concept inventory at the end of the semester. There was no significant difference between students' overall performances on the CS1 despite the mean being almost 10% higher in the flipped course. (See Table 2). An item analysis was conducted for specific topics covered on the concept inventory (See Table 3). Students in the traditional lecture course outperformed students in the flipped course on for- and while-loop topics. Students in the flipped course outperformed on function return values. The team is cautious in interpreting these differences as meaningful for two reasons. First, there were few items comprising each factor (2-3 questions for each topic). Second, the entire CS1 concept inventory was too long to implement in a standard class session. The instructions for implementing the concept inventory stated the instrument was designed to be implemented in 60 minutes. Most students did not complete the concept inventory the first time it was implemented (over 50% did not complete questions 17-27; 40% of all answers blank). Based on this experience, the team decided to modify the implementation for subsequent sections. Students were only presented the first 17 questions of the CS1 concept inventory. These questions were chosen because they represented the key learning objectives of the course. The results reported in Tables 2 and 3 reflect the analysis on the common 17 questions asked of all students.

Table 1: CS1 Concept Inventory Overall Results

Traditional Lecture Course	**Flipped Course**

Sample size= 46
Mean= 7.26 (out of 17)
SD= 3.12
	Sample size= 93
	Mean= 7.88 (out of 17)
	SD= 2.78

Table 2: CS1 Concept Inventory Topic Results

Concept	**Traditional Lecture**	**Flipped Course**	**Differences**
For (3 items)	N=46	N= 93	Traditional Lecture Course Better (p<.01)
	Mean=1.41	Mean= 1.01	
	SD=.8987	SD=.6134	
Logical Operators (2 items)	N=46	N= 93	No difference
	Mean=1.22	Mean= 1.35	
	SD=.778	SD=.728	
While (3 items)	N=46	N= 93	Traditional Lecture Course Better (p<.001)
	Mean=1.76	Mean= .89	
	SD=.913	SD=.47	
Arrays (2 items)	N=46	N= 93	No difference
	Mean=.587	Mean= .785	
	SD=.645	SD= .73	

Function return values
(3 items)
	N= 46
	Mean= .870
		N= 93
		Mean= 1.484
			Flipped Course Better (p<.001)

	SD= .875	SD= 1.023	

Computational Beliefs

The team explored the differences in students' confidence in their computational abilities, beliefs about the utility of computation, and intention for future study of computation between the two courses. Students completed a survey of nine questions regarding

- confidence with computation skills (e.g., "I am confident that I can successfully write a computer program.");
- perceived utility of computation (e.g., "I feel that the knowledge of computation will be useful in my studies."); and
- intentions of leveraging computation in the future (e.g., "I intend to use computation in my future career.").

A full list of questions is available in Appendix 1.

The only significant difference between the two courses was that students in the flipped course rated their confidence producing data visualizations higher ($p < 0.05$) and that the knowledge of computation will be useful in their careers at a higher level ($p < 0.1$). The higher beliefs in the utility of computation in careers for the flipped course may reflect the more contextualized and applied approach to how the course is taught. Complete results are shared in Table 4.

Table 3: Independent Samples T-Test for Computational Beliefs, Utility, and Intentions

Topic	Lec Mean	Lec SD	Flip Mean	Flip SD	t	df	p
Algorithm confidence	3.08	1.21	3.23	0.93	0.720	112.0	0.473
Programming confidence	3.67	0.90	3.63	0.93	-0.054	112.0	0.957
Data Visualization confidence	**2.25**	**1.16**	**2.78**	**1.04**	**2.206**	**110.0**	**0.029**
Useful in my studies	3.96	0.98	3.93	0.96	0.032	108.0	0.975
Useful in my professional development	3.92	0.76	4.04	0.95	0.838	105.0	0.404
Useful in my career	**3.58**	**1.26**	**4.02**	**1.02**	**1.902**	**104.0**	**0.060**
Intend to seek out computation courses	3.62	1.03	3.93	1.21	1.263	103.0	0.209
Intend to seek out comp prof development	3.79	0.82	3.96	1.12	0.876	103.0	0.383
Intend to use computation in my career	3.54	1.26	3.80	1.22	1.006	103.0	0.317

