Learners' Differences in Blended Learner-Centric Approach for a Common Programming Subject

Qi Cao*, Chee Kiat Seow, Li Hong Idris Lim, Sye Loong Keoh, Vicki Dale, Sarah Honeychurch, Nathalie Tasler, and Duncan Bremner

***Abstract*—As the number of students entering higher education increases with a growing diversity of background, educators of programming courses face increasing challenges. Different teaching pedagogies need to be explored for students with different background knowledge. Some students find programming courses difficult to understand and practice. It may lead to de-motivation and disengagement in learning process with consequential impact on their grades. Addressing these issues demands approaches for effective teaching programming courses to multidisciplinary cohorts. This article investigates how computing science (CS) and engineering cohorts respond differently to teaching approaches in a common module, Fundamentals of Programming. Both traditional teacher-centric teaching and a blended learner-centric approach have been explored in a diverse group of students. The blended learner-centric approach combines classroom teaching and self-paced blended learning using work examples videos method. These two teaching approaches have been evaluated in Academic Year 2019/2020. It can be seen from the evaluation results of 92 CS and 150 Engineering students who participate in this research that the performance is improved by about 5% through blended learner-centric approach. It is further observed that quantitatively the performance gap between CS and Engineering students has been reduced. Questionnaire survey has also been conducted with 54 CS and 89 Engineering students being responded. The learners' perceptions of the blended learner-centric approach have also been compared between these two cohorts.**

***Index Terms*—Diverse learners, learners' difference, learners perceptions, learning of programming**

I. Introduction

Software programming skills have become key for 21st century students [1]. Software programming courses in computing science (CS) have become essential courses to students in multiple fields. In the science, technology, engineering, and mathematics (STEM) areas, the software programming education is no longer just for computing science (CS) students, but also for other majors, who have a wide spectrum of prior background knowledge and programming experiences [2]. Some CS fundamental subjects have become popular options for non-CS majors [3]. It is an interesting topic of how to manage the diversity of learner differences in teaching programming subjects.

Teaching programming subjects to first year students is a difficult task in higher education [4], with many challenges encountered by lecturers [3], as large cohorts may exhibit a wide spectrum of prior programming experience, background, and interests [5]. On one end of the spectrum, some students have minimum prior programming skills, while on the other end, some of them already have substantial hands-on programming experience [6]. This diversity brings difficulties in teaching programming courses in higher education institutions (HEIs), as course contents must cater for all students [3, 7].

The fundamentals of programming (FoP) subject which provides basic knowledge and coding training to students is an introductory programming subject in HEIs [8]. If students can excel in the introductory programming subjects, it may prevent students from switching to other majors [9] and reduce the dropout rate. However, learning programming skills can be challenging. Robins *et al*. [10], further supported by Grover and Pea [11], suggest that much of the challenges are the acquisition of Computational Thinking skills. As a fundamental competence, Computational Thinking skills include formulating problems, representing data, analysing data, implementing solutions [12]. Many students feel programming is not easy to learn [13], and as a subject it is hard to score good grades [14]. Some students even perceive software programming as one of their hardest subjects [6, 15, 16]. It results in a high dropout or failure rates, as compared to other subjects [17–21], which could be up to 30-50% [20, 21]. Several potential reasons include variations in reference materials, lectures delivery, lab session approaches, problem solving ability, time management, students' characteristics, and motivation to study programming [4, 14]. Another reason is large number of students present in single programming labs [6, 22].

Prior programming knowledge of students could be a possible predictor of success in programming courses [6, 23]; it can trigger better motivation and confidence levels to achieve learning goals [5]. Students with prior hands-on programming experience have a more positive attitude and lower chances of dropping out, as compared with students without prior programming knowledge [9]. Students from non-CS backgrounds may have difficulty in applying theorical knowledge to real world programming problems [24, 25].

Various approaches have been explored to improve teaching of programming courses, with different levels of curriculum for students with less prior experience [5]. Separate optional introductory courses have been offered to students with less experience [26, 27]. Some redesigned introductory course content is provided to students with less

Manuscript received October 9, 2022; revised November 15, 2022; accepted December 12, 2022.

Qi Cao, Chee Kiat Seow, and Sye Loong Keoh are with School of Computing Science, University of Glasgow, UK.

Li Hong Idris Lim is with School of Engineering, University of Glasgow, UK.

Vicki Dale, Sarah Honeychurch, and Nathalie Tasler are with Academic and Digital Development, University of Glasgow, UK.

Duncan Bremner is with James Watt School of Engineering, University of Glasgow, UK.

*Correspondence: qi.cao@Glasgow.ac.uk (Q.C.)

doi: 10.18178/ijiet.2023.13.6.1886

prior experience [28]. A computing approach with digital media materials may reduce failure rates for students without prior experiences [16]. The value of a collaborative learning approach is reported in [22, 29], for students to help each other in group assignments through discussions and teamworking. A learner-centred approach is another possible method for teaching programming courses [30].

Existing literature suggests that blended learning is effective in improving the student learning experience [31, 32]. Blended learning, also known as hybrid learning, is explored in teaching introductory programming courses [33, 34], including online collaboration learning model, and online self-paced learning models, etc. [4]. Online self-paced learning models provide flexibility to students, at different locations and time. According to students' feedback, blended learning better enables them to understand programming concepts [35]. Blended learning results in improved outcomes compared with traditional teacher-centric learning method in programming courses [4].

