Challenge-Based and Competency-Based Assessments in an Undergraduate Programming Course

https://doi.org/10.3991/ijet.v16i13.23147

Gaganpreet Sidhu, Seshasai Srinivasan (✉), Nasim Muhammad
W Booth School of Engineering Practice and Technology,
Hamilton, Canada
ssriniv@mcmaster.ca

Abstract—In this work, we investigate an optimal assessment strategy to measure student learning in the first-year undergraduate engineering course at W Booth School of Engineering Practice and Technology at McMaster. Specifically, we evaluate and compare *challenge-based* and *competency-based* assessment strategies. In the challenge-based approach, the students are required to design a C++-based application that meet the required design objectives. The competency-based assessment involves assessing learning by asking a variety of pointed questions pertaining to a single or a small group of concepts. After studying the performance of 207 students, we found that in the challenge-based assessment, due to the complex nature of the questions that assess numerous concepts simultaneously, students who are not very thorough with even one or two concepts fared very poorly since they were unable to finish the challenge and present a functional prototype of the program. On the other hand, the competency-based assessment allowed for a more balanced approach in which the students' learning was reflected more accurately by their performance in the various assessments.

Keywords—Challenge-based learning, competency-based learning, assessment

1 Introduction

Software is integral to numerous applications in this world. In almost every engineering program offered in any university, there is at least one foundational course that focuses on the basic principles of programming. There are numerous programming languages available for this purpose, and the choice is usually based on the utility of the programming language in the other courses that the students have to take as part of their curriculum, towards their degree.

Teaching computer programming has evolved through the decades [1,2]. The challenge in teaching a programming course at an introductory level can be attributed to a variety of reasons, namely, student characteristics and aptitude for programming, the pedagogical techniques applied to teach the course, and the nature of the programming language itself [3-5]. For example, some students prefer to learn in groups whereas other prefer working along and are more comfortable with self-study [6]. A major reason for a poor performance or a high student dropout rate in a programming course is their lack of aptitude and interest towards the subject [3,7]. Also, it is now well known that traditional teaching approaches in which instructors use most of the course time in teaching syntax of the programming is not an optimal pedagogical strategy. Instead, there should be more focus on problem solving skills, employing more student-centric pedagogies that include more collaboration [4,7].

A literature review will show numerous pedagogical techniques are available to teach a course, namely, problem based learning [8-10], small group and co-operative learning [11,12], active learning [13-16], undergraduate research-based learning [17-19], challenge-based learning [20] and inquiry-based learning [21]. For a programming course, a blended-learning approach that includes a combination of lectures, individual and group activities, self-paced activities, and opportunities to discuss and debate ideas, is an effective pedagogical strategy [22,23]. Alammary has presented a detailed review of the blended learning approach that has been used to teach introductory programming courses [3].

In this work, we use a blended learning approach to teach an introductory course in programming in C++. The course is taught to students at the Level 1 of the undergraduate engineering programs at McMaster University's W Booth School of Engineering Practice and Technology (SEPT). Specifically, in our pedagogical approach, we integrate the key benefits of all three influences of learning theory, namely, behavioral, cognitive, and constructivist philosophies. Behaviorism was used to provide an instructor-controlled environment for setting the expectations and ensuring that the deliverables are met in a timely manner. Theories of constructivism and cognition were employed to ensure that the learning environment is enriched with adequate opportunities to engage with peers and instructor to discuss and deliberate upon their ideas and designs. This in turn helps create sound mental construction of the concepts.

Assessing the learning appropriately, that is the focus of this study, is critical to ensure that the students have learnt the material and are well prepared for future courses. The main motivation behind this investigation is the fact that at present there are two assessment approaches being advocated at SEPT. The first is the challenge-based assessment in which students are required to design fairly sophisticated C++-based applications that involve several concepts. The functionality and features of the resulting application will be a direct measure of the learning, and testimony to the employability of future graduates. Advocates of this assessment method point to its advantage of of being able to measure the students' understanding of several concepts, and their ability to integrate and apply several concepts to design an application. The other approach to measure student learning is via competency-based assessments. In this, the student learning is measured via a variety of questions or short programming problems that measure specific competencies with respect to the individual concepts or a collection of 2-3 concepts. Advocates of this method stress that we could measure knowledge, understanding, coding, debugging, and tracing abilities uniquely, giving us a precise measure of the students' learning. In this study, we evaluate both assessment techniques and present our findings for the reader to choose an optimal approach that is suited for their cohort.