A deeper analysis of the results revealed that most of the differences between the courses was actually driven by one instructor. Table 5 shows the results for each of the three instructors who taught the flipped course. Students in Instructor 3's section scored significantly higher on all three measures of programming confidence, the usefulness of programming in future careers, and intention to seek out programming experiences in future computational courses and professional development experiences. It turns out this was likely not an instructor effect, but a student effect. Instructor 3's students were predominantly computer science majors. It would be expected they will rate themselves higher on programming confidence and intentions to use programming in the future. This provides evidence of the validity of the questions asked, however, the project team interprets the results as null finding for the overall impact of the flipped course compared to the traditional lecture course.

Table 4: Computational Beliefs Results

Concept	Instructor 1	Instructor 2	**Instructor 3**	**p-value**

Algorithm confidence	N= 23
	M= 2.96
	SD= .71
		N= 61
		M= 3.08
		SD= .80
			N= 25
			M= 3.84
			SD= .94
				< .001*

Programming confidence	N= 23
	M= 3.35
	SD=.71
		N= 61
		M= 3.12
		SD= 1.03
			N= 24
			M= 4.17
			SD= .87
				< .001*

Data Visualization confidence	N= 23
	M= 2.43
	SD= .90
		N= 61
		M= 2.67
		SD= .85
			N= 22
			M= 3.36
			SD= 1.26
				.004*

Useful in my studies	N= 23
	M= 3.83
	SD= .83
		N= 61
		M= 4.08
		SD=.86
			N= 20
			M= 4.05
			SD= 1.00
				.491

Useful in my professional development	N= 23
	M= 3.87
	SD= .81
		N= 61
		M= 4.16
		SD= .92
			N= 17
			M= 4.41
			SD= .71
				.142

Useful in my career	N= 23
	M= 3.78
	SD= 1.00
		N= 61
		M= 4.02
		SD= 1.00
			**N= 16
			M= 4.63
			SD= .50**
				.021*

Intend to seek out computation courses	N= 23
	M= 3.87
	SD= 1.01
		N= 61
		M= 3.28
		SD= 1.49
			**N= 15
			M= 4.5
			SD= .83**
				.003*

Intend to seek out comp prof development	N= 23
	M= 3.87
	SD= 1.01
		N= 61
		M= 3.30
		SD= 1.34
			**N= 15
			M= 4.6
			SD= .83**
				< .001*

Intend to use computation in my career	N= 23
	M= 3.57
	SD= 1.16
		N= 61
		M= 3.67
		SD= 1.29
			N= 15
			M= 4.27
			SD= .83
				.211

Student perceptions

This section reports key findings from the end-of-semester surveys, standard course evaluations, and focus groups. Responses rates for these data collection strategies are listed in Table 6. All activities were voluntary and did not contribute to the students' final grades. Response rates are higher for the end-of-semester course evaluations because grades are embargoed for two weeks after the semester unless a student completes the standard course evaluation. Grades are released immediately after completing the evaluation. This is standard practice for all courses in the engineering school.

Table 5: Responses Rates Surveys and Focus Groups

Data Source	**Traditional Lecture Course (67 students)**	**Flipped Course (144 students)**
End-of-semester Survey	46 responses (69%)	93 responses (65%)
Course evaluations	61 responses (91%)	143 responses (99%)
Focus Groups (2 conducted per course)	10 students	16 students

Student perceptions of the course reflected, not surprisingly, the structure of each course. Analyzing these differences helps to identify the strengths and weaknesses of each approach to inform future implementations pulling from both courses. Key insights came from recognizing common explanations about what students liked in one course and disliked in another. For example, the importance of purposefully structuring student-teacher interactions with an instructor was evident by the fact that traditional lecture students talked about the value of consulting with the instructor during office hours, but students in the flipped course felt the interaction with instructors during class could be more structured.