It is interesting to explore the teacher-centric teaching approach and a blended learner-centric approach in the common subject, Fundamentals of Programming (FoP) for CS and Engineering students. It is particularly interesting to investigate what extent does a blended learner-centric approach benefit students majoring in CS versus Engineering in the programming subject. It is also worth to study what extent do learners' perceptions of the blended learner-centric approach differ between CS and Engineering cohorts when learning programming courses. In this paper, the comparisons of the differences in learning a common subject, FoP between Year 1 CS and Engineering students are investigated with the blended learner-centric approach. To the best of our knowledge, this is the first work comparing differences between CS and Engineering cohorts, in their response to a blended learner-centric approach on a common programming subject in HEIs.

The University of Glasgow, Singapore (UGS) partners with Singapore Institute of Technology (SIT) to offer a joint degree with honours programmes. The teaching approach of the University of Glasgow emphasises active learning and independence in learning [36], which are retained in the offering of degree programmes in Singapore.

FoP is a common module for students of CS, Mechanical Engineering (MEC), and Aerospace Engineering (AEE). It is taught in the first trimester of Year 1. Students have the options to opt for exemption from this subject, if they can show their syllabus and transcripts with results higher than Grade B in their prior study. The content of this subject is designed to be taken by any student interested in acquiring programming skills. The topics are covered in this subject as follows:

1) Introduction to social context of computing.
2) Concepts and knowledge in programming, such as data types, control structures, functions, arrays, pointers, and files input/output.
3) Secure coding, such as input validation, data sanitization, and buffer overflows.
4) Compiling, running, debugging, and testing programs.
5) Overview of programming paradigms.

The programming concepts are practiced and demonstrated in standard programming languages, i.e., C and C++. In each week, the contact hours include two-hour lectures, and a one-hour lab session. The lab session involves hands-on practice and skill-based learning activities with specialised software. Performance of students is assessed by a final exam, as well as continuous assessments (CA) including lab exercises and project assignments. The weighting between the final exam and CA is 60%:40%. Details of the case study and performance evaluations will be discussed in this paper.

II. Structure of Assignment Projects

In this subject, there were two project assignments as the continuous assessments (CA), to assess students' hands-on programming skills by solving real-world problems. Considering the projects' difficulty, students were divided into small groups and solve the problems through teamwork. Each group was required to submit a final solution report with all source codes attached.

In the first project assignment, students were asked to develop a C program to perform a linear regression analysis on a given dataset (10,000 data pairs) with certain noise. Compared to the first one, the second project assignment was more difficult in terms of scopes and deliverables. It required students to not only perform analyses on noise probability and obtain random noise statistics in a large noisy dataset, but also compute probability density function for this dataset.

To compare how CS and Engineering cohorts responded differently to two alternative learning approaches, there were two project assignments being conducted in this FoP subject. The traditional teacher-centric teaching approach was used for the first assignment, its project specifications, scope, and deliverables explained to all students in a face-to-face (F2F) lecture. Before the F2F lecture, the project assignment documents were shared online for students' pre-reading through the learning management system (LMS). Students could follow instructions in these project documents and work in groups. A F2F Q&A session was also provided to CS and Engineering students separately, with a half hour duration.

The blended learner-centric approach was adopted for the second project assignment. Besides the traditional teacher-centric F2F lecture session and project documents shared online via the LMS, worked example videos as complementary materials were also provided. Worked example videos illustrated step-by-step problem-solving processes for students [37]. In the second project assignment, the worked example videos were recorded using an iPad with a pen. Students could watch the complementary worked example videos repeatedly in their own time. One F2F Q&A session was also offered to CS and Engineering students separately, with a half hour duration.

For this FoP subject, 92 students from CS, and 150 students from Engineering (both MEC and AEE) were enrolled in AY2019/2020. These students came from diverse backgrounds in terms of their programming knowledge and skills. The spectrum distributions of prior programming experiences of CS cohort and Engineering cohort were different.

One of the admission criteria to the School of CS was that applicants had been learned a few programming courses in their prior study, evidenced on their transcripts. Most CS students graduated from an Information Technology (IT) related Diploma in Singapore polytechnics with prior programming knowledge. Some CS students even had few years' working experience in the IT industry with good hands-on programming skills. Only a small percentage of CS students who graduated from a non-IT related Diploma did not learn programming courses previously, but they usually obtained a high GPA in their prior study illustrating good learning capabilities, with some programming theory knowledge by self-study.

In contrast, most Engineering (both MEC and AEE) students did not learn programming courses when they matriculated into the joint degree programmes, according to their transcripts and syllabus at the admission application stage. However, there was a small percentage of Engineering students having gained programming knowledge through previous self-study, indicated by these students in lectures and tutorials' discussions of the FoP subject.