The rest of the study is organized as follows: Section 2 outlines the materials and methods used in this study. In this, we describe the course design, procedure for delivery of the course content, and the assessments. The results and discussions are presented in Section 3 were we layout the outcomes of the two assessment techniques and describe the major findings. The conclusions from these findings are drawn in Section 4 that summarizes the outcome of this study.

2 Materials and Methods

SEPT at McMaster offers registration in three undergraduate engineering technology programs, namely, Automotive and Vehicle Engineering Technology, Automation Engineering Technology, and Biotechnology. As part of their curriculum, students enrolled in the 4-year undergraduate engineering programs offered at the SEPT are required to complete programming course in C++. C++ is included in the curriculum because it is an object-oriented programming language that can serve students in the different streams in the upper year courses. Specifically, it is used to program microprocessors and IC’s to carryout signal processing tasks, simulate system processes and faults, and to calculate stress on mechanical parts. Additionally, it can also be used as a high-level programming platform to design and develop software applications. Due to the importance and application of C++ concepts in the curriculum in the upper years, it is offered to all first-year students in the Fall semester. The objective of this course is to teach the basic concepts in programming and develop critical thinking and problem-solving skills by applying scientific and quantitative reasoning to solve science and engineering related problems. Every student registered in Automotive and Vehicle Engineering Technology or Automation Engineering Technology program is required to successfully complete the introductory course in C++ before taking other advanced courses in the areas of object-oriented programming, computational modelling, and numerical linear algebra.

2.1 Course design

This course introduces students to fundamental programming concepts and methodologies for creating C++ programs. It is taught as the first programming course, assuming no prior knowledge of any programming language. Students learn to design, create, debug, and run a variety of C++ programs. Topics covered in this course include: data types, conditional and loop structures, standard input/output, file input/output, functions, arrays, cstring and pointers.

The course was delivered and designed to realize the following learning outcomes:

1. Understand the different data types in C++, select and apply appropriate data types and identifier names for the different applications.
2. Construct and use functions. Write correct function prototypes, definitions, and calls to the functions. Select the appropriate method to pass values or references. Differentiate between void and valued functions. Identify the scope of automatic, static and global variables.
3. Use appropriate input/output methods for different data types and formats.
4. Understand and apply the decision structures, i.e., if, if-else and switch statements.
5. Understand and apply the appropriate looping mechanisms, i.e., for, while, and do-while loops, in applications.
6. Declare, initialize, and manipulate one-dimensional and two-dimensional arrays. Use arrays as function parameters.

2.2 Procedure

The course is delivered for a duration of 13 weeks in a computer lab. Since a computer lab can accommodate a maximum of 40 to 50 students, the course was offered through multiple sections, and students could enroll in any section of the course depending upon their schedule. Thus, each section had a random mix of students from the three programs.

As mentioned in the Introduction section, principles of behaviorism, constructivism, and cognition were used to deliver the concepts to the students. Behaviorism: The instructor outlines the learning objectives to be accomplished, expectations from the students, activities inside as well as outside the classroom and the associated assessments for evaluating students' learning. Cognition: To ensure that student creativity is not lost in an instructor-controlled environment, we ensure that the aspects of behaviorism is only used to initiate the students and maintain the coherence of the group to meet the learning objectives in a timely manner. After setting the stage, the students are involved in a variety of exercises such as live coding in an individual and group setting, group discussions to debug codes, observe coding techniques via live examples from the instructor etc. Thus, with the problem-solving sessions, the students can acquire, infer, develop and evaluate new knowledge by investigating the problem and the applicable solution strategies themselves. By engaging with the peers in doing these activities, the multitude of mental processes will enable cement the concepts in the minds [24,25]. Constructivism: The problem-solving sessions also enable a continuous active learning setup in which the students collaborate with peers and produce meaningful solutions. This exercise involves discussions, sharing problems and solutions, and participation in self-reflection. Collective these acts foster construction of personal interpretations and helps students assign meaning to the knowledge they have gained from this experience [24-26].

More precisely, in this course, the class met twice a week for a duration of two hours per meeting. Students learnt basic programming concepts during the first class of each week. In this, the instructor introduced a new programming concept(s), provided illustrative examples, and engaged in short in-class activities in which the class collectively applied the concepts to develop a program to solve a scientific or engineering related problem. Thus, the live coding exercises was a key pedagogical approach of the course, involving significant amount of interaction between the instructor and the students, creating an interactive active learning environment.