What did students like about the flipped course? On the course evaluation, students answered two open questions, "What are the best aspects of the course?" and "What are the worst aspects of the course?" Table 7 provides a list of the coded categories for what students rated as the best and worst aspects of the flipped course. Only the most frequently cited aspects will be discussed in this paper for brevity.

<u>Perceptions of Students in the Flipped Course: Positive</u>

The most frequently cited favorite aspect of the flipped course was the four challenge projects (n=36). Students articulated that the projects motivated them, pushed their thinking, and assessed their learning in a realistic way for a programming course. One student wrote, "The projects could be very challenging at times, but they were a good test of our understanding of the material." Another student shared a similar sentiment that was not uncommon. "The projects are interesting and push you to explore different solutions to somewhat complex problems."

The second most frequently listed aspect of the course was the content (n=26). Twenty-one students also listed the interactive textbook as a valued resource. "[The online textbook] is interactive and helpful with teaching concepts. It is very useful in studying for lab assessments." These perspectives were shared in the focus groups with students commenting how the content was well organized in the interactive textbook. They also felt the topics were clearly explained. One student, however, commented that the textbook was used primarily as a reference source and not the primary place for learning. While students enjoyed learning content from the interactive textbook, they did not enjoy the assessment tool associated with it as described below.

Students spent time during class working on coding examples. Twenty students listed this as one of the best aspects of the class. Some students liked coding in class. They felt it was helpful because it gave them authentic practices and was interesting. "Classwork and projects are very hands-on and interesting." Another student wrote, "in-class activities help to solidify concepts really well on a high level and to ensure that students can implement them." Several students said they liked that class was small and they worked in groups. "You have a small class so everyone is pretty close." It is important to note that classwork was also listed as one of the worst aspects of the course for reasons described below.

The teachers (n=17) and teaching assistants (n=16) were listed as one of the favorite aspects of the course. Words to describe the instructors and teaching assistants were "friendly," "knowledgeable," "helpful," and "encouraging."

Seventeen students specifically listed the course structure as the best aspect of the course with some describing it as the flipped approach. "I think the 'flipped classroom' is really effective; [the online textbook] is great and really helpful for learning basic knowledge about Java, and I think reading it and doing the problems ourselves is better than just listening to instructors talking about this in class." Eight students specifically mentioned they liked the self-paced aspect of the course. Students could schedule to take the weekly quiz at times convenient to them. Six students said they liked the pacing of the course/workload. Some of the comments about more specific aspects of the course (interactive textbook, classwork) listed above provide additional examples of why students liked the course structure.

Perceptions of Students in the Flipped Course: Negative

While some students liked the course structure, others did not. This is not uncommon in that students' perceptions of the flipped classroom can be mixed [9]. There were fewer negative comments captured by the question "What were the worst aspects of the course?" (83 responses) than positive responses (121 responses). However, the intensity of negative responses was deeper based on the length of comments. Some of these comments ran multiple paragraphs.

Table 7 lists all the categories used to code the negative comments in the flipped course. The biggest complaint was the requirement to complete work independently outside of class time (n=29). This is not surprising. The instructional design team supporting this project has helped several departments flip courses, and one of the most frequent student complaints is completing work outside of class time. The team plans to explore this perception of workload more in future studies. Related to this perception, 20 students mentioned increased workload as the worst aspect of the course in addition to the 29 mentions of independent learning outside of class. For the flipped course, the weekly quiz and challenge problems required to be taken at a computer lab outside of scheduled class time was listed as a burden. The faculty wanted students to have the flexibility to take these assessments once they felt prepared. While eight students said this self-paced aspect of the course was beneficial, many more students did not. They considered it a fourth class session with some students arguing that the course should be increased to four credits for this reason.

- "It was sometimes difficult to schedule the weekly assessment and there is so much out-of-class learning that I never had time to complete the in-class exercises that we never finished outside of lecture."
- "Needing to fit another hour of weekly tests into my schedule. This was said to be a three credit course, but there are four hours a week that you need to be with your professor or t.a."