Teamwork in group projects was very important to a successful project. The grouping of students was conducted as the voluntary basis, that students formed their grouping with whom they were comfortable to work. Due to this consideration, the students with prior programming knowledge were not specifically distributed into different groups. CS students were divided into groups of three. Engineering students typically worked in a larger group size, as their subjects often involved use of electronic hardware or mechanical components. Engineering students requested to be divided into groups of four for the assignments, as they had less prior programming knowledge than the CS cohort, allowing more of a collective effort and manpower to apply programming theory in solving real-world problems. It could increase their confidence level to complete the project assignments on time. The group size is suggested to be four or five that tends to work better [38, 39].

In view of the learners' differences between the CS and Engineering cohorts, the requests of Engineering students were considered reasonable. In response, 92 CS students were divided into 28 groups of three and two groups of four while 150 Engineering students were divided into 36 groups of four members, and two groups of three members. The groupings were kept the same for both project assignments.

In a quick summary, the first assignment was less difficult with the traditional teacher-centric teaching approach, while the second assignment was less difficult with the blended learner-centric approach. Students were given three weeks to complete each assignment. The first assignment was released in Week 5 in the semester, the second in Week 9, to give students enough exposure to the course content before the assignments were given. At the end of the semester, an evaluation was conducted for comparison analysis from the perspectives among learners with different backgrounds, i.e., between the CS cohort and the Engineering cohort.

III. Results Analysis

In this section, the results analysis is presented, with quick summary as follows:

1) Performance comparisons between CS and Engineering cohorts were conducted taking a common programming subject. Results showed a consistent performance improvement with about 5% by the blended learner-centric approach for both cohorts. The performance gap between these two cohorts comparably had been reduced.
2) Feedback from CS and Engineering students had been collected through the survey questionnaire with about 59% response rate (54 CS students and 89 Engineering students). Detailed analysis had been conducted on learners' perceptions of the blended learner-centric approach, revealing key behavioural differences between the two cohorts.
   - CS students reported higher acceptance to learning programming with the blended learner-centric approach and showed a higher appreciation for the worked example videos that covered profound mathematical concepts and the detailed programming steps. On the other hand, it was found that Engineering students were less appreciative of them and more likely to have struggled with watching them, although some of them watched the videos repeatedly to gain better understanding.
   - Although more time was needed for independent learning with the blended learner-centric approach, Engineering students preferred to learning contents consisting of a high proportion of worked example videos at about 60%–80% versus F2F lecturing for programming courses, illustrating that more Engineering students realised the benefits of blended learner-centric approach on programming courses.

A. Academic Performance Comparisons

The mean score differences between CS and Engineering cohorts for these two project assignments are shown in Table I. It is observed that the mean score of Assignment 2 was higher by about 4.35% than that of Assignment 1 for the CS students. The mean value of Assignment 2 was higher by about 5.2% than that of Assignment 1 for the Engineering students. Comparing the CS and Engineering cohorts, it is observed that the mean score of CS students was higher by about 1.55% than that of Engineering students with the traditional teacher-centric teaching approach in Assignment 1. The mean score of CS students was higher by about 0.74% than that of Engineering students after the adoption of the blended learner-centric approach in Assignment 2. It is observed that both CS and Engineering cohorts benefited from the blended learner-centric approach in Assignment 2, and that the performance gap between the CS and Engineering cohorts had been reduced after the adoption of the blended learner-centric approach.

TABLE I: Results Comparisons between Two Cohorts

Average Marks	Computing Science (92 students)	Engineering (150 students)	Difference
Assignment 1	73.42 marks	72.30 marks	1.55%
Assignment 2	76.62 marks	76.06 marks	0.74%
Improvement	4.35%	5.20%	

The t-test was used to evaluate if there was a statistically significant difference of the mean values of CS and Engineering cohorts. For the t-test, the value of the alpha level was set as 0.05. The results of the t-test between Assignment 1 and Assignment 2 of the CS cohort are shown in Table II, representing improvements after adopting the blended learner-centric approach. The P-value of the t-test (i.e., P << α level) indicated that there was a statistically significant difference between the mean scores of CS students in Assignment 1 and Assignment 2, showing significant improvement with the blended learner-centric approach.

TABLE II: T-TEST FOR CS STUDENTS IN ASSIGNMENT 1 AND 2

	Assignment 1 of CS Students	Assignment 2 of CS Students
Mean scores	73.41	76.62
Variance	31.06	28.92
Observations	92	92
P value (at α = 0.05; two-tail)	1.46E−10	

The results of the t-test between Assignment 1 and Assignment 2 of the Engineering cohort are shown in Table III, i.e., before and after adopting the blended learner-centric approach. The P-value of the t-test (i.e., P << α level) indicated a statistically significant difference between the mean scores of Engineering students in Assignment 1 and Assignment 2, illustrating significant improvement.

TABLE III: T-TEST FOR ENGINEERING STUDENTS IN ASSIGNMENT 1 AND 2

	Assignment 1 of Eng. Students	Assignment 2 of Eng. Students
Mean scores	72.30	76.06
Variance	27.77	30.12
Observations	150	150
P value (at α = 0.05; two-tail)	4.53E−15	

The results of the t-test between CS and Engineering cohorts comparisons for Assignment 2 are shown in Table IV. The P-value of the t-test (i.e., P = 0.44) is much larger than the alpha level (i.e., α = 0.05). Hence, there was no statistically significant difference between the mean scores of CS and Engineering cohorts in Assignment 2. The difference was small, although CS students obtained slightly higher mean scores with the blended learner-centric approach.