The second class of the week was designed to promote learning by practice. In this class, students solved 2 to 3 science and engineering related programming exercises. Specifically, such problem-solving sessions involved creating a C++ application for some real-world problems using the concepts taught in the previous lectures. For this, the instructor set the stage by explaining these exercises at the beginning of class, laying out the expectations from the students, and the assessment rubric. Subsequently students were allowed to engage in discussions with each other as well as the instructor to develop solutions to these problems. This interactive problem-solving session required the students to engage in three processes [27]: 1. Inferring new information that is not directly provided in the question. 2. Evolving the current knowledge and applying a set of concepts to solve the problem, and 3. Evaluating the solution via test cases. Thus, such sessions helped them acquire, infer, develop, and evaluate new knowledge via explorative and investigative approaches. The multitude of mental processes involved in this activity helps cement the concepts in the minds of the students [24]. Students were given adequate time to submit the solutions, creating an opportunity for them to pursue the discussions and learning outside the class.

2.3 Assessments

Both informal and formal assessments were used to evaluate the performance of the students. Informal assessments were made by engaging the students in the live coding exercises. Students' participation and responses were used to understand the overall learning in the classroom. This allowed for corrective measures and calibrations to the lecture content to ensure a better delivery of the concepts. This may include reviewing the concepts, providing additional examples, as well as providing additional content for review outside the class. In the formal assessments, students were expected to complete 7 lab exercises, and 3 assignments to master the concepts taught throughout the semester. They also appeared for two term tests, and a comprehensive final exam, to measure the student learning in the course.

Labs were based on weekly topics covered in lectures and worth 10% of the total grade. These labs contained 2-3 programming questions that require designing a program with 1-2 concepts that were taught in the previous class. Assignments were given in the weeks 4, 8 and 12, prior to each term test and the final exam. The assignments are collectively worth 10% of the total course grade. These assignments contained 2-3 programming exercises and involved problems that were more challenging than the lab exercises. Each problem requires a set of concepts to be collectively employed to design the solution. The labs and assignments were aimed at preparing the students for the term tests and the final exam.

Term tests and final exams were used to conduct a more elaborate assessment of the learning outcomes. In this course, the term tests were administered in the 5th and

9th week of the course, each for a duration of two hours. Both term tests were closed book exams and were collectively worth 45% of the total grade. The 2 ½ hour final exam scheduled by the registrar's office was also a closed book exam, and was worth 35% of the total grade. The major focus of this work is to determine an effective approach to make these high stakes assessment. Specifically, two approaches were investigated in this study, namely, a challenge-based exams and a competency-based exams. These are described below.

Challenge-based exam: In Cohort-A that appeared for challenge-based term tests and the final exam, students developed applications using Microsoft Visual Studio during the assessment. More precisely, in each term test they developed two application problems which check the understanding of selected topics. In Term test-1 the assessment was over datatypes, standard I/O, conditional structures, and loop structures, whereas in Term test-2 we assessed functions and pointers. Similarly, they developed three applications in the final exam, that collectively assess them on all the topics covered in the course. Thus, the applications developed by the students are to directly demonstrate their understanding of the concepts and generate a functional applications, and indirectly measure their ability to trace and debug the code.

Competency-based exam: The competency-based exams given to Cohort-B covered the same topics as previous challenge-based exams. This was also a closed book closed notes exam and students did not have access to computers. As an important variation from the challenge-based format, the assessment objectives were decoupled. More precisely, the concepts were assessed through a much larger variety of shorter questions. Each question tested either a single concept or a combination of two or three concepts. To do this, the competency-based exam was split into the following four sections:

1. Knowledge and Understanding: This section of the exam checks the students' understanding of concepts in different topics of the course. It contained 8-10 short questions requiring a written response for each question.
2. Trace the Code: This section had about 4-5 questions in which the student determines the output of a given code. Thus, this section checks the students' ability follow and interpret the logic in a code. Note that this skill was not explicitly tested in the challenge-based exams.
3. Debugging: In this section, consisting of about 2-3 questions, students were required to correct an erroneous C++ program which contains syntax errors, logical errors, or both. This section tests the ability to debug a C++ program without using a computer, demonstrating proficiency in the syntax and logical thinking.
4. Coding: This section tests their ability to develop a C++ program. Students