Student complaints about the challenge problems focused on the inflexibility of the auto-grading platform associated with the interactive textbook. The auto grading tool would assign a zero for minor errors in an otherwise well-constructed program. A student in a focus group gave this example. "I had this one test where all my codes were getting the correct numbers except the last two decimal places out of 12 decimal places were wrong for all my numbers and it turned out it was because, instead of saying to the power, I was multiplying them together, which should be the same mathematically but … it ended up in a rounding error in the code." Comments from the course evaluation included the following.

- "The auto-graded quizzes are perhaps the single most frustrating assessment tool I've ever encountered in my academic career. There have been numerous instances where I solved the problem and my output matched the answer output almost exactly, yet due to one extremely minor elusive detail, resulted in awful grades on an assessment that I otherwise completed 95% correctly."

- "Most of our assessments were all or nothing. If your code didn't compile, that is a shame. You get a 0, even if you have the logic for everything else down correctly."

While students didn't like the strict auto-grading rubric, some students said they preferred weekly quizzes to periodic mid-terms because it reduced the stress that comes with high-stakes evaluations. "Having the weekly module quizzes take the place of three or four midterms. I felt like [that] took a lot of the stress off of the main portion of the class because I wasn't worried about having a big chunk of my grade represented by a single midterm."

As noted above, some students mentioned classwork as the best aspect of the course. Almost an equal number of students mentioned it was the worst aspect of the course (n=20). Students felt the facilitation could be more purposefully structured, and in some cases, implied they were hesitant to ask for help. This hesitance originated with the flipped format. *Students worried if they asked questions the instructor might think they not did complete the pre-work even if the student had done so.* While some students felt the classwork was engaging (see previous section), others felt it was too challenging for the 50-minute class meetings. Some students were also frustrated that there was no incentive to attend class. Problems completed during class were not graded, and as noted above, students did not attempt to complete the problems if they were not finished during class.

- "Given how difficult it is to learn outside of class and how the curriculum itself doesn't prompt the class's teachers or TAs to actually lecture, students would sit through class painfully as they struggle between trying to figure out how to do the day's assignment without the tools to complete it, mustering up the courage to either ask the teacher or T.A. about something that they 'should have seen in the pre-lecture material.'"
- "I'm not really a fan of the flipped classroom learning. I don't think it's very effective because half the time we only get thorough half a question of the three that are put up each class."
- "I think a little more should have been gone over in class. It seemed like it was assumed that we understood the material perfectly before we were asked to do it in class. For some of the harder subjects it would have been nice if we had a period at the beginning of the week to discuss questions."
- "Classwork is often way too long and never completed."

Perceptions of Students in the Traditional Course: Positive

For the traditional lecture course, students listed the instructor most frequently as one of the best aspects of the course (n=18). Students described the faculty member as "extremely helpful" and "explains things clearly." "Helpful" was the most frequently used word to describe why the instructor was the best aspect of this course. It appears this may reflect the instructor's availability outside of class as much as the help provided during class. Students listed teaching assistants as one of the best aspects of the course 11 times for similar reasons. *This suggests students place a high priority on the value of developing a relationship with the instructor and* *the need for help clarifying difficult concepts.* While the flipped course provided this opportunity for feedback during class, some students in the lecture course actively sought the same help through office hours.