TABLE IV: T-TEST FOR TWO COHORTS IN ASSIGNMENT 2

	Assignment 2 of CS Students	Assignment 2 of Engineering Students
Mean scores	76.62	76.06
Variance	28.92	30.12
Observations	92	150
P value (at α = 0.05; two-tail)	0.44	

B. *Comparison of Learners' Perceptions*

A survey was conducted after the last lecture of the FoP subject. All students enrolled in this subject (92 CS students and 150 Engineering students) were invited to participate. It was made known to the students that their responses were fully anonymous, and that completion of the survey was voluntary, and that participation would not have any impact on their academic results. We designed the survey questions, consisting of nine major questions that will be illustrated next in this section. The response rates of CS and Engineering students were about 59%, with 54 CS students and 89 Engineering students as shown in Table V.

TABLE V: SURVEY RESPONSE RATES

	No. of Enrolled Students	No. of Responses	Response Rate
CS	92	54	59%
Engineering	150	89	59%

The perceptions on blended learner-centric approach between the CS and Engineering cohorts had been explored. Students' feedback had been collected through a survey. The results of the survey questionnaires are discussed next.

1) *Q1: Acceptance of blended learner-centric approach*

The feedback comparisons of the two cohorts for Question 1 in the survey are shown in Fig. 1. It is observed that CS students were more comfortable with the blended learner-centric approach than Engineering students, as the mean value of CS students was 4.11, while that of Engineering students was 3.64. It has the positive impact to better academic performance of CS students achieved.

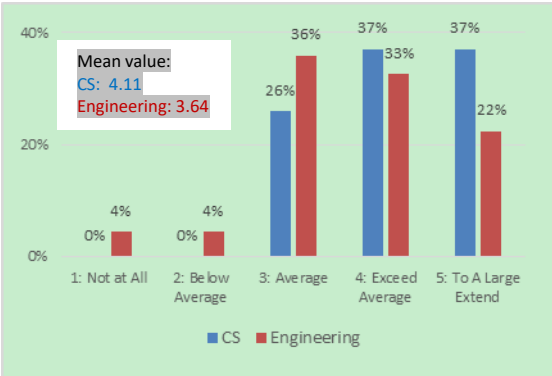

Fig. 1. Comparison on Question 1: *To what extent you like blended learner-centric approach and would you recommend it?*

2) *Q2: Acceptance of worked example videos*

For Question 2 in the survey, the feedback from the two cohorts is compared in Fig. 2. It is observed that CS students exhibited higher percentages at both extreme ends suggesting a bimodal distribution, while Engineering students' feedback showed higher percentages in the middle of the spectrum. The mean value of CS students was 3.93, while that of Engineering students was 3.65. It shows that CS students have better average learner satisfactions in the learning.

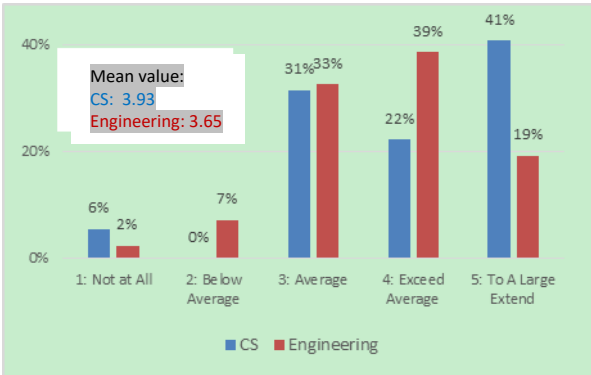

Fig. 2. Comparison on Question 2: *To what extent you like online lecture explanation through worked examples videos?*

3) Q3: Viewing completion of worked example videos

The feedback from CS versus Engineering students for Question 3 are shown in Fig. 3. It is shown that 98% CS students completed watching the videos, while only 78% Engineering students completed watching the videos. It is in line with the academic performance achieved for these two cohorts.

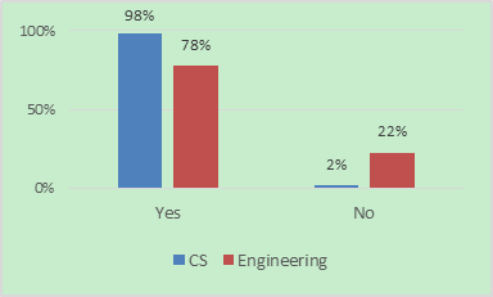

Fig. 3. Comparison on Question 3: *Did you complete entire worked examples videos of Assignment 2?*

4) Q4: Effectiveness of worked example videos

For survey Question 4, the student comparisons are shown in Fig. 4. The mean value of CS students was 4.30, while that of Engineering students was 3.62. It is observed that more CS students appreciated the worked example videos. Engineering students showed less appreciation for the videos. It suggests better learner satisfactions for the CS students, and it could be one of the reasons for better academic results of CS students.