Table 6: Coded Student Comments on Flipped Course Evaluations

Best Aspect of the Course	**Worst Aspect of the Course**
Projects (n=36)	Class Structure (n=35)
Topics (n=25)	Working Independently Beyond Class Time (n=29)
Interactive Textbook (n=21)	Classwork (n=21)
Classwork (n=20)	Pace/Workload (n=20)
Class Structure (n=17)	Weekly Quizzes (n=18)
Teacher (n=17)	No Incentives to Attend Class (n=13)
Teaching Assistants (n=16)	Curriculum Misalignment (n=9)
Self-paced Course (n=8)	Self-paced Course (n=9)
Video Lectures (n=6)	Interactive Textbook (n=7)
Pace/Workload (n=6)	Video Lecture (n=7)
Nothing (n=5)	Not sharing solutions/Feedback (n=7)
Discussion Board (n=4)	Inflexible Auto-grading (n=5)
Weekly Quizzes (n=3)	Teacher (n=5)
	Projects (n=3)
	Topics (n=3)
	Nothing (n=2)
	Final Exam Format (n=2)
	Teaching Assistants (n=2)

Learning how to program JAVA was the second most-frequently mentioned item as the best aspect of the course (n=12). Like the flipped course, the projects were listed as one of the more popular aspects of the course (n=11). Homework was listed 11 times as well. The flipped and lecture courses did not use the exact the same projects and homework, but they did assess similar objectives. *The student responses from both courses suggest they like opportunities to apply course concepts to contextualized engineering problems that push them to extend their thinking.* In the focus groups, students shared that homework and projects contributed to their learning. “I definitely think the homework is the most important part though. Just being able to take what you've learned - the theory of and actually implementing it - is like the only way to really learn it.” One student in the flipped course focus group described how the homework combined with support through office hours was important. “One of the biggest aspect of the course that helped me was the [course assistant] office hours, … it helped me a lot with homeworks, and the homeworks helped me a lot, like just understanding the material.”

Perceptions of Students in the Traditional Course: Negative

Students’ most frequent complaint about the traditional course was the grading and feedback (n=13). Students wanted the assignments returned sooner and the instructor and TAs to answer emails more quickly. One student mentioned the grading was “harsh,” but in general the promptness of feedback on graded work was the biggest reason for listing feedback as the worst aspect of the course. Again, this supports the importance students place on consulting the instructors and TA for help.

The next most frequent complaint was class time (n=7). Students felt lectures were sometimes "ineffective" and that it "Gets boring in class." This may have been impacted by the fact that class meetings were 75 minutes. "Lectures feel extra long because they're over an hour long," wrote one student.

Despite these complaints "Nothing" or "N/A" was listed 7 times as the worst aspect of the course.

Like the flipped course, students did not like taking the final exam with pen and paper (n=6). They wanted to be tested in an environment similar to how they coded other assignments. Specifically, they wanted a debugger to check their work. Students in the focus groups shared similar concerns. "I personally thought having to write down my code was not the most practical way to do the exams, I guess, because in the real world everyone is gonna write code on the computer." On the course evaluations, students commented that they understood why this format was used – to minimize cheating – but felt it was a challenge that should be addressed. One student in the focus group recognized it may reflect a practice used in job interviews. "The thing that I have about that is like in in-person interviews, I heard that they ask you to write code in front of them sometimes, like on a whiteboard. So in terms of that I think it was just like, like even though in an actual job you probably wouldn't have to do that, it's still something that you might have to do in terms of like interviewing, stuff like that." It may be useful to explain this to students so they understand the rationale for using pen-and-paper exams.

Table 8 provides a complete list of the coded categories for what students rated as the best and worst aspects of the lecture course.

Table 7: Coded Student Comments on Traditional Lecture Course Evaluations

Best Aspect of the Course	**Worst Aspect of the Course**
Instructor (n=18)	Grading/Feedback (n=13)
Learning to Program in JAVA (n=12)	Classwork/Lecture (n=7)
Teaching Assistants (n=11)	Nothing (n = 7)
Projects (n=11)	Final Exam Format (n=6)
Homework (n=11)	Pace/Workload (n=5)
Pace/Workload (n=4)	Teacher (n=5)
Feedback (n=4)	Course Materials (n=3)
Course Structure (n=4)	Topics (n=1)
Clicker Questions (n=1)	Homework (n=1)
Lectures (n=1)	Discussion Board (n=1)
Class Materials (n=1)	
Gradescope (n=1)	
Discussion Board (n=1)	
Nothing (n=1)	

Discussion

The curriculum design team analyzed the data to identify changes to future implementations of the course. Only general interpretations and recommendations that are relevant to a broader audience are shared here.