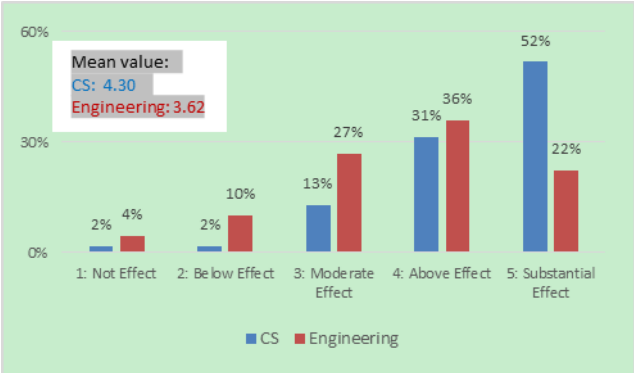

Fig. 4. Comparison on Question 4: *To what extent, worked examples videos help in your understanding assignment requirements and deliverables?*

5) Q5: Inspirational properties of the worked example videos

These worked example videos explain relevant knowledge and working procedure of the project assignments for students. For survey Question 5, shown in Fig. 5, similar opinions among CS and Engineering cohorts were observed for most of the options, except for the second option on the inspirational feature, where more engineering students watched the videos to know more about the course concepts.

6) Q6: Repeated viewing of worked example videos

For Question 6, it is observed in Fig. 6 that 10% Engineering students frequently repeatedly watched the worked example videos. The mean value of CS students was 2.71, while that of Engineering students was 2.95. It shows that Engineering students may spend much time in watching the worked examples videos, when they worked in the project assignments. It shows that the videos provided some useful learning materials that Engineering students may take some time to digest, compared to CS students.

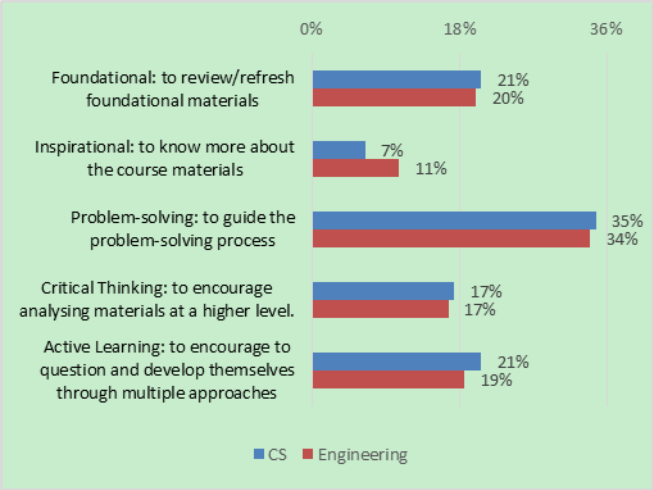

Fig. 5. Comparison on Question 5: *Which approach below best describes the properties of the work examples videos?*

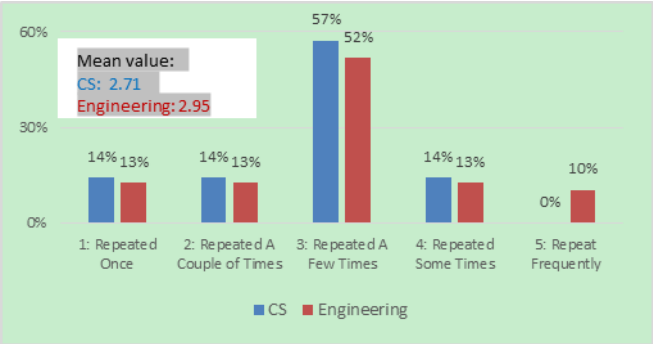

Fig. 6. Comparison on Question 6: *Did you have to repeat the viewing of the videos to understand the assignment better?*

7) Q7: Challenges experienced with blended learner-centric approach

Responses to Question 7 are shown in Fig. 7. It is observed that a slightly lower percentage of CS students reported encountering challenges compared to Engineering students. It is in line with the responses on Question 6 where some students watched the videos repeatedly, when they faced challenges in the learning.

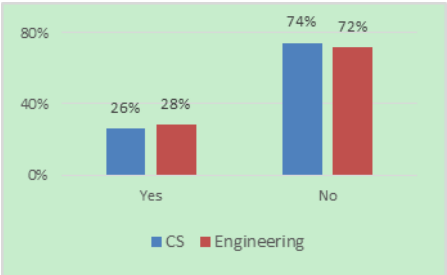

Fig. 7. Comparison on Question 7: *Did you face any challenges of learning in this blended learner-centric approach?*

8) Q8: Future use of the blended learner-centric approach

The feedback comparisons of the survey Question 8 are shown in Fig. 8. It shows that CS students were slightly more positive to have more subjects using the blended learner-centric approach. About 2/3 students from both

cohorts indicate positively with the blended learning-centric approach. It shows the good learner satisfaction achieved, where CS students show slightly higher satisfaction compared to the Engineering students. It is in line with the academic performance achieved for two cohorts.

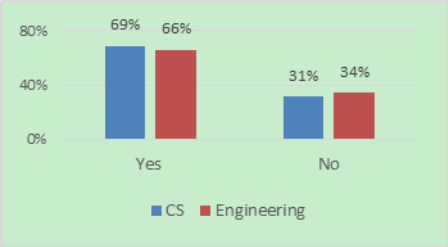

Fig. 8. Comparison on Question 8: *Do you think more courses should be taught this way?*

9) Q9: Optimal proportion of F2F v.s. worked example videos

Question 9 responses are shown in Fig. 9. It is observed that more CS students preferred learning contents consisting of a higher proportion of worked example videos at 60%–80%. It is in line with the observation from the previous survey questions that CS students achieved higher learner satisfactions and faced lower learning challenges in the blended learner-centric approach.

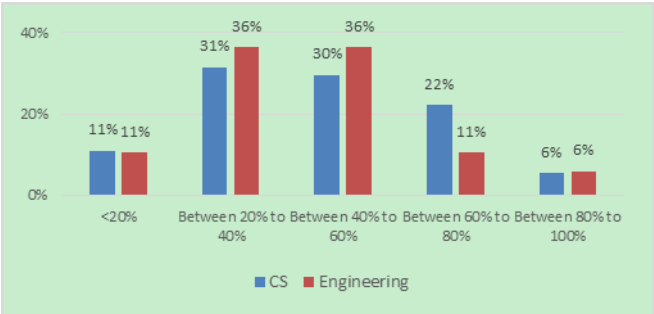

Fig. 9. Comparison on Question 9: *What proportion of work examples videos versus face-to-face lecturing are you comfortable with?*

C. Findings and Discussions

It is observed from the results of academic performance comparisons in Table I–Table IV that both cohorts with varied prior programming knowledge benefited from the blended learner-centric approach, which helped reduce the academic performance gap between CS and Engineering cohorts. The observations are in line with prior works exploring the blended learning for programming courses in the literature [4, 35, 40]. It showed Engineering students benefited more from the blended learner-centric approach comparatively, given their less programming experiences.

The comparison results of survey responses between the two cohorts showed different perceptions of the blended learner-centric approach. Observed from the feedback, CS students reported higher learner satisfactions to learn programming courses with the blended learner-centric approach. The worked example videos for Assignment 2 could help students better understand the profound mathematical concepts and the programming steps to achieve the goal. Observed from the survey results, CS students appreciated the worked example videos more and benefited from watching the videos in the project assignments. It is observed that CS students learned in a steady pace, with almost all students finished watching the videos. Engineering students benefitted more from the worked example videos than CS students in terms of the academic performance obtained. It is through the repeatedly watching the videos to gain better understanding when Engineering students encountered learning challenges. Yet they were less appreciative of them and more likely to have struggled with watching them, as 22% Engineering students did not finish watching these videos. This suggests that some enrolled Engineering students encountered more challenges, similar to those encountered by the study reported in [34]. This may be caused by the lack of prior programming knowledge. As such, Engineering students may need more time to digest the contents of the worked example videos, while they can learn better than the F2F learning. The blended learner-centric approach enabled Engineering students in learning with their own paces, which could be better than the F2F learning. It is observed from the experiment results, more Engineering students realised the benefits of the blended learner-centric approach on programming courses.

The research findings of the comparisons between the enrolled CS and Engineering cohorts are important. However, there are a few limitations in this research as follows.

1) These two group project assignments assumed that every group member contributed equally and proactively. Group assessments were marked with the same score assigned to all members in the group. However, one group complained that one of their members contributed very little in both assignments, known as social loafing in cooperative learning or group work which negatively impacts the group spirit [41, 42]. The team dynamics are an important topic in almost all group projects, that potentially impacts the success of group projects. To address this issue, peer evaluation scores should be incorporated for every student on top of a group assessment score, to motivate all students.
2) The assessment comparisons were only performed by the enrolled students of this subject, not all students, and there could have been a response bias in relation to the survey response rate. The research only involved a single case study in a single institute. The methodological assumptions and findings may not be generalisable to other institutes. However, the transparency in conveying the research design should be helpful with potential transferability to other similar HEIs that teach programming courses.
3) The group sizes were not equal for CS and Engineering cohorts. The reason was the change of the original plan of three members per group, due to the strong requests of Engineering students to increase one more member by assuming themselves less prior programming experience. For a fair comparison in future improvement, the group size of different cohorts should be the same. But it is a minor limitation, due to the analysis by the t-test. The t-test is not necessary to have the assumptions of equal sample size.

IV. Conclusion

The FoP is a compulsory subject to Year 1 students majoring in CS and Engineering in the joint degree programmes of UGS and SIT in Singapore. There is a wide diversity of backgrounds and prior knowledge of students across multi-disciplinary fields. Some students have prior experience in programming, while others may have no programming skills. This research work investigates the difference comparisons on performances and learners' perceptions between CS and Engineering cohorts in learning the programming subject. By adopting a blended learner-centric approach on this FoP subject, the academic performance of both CS and Engineering cohorts were consistently improved. The comparison analyses also show that the performance gap between CS and Engineering cohorts was reduced. Furthermore, the survey of student perceptions reveals that both cohorts are satisfied with the blended learner-centric approach. Although differences in learning procedure and feedback are observed between two cohorts, they are open to the benefits of learning programming courses with such an approach.

V. Future Directions

Moving forward, in the next phase of research, the blended learner-centric approach will be enhanced on both cohorts with different project assignments, thus addressing the limitations presented in Sub-section III-C. The performance results and learners' perceptions will be compared and analysed between two cohorts again.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Q. Cao, C.K. Seow, and L.H.I. Lim conducted research and analyzed the data; Q. Cao, C.K. Seow, L.H.I. Lim, S.L. Keoh, V. Dale, S. Honeychurch, N. Tasler, and D. Bremner wrote and revised the paper. All authors had approved the final version.