Lessons Learned

It is clear *students value the role of the instructor*. Students in both classes commented on the importance of the faculty member to motivate and help them. In the traditional lecture course, students described the importance of the faculty member in explaining difficult concepts, however, this was often done through office hours outside of class. Students in the flipped course also consulted faculty and teaching assistants through office hours, however, they expressed the need for more structured engagement as they completed classwork. Formative assessment techniques during class can provide feedback to the instructor during class to know when to help individual groups or when to debrief to the whole class.

While active-learning has been shown to be more valuable than lecture in previous research [4] [5], *there is still a role for lecture during class*. Students in the flipped course requested that the instructor start each class with a review of the key concepts presented in the pre-work. One student in a focus group session said, "We did get videos, but I felt like it would be better if there was something inside the class like lecture like ten, fifteen minutes in beginning to go through the topics that are going to come in the following week."

There were two other findings that are worth sharing. First, *students may prefer a lecture approach more than anticipated*. Comments from students in the traditional lecture suggest they could not envision other ways of learning computer science. One student in the focus groups said, "I won't say there's another way, as if like a better way to be taught [than traditional lecture]. Sometimes it just feels like in terms of programming courses it is probably one of the only ways that it can be taught." Students limited exposure to student-centric pedagogies may limit their understanding of how courses could be taught differently than lecture or their openness to learning through a flipped method.

Second, *students in a lecture-section focus group shared they would prefer not to learn through a flipped course*. They felt watching lectures outside of class was not as impactful as watching them during class. Here is an exchange in the focus group between several students that illustrates this point.

> Student 1: If you're watching the lectures outside of class, I think I put less energy into 'em.
>
> Students 2: Yeah, I've never come out of a flipped classroom experience feeling like I understood or mastered the material.
>
> Student 3: Yeah, I'd say, the professor usually does a better job of teaching than the videos that we have to watch

> Student 2: It's definitely easier to pay attention to a lecturer when the person's right in front of you, and maybe that's a me problem, but I get really distracted by computer screens.
>
> Student 1: Yeah, I agree with that.
>
> Students 2: Right. 'Cause we're busy, and also, I always end up, like, watching those lectures is like the last thing I do before I go to sleep - So I'm tired and I've had a long day. Whereas, if you actually do it during class time, it's earlier in the day, before you're stressed out and fatigued. But I come out of every traditional, well not every, but, almost all of traditional lectures, I feel like I actually understood and learned something.

This leads to another observation. *Students can be intimidated to ask for help in a flipped environment because of the expectation that they learned the material through pre-work.* Students shared they were worried that if they asked questions, the instructor would think they had not completed the pre-work even if they had. Students generally rated the faculty as approachable and accessible. But it is important to recognize that faculty-student power hierarchies can cause students to hesitate to ask for help if they think it will signal they have not done the pre-work. Some students reported consulting teaching assistants during office hours because they did not want to ask questions during class. This provided them another outlet to clarify concepts, but it would be better if they asked questions during class. Prompt feedback is important to helping students learn [15].

Another challenge for class time was providing students incentives to attend. In the flipped course, students weren't graded on the work they completed during class. It was meant to be a low-stakes opportunity to apply key concepts in computing. The unintentional outcome was that some students did not feel it was worth coming to class, and if they did not finish the challenge problems during class, they did not bother doing so after class. This was amplified by the fact that students in the flipped course felt overwhelmed with the work required outside of class. The biggest complaint was scheduling 50 minutes to take a quiz outside of class. The team originally thought students would appreciate the flexibility to schedule this evaluation when they felt ready. Some students did comment they liked this self-paced aspect of the course. Many more students, however, felt it was essentially a fourth hour of class each week.