Acknowledgment

We would thank our UGS Learning & Teaching Administrative Officers, Ms. Sheila Devi Rajoo and Ms. Sani Sanifah, as well as the programme directors of UGS for their contributions and support in this research.

References

[1] S. Kanbul and H. Uzunboylu, "Importance of coding education and robotic applications for achieving 21st-century skills in North Cyprus," *International Journal of Emerging Technologies in Learning*, vol. 12, 2017, doi: 10.3991/ijet.v12i01.6097.

[2] Q. Cao, L. H. I. Lim, V. Dale, and N. Tasler, "Experiences in python programming laboratory for civil engineering students with online collaborative programming platform," in *Proc. 14th Annual International Conference of Education, Research and Innovation*, 2021, pp. 5784-5791, doi: 10.21125/iceri.2021.1305.

[3] S. Alhazmi, M. Hamilton, and C. Thevathayan, "CS for all: Catering to diversity of master's students through assignment choices," in *Proc. 49th ACM Technical Symposium on Computer Science Education*, 2018, doi: 10.1145/3159450.3159464.

[4] A. Alammary, "Blended learning models for introductory programming courses: A systematic review," *PloS one*, vol. 14, no. 9, 2019, doi: 10.1371/journal.pone.0221765.

[5] C. Alvarado, G. Umbelino, and M. Minnes, "The persistent effect of pre-college computing experience on college CS course grades," *49th ACM Technical Symposium on Computer Science Education*, 2018, doi: 10.1145/3210551.

[6] M. Pedroni and M. Oriol, "A comparison of CS student backgrounds at two universities," *ETH Technical Reports*, 2009, doi: 10.1145/366413.364580.

[7] M. Babes-Vroman, I. Juniewicz, B. Lucarelli *et al*., "Exploring gender diversity in CS at a large public R1 research university," *ACM SIGCSE Technical Symposium on Computer Science Education*, 2017, doi: 10.1145/3017680.3017773.

[8] F. Moller and T. Crick, "A university-based model for supporting computer science curriculum reform," *Journal of Computers in Education*, vol. 5, pp. 415–434, 2018.

[9] C. Chen, S. Jeckel, G. Sonnert, and P. Sadler, "Cowboy and cowgirl programming and success in college computer science," *International Journal of Computer Science Education in Schools*, vol. 2, no. 4, 2019, doi: 10.21585/ijcses.v2i4.34.

[10] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: a review and discussion," *Computer Science Education*, vol. 13, pp. 137–172, 2003, doi: 10.1076/csed.13.2.137.14200.

[11] S. Grover and R. Pea, "Computational thinking in K-12: A review of the state of the field," *Educational Researcher*, vol. 42, 2013, doi: 10.3102/0013189X12463051.

[12] F. Restrepo-Calle, J. J. Ramírez Echeverry, and F. A. González, "Continuous assessment in a computer programming course supported by a software tool," *Computer Applications in Engineering Education*, vol. 27, pp. 80–89, 2019, doi: 10.1002/cae.22058.

[13] H. Özyurt and O. Özyurt, "Analyzing the effects of adapted flipped classroom approach oncomputer programming success, attitude toward programming, and programming self-efficacy," *Computer Applications in Engineering Education*, vol. 26, pp. 2036–2046, 2017, doi: 10.1002/cae.21973.

[14] M. Rahmat, S. Shahrani, R. Latih *et al*., "Major problems in basic programming that influence student performance," *Procedia-Social and Behavioral Sciences*, vol. 59, pp. 287–296, 2012, doi: 10.1016/j.sbspro.2012.09.277.

[15] X. Fang, "Application of the participatory method to the computer fundamentals course," *Affective Computing and Intelligent Interaction*, pp. 185-189, 2012, doi: 10.1007/978-3-642-27866-2_23.

[16] M. Guzdial, "Exploring hypotheses about media computation," *ACM Conference on International Computing Education Research*, 2013, pp. 19–26, doi: 10.1145/2493394.2493397.

[17] S. Bergin and R. Reilly, "The influence of motivation and comfort-level on learning to program," in *Proc. 17th Workshop of the Psychology of Programming Interest Group*, 2005.

[18] S. Dasuki and A. Quaye, "Undergraduate students' failure in programming courses in institutions of higher education in developing countries: A Nigerian perspective," *Electronic Journal of Information Systems in Developing Countries*, vol. 76, pp. 1–18, 2016, doi: 10.1002/j.1681-4835.2016.tb00559.x.

[19] Ö. Korkmaz and H. Altun, "Adapting computer programming self-efficacy scale and engineering students' self-efficacy perceptions," *Participatory Educational Research*, vol. 1, pp. 20–31, 2014, doi: 10.17275/per.14.02.1.1.

[20] L. E. Margulieux, B. B. Morrison, and A. Decker, "Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples," *International Journal of STEM Education*, vol. 7, 2020, doi: 10.1186/s40594-020-00222-7.

[21] L. Porter, M. Guzdial, C. McDowell, and B. Simon, "Success in introductory programming: what works?" *Communications of the ACM*, vol. 56, pp. 34–36, 2016, doi: 10.1145/2492007.2492020.

[22] C. Boudia, A. Bengueddach, and H. Haffaf, "Collaborative strategy for teaching and learning object-oriented programming course: A case study at Mostafa Stambouli Mascara University, Algeria," *Informatica*, vol. 43, 2019, doi: 10.31449/inf.v43i1.2335.

[23] E. Holden and E. Weeden, "The impact of prior experience in an information technology programming course sequence," in *Proc. 4th Conference on Information Technology Curriculum*, 2003, doi: 10.1145/947121.947131.

[24] A. Badawy, K. Schmitt, S. Kramer, et al., "Expectations of computing and other STEM students: a comparison for different class levels, or (CSE $\neq$ STEM - CSE)," *IEEE Frontiers in Education Conference*, 2013, doi: 10.1109/FIE.2013.6685120.

[25] M. Tom, "Five Cs framework: a student-centered approach for teaching programming courses to students with diverse disciplinary background," *Journal of Learning Design*, vol. 8, 2015, doi: 10.5204/jld.v8i1.193.

[26] M. S. Kirkpatrick and C. Mayfield, "Evaluating an alternative CS1 for students with prior programming experience," *ACMSIGCSE Technical Symposium on Computer Science Education*, 2017, pp. 333–338, doi: 10.1145/3017680.3017759.

[27] C. Marling and D. Juedes, "CS0 for computer science majors at Ohio University," in *Proc. 47th ACM Technical Symposium on Computing Science Education*, pp. 138–143, 2016, doi: 10.1145/2839509.2844624.

[28] Z. Dodds, R. Libeskind-Hadas, C. Alvarado, and G. Kuenning, "Evaluating a breadth-first CS1 for scientists," *ACM Technical Symposium on Computer Science Education*, vol. 40, no. 1, 2008, doi: 10.1145/1352322.1352229.

[29] A. Arboleda, L. Mazuera, and G. Montemiranda, "Competences that facilitate the achievement of the objectives of an introductory programming course," *International Conference on Interactive Collaborative Learning*, 2015, doi: 10.1109/ICL.2015.7318166.

[30] W. Lau and A. Yuen, "Exploring the effects of gender and learning styles on computer programming performance: implications for programming pedagogy," *British Journal of Educational Technology*, vol. 40, pp. 696-712, 2009, doi: 10.1111/j.1467-8535.2008.00847.x.

[31] M. J. Kintu, C. Zhu, and E. Kagambe, "Blended learning effectiveness: the relationship between student characteristics, design features and outcomes," *International Journal of Educational Technology in Higher Education*, vol. 14, no. 7, 2017, doi: 10.1186/s41239-017-0043-4.

[32] G. El-Sayad, N. M. Saad, and R. Thurasamy, "How higher education students in Egypt perceived online learning engagement and satisfaction during the COVID-19 pandemic," *Journal of Computers in Education*, vol. 8, pp. 527–550, 2021.

[33] T. B. Bati, H. Gelderblom, and J. Van Biljon, "A blended learning approach for teaching computer programming: design for large classes in Sub-Saharan Africa," *Computer Science Education*, vol. 24, no. 1, pp. 71–99, 2014, doi: 10.1080/08993408.2014.897850.

[34] C. Jacobs, G. Gorman, H. Rees, and L. Craig, "Experiences with efficient methodologies for teaching computer programming to geoscientists," *Journal of Geoscience Education*, vol. 64, no. 3, pp. 183–198, 2016, doi: 10.5408/15-101.1.

[35] K. Tritrakan, P. Kidrakarn, and M. Asanok, "The use of engineering design concept for computer programming course: a model of blended learning environment," *Educational Research and Reviews*, vol. 11, no. 18, pp. 1757–1765, 2016, doi: 10.5897/ERR2016.2948.

[36] *Remote and Blended Teaching*, University of Glasgow, [online]: https://www.gla.ac.uk/myglasgow/anywhere/blendedteaching/.

[37] S. Dart, S. Cunningham-Nelson, and L. Dawes, "Understanding student perceptions of worked example videos through the technology acceptance model," *Computer Applications in Engineering Education*, vol. 28, pp. 1278–1290, 2020, doi: 10.1002/cae.22301.

[38] B. G. Davis, *Tools for Teaching*, Jossey-Bass Inc., San Francisco: California, 1993.

[39] S. A. Burke, "Group work: How to use groups effectively," *The Journal of Effective Teaching*, vol. 11, pp. 87–95, 2011.

[40] I. Cabrera, J. Villalon, and J. Chavez, "Blending communities and team-based learning in a programming course," *IEEE Transactions on Education*, vol. 60, no. 4, 2017, doi: 10.1109/TE.2017.2698467.

[41] P. Aggarwal and C. O'Brien, "Social loafing on group projects: Structural antecedents and effect on student satisfaction," *Journal of Marketing Education*, vol. 30, no. 3, pp. 255–264, 2008, doi: 10.1177/0273475308322283.

[42] Ş. B. Tosuntaş, "Diffusion of responsibility in group work: Social loafing," *Journal of Pedagogical Research*, vol. 4, no. 3, pp. 344–358, 2020, doi: 10.33902/JPR.2020465073.

Copyright © 2023 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).