Proposed Future Changes

The redesign team made several changes to address the concerns about more structured faculty support and the workload required outside of class. First, the weekly quiz will be conducted during class on Friday to reduce the workload outside of class. Second, faculty will provide more instruction during class time not only explaining difficult concepts but also helping students recognize how concepts are related. Solutions to in-class activities will be posted so students who do not finish them during class can check their work when they are finished. The team will also implement new methods for addressing concerns about auto-graded quizzes. Assessments assigned a zero will automatically be checked by a teaching assistant. Non-zero submissions will be randomly spot checked to identify potential auto-grading problems.

Despite the challenges to the pilot implementation of the flipped course, there is enough evidence for the team to continue the pilot. The most important revelation was that students enjoy being challenged by projects and homework and recognize they were some of the most effective ways to learn the content. Some students reported the assignments helped them learn more than how to program; they also learned critical thinking skills. "…teaching yourself language is usually a very valuable lesson you can do. It's more like problem solving skills and those types of things will transfer over [to other courses]." Therefore, there is reason to believe that improving the facilitation of in-class challenge problems will help students learn the concepts better. Faculty will more purposefully facilitate in-class activities. They will also provide solutions to allow students to check their work.

Conclusion

This paper describes a comparative analysis of introductory JAVA courses – one taught with traditional lecture and another using a flipped course approach. While there were no significant learning differences measured between the courses, this likely reflected challenges to the implementation of the CS1 concept inventory. There was limited evidence that the flipped course contributed to improved student confidence with some skills (data visualization) and usefulness of computing compared. Upon deeper investigation it appears this result reflects the perspectives of computer science majors embedded in the flipped course sections. The team will investigate this more in the future. The paper also reported what students liked and disliked about each approach through various data collection methods.

There are some limitations of the study that should be considered by other faculty who may be interested in using the results to inform their own teaching. First, students were not randomly assigned to each course. Second, each course was taught by a different instructor – with three instructors facilitating the eights sections of the flipped course. Third, students were assessed differently in each course so the application of course content was different in each class, however, these choices reflect the pedagogical design of the course.

Another potential limitation is the history of the course. Introductory JAVA has been taught with a lecture method for years. The instructor of that course had experience teaching with the lecture method and leveraged existing course materials. The flipped course instructors had to create new course materials during the previous summer and learn how to integrate the interactive textbook into the curriculum while also experimenting with new active-learning teaching strategies. As the instructors gain more experience teaching in this new way, the impact on student learning and perspectives may change.

It is also worth considering that some students may not have been open to learning in the flipped environment. Some students in the lecture course focus group expressed unprompted criticism of the flipped method. Normative expectations of how course are taught may influence student outcomes, and will be explored more in the future.

Even with these limitations, the authors hope these findings are useful to other instructors teaching introductory or gateway courses.

References

[1] National Academies of Sciences, Engineering, and Medicine, and National Academies of Sciences, Engineering, and Medicine. *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments.* Washington, DC: The National Academies Press, 2017.

[2] J.B. Adams. “Computational science as a twenty-first century discipline in the liberal arts.” *Journal of Computing Sciences in College*, vol. *23*, no. 5, pp: 15–23. 2008.

[3] L.K. Soh, A. Samal, S. Scott, S. Ramsay, E. Moriyama, G. Meyer, B. Moore, W.G. Thomas, and D.F. Shell. “Renaissance computing: An initiative for promoting student participation in computing.”w *SIGCSE Bulletin*, vol. 41, no. 1, pp: 59–63. 2009.

[4] M. Prince. “Does active learning work? A review of the research.” *Journal of Engineering Education*, vol. 93, no. 3, pp: 223-231. 2004.

[5] National Research Council. *How People Learn: Bridging Research and Practice*. Washington D.C.: National Academies Press, 1999.

[6] C.M. Viera, A.J. Magana, A.M. Roy, M.L. Falk, M.J. Reese Jr. “Exploring undergraduate students’ computational literacy in the context of problem solving.” *The ASEE Computers in Education Journal,* vol. 7, no. 1, pp: 100-112. 2016.

[7] A.J. Magana, M.L. Falk, M.J. Reese Jr. “Introducing discipline-based computing in undergraduate engineering education.” *ACM Transactions on Computing Education,* vol. 13, no. 4, pp: 16-24. 2013.

[8] A.J. Magana, M.L. Falk, C. Vieira, M.J. Reese. “A case study of undergraduate engineering students' computational literacy and self-beliefs about computing in the context of authentic practices.” *Computers in Human Behavior*, vol. 61, pp: 427-442. 2016.

[9] J.L. Bishop, J. Lowell, M. A. Verleger. "The flipped classroom: A survey of the research." In *Proceedings of the 120th American Society of Engineering Education Annual Conference and Exposition, Atlanta, GA*, vol. 30, no. 9, pp. 1-18. 2013.

[10] J. Bergmann and A. Sams. *Flip Your Classroom: Reach Every Student in Every Class Every Day*. International Society for Technology in Education. 2012.

[11] F.D. Davis. "User acceptance of information technology: system characteristics, user perceptions and behavioral impacts." *International Journal of Man-Machine Studies,* vol. 38, no. 3, pp. 475-487. 1993.

[12] A.J. Magana, M.L. Falk, M.J. Reese Jr. C. Vieira. "Materials science students’ perceptions and usage intentions of computation." In *Proceedings of the 120th American Society of Engineering Education Annual Conference and Exposition, Atlanta, GA*. 2013.

[13] A.E. Tew and M. Guzdial. "Developing a validated assessment of fundamental CS1 concepts," in *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, Milwaukee, WI, USA, Mar 10-13, 2010, G. Lewandowski, S. Wolfman, T. J. Cortina, E. L. Walker, and D. R. Musicant, Eds. ACM, 2010. pp. 97-101.

[14] J.M. Corbin and A.L. Strauss, *Basics of qualitative research: Techniques and procedures for developing grounded theory*, Sage, 2008.

[15] S. A. Ambrose, M. W. Bridges, M. DiPietro, M. C. Lovett, and M. K. Norman. *How Learning Works: Seven Research-based Principles for Smart Teaching*. San Francisco, CA: John Wiley & Sons, 2010.

Appendix 1 – Computational Beliefs, Utility, and Intention Survey

Please select the option that best represents how you feel for each statement.

I am confident that I can successfully design an algorithm.

Not at all confident - Not so confident - Somewhat confident - Very confident - Extremely confident

I am confident that I can successfully write a computer program.

Not at all confident - Not so confident - Somewhat confident - Very confident - Extremely confident

I am confident that I can successfully produce data visualizations (transform data into visualizations).

Not at all confident - Not so confident - Somewhat confident - Very confident - Extremely confident

I feel that the knowledge of computation (e.g., algorithm design, modeling and simulation, data visualization) will be useful in my studies.

Not at all useful - Not so useful - Somewhat useful - Very useful - Extremely useful

I feel that the knowledge of computation (e.g., algorithm design, modeling and simulation, data visualization) will be useful for my professional development.

Not at all useful - Not so useful - Somewhat useful - Very useful - Extremely useful

I feel that the knowledge of computation (e.g., algorithm design, modeling and simulation, data visualization) will be useful for my career.

Not at all useful - Not so useful - Somewhat useful - Very useful - Extremely useful

I intend to purposefully seek courses that will allow me to increase my knowledge about computation (e.g., algorithm design, modeling and simulation, data visualizations).

Strongly disagree – Disagree – Neither agree or disagree – Agree – Strongly agree

I intend to purposefully seek opportunities and resources that will allow me to increase my knowledge about computation (e.g., algorithm design, modeling and simulation, data visualizations).

Strongly disagree – Disagree – Neither agree or disagree – Agree – Strongly agree

I intend to use computation (e.g., algorithm design, modeling and simulation, data visualizations) in my future career.

Strongly disagree – Disagree – Neither agree or disagree – Agree – Strongly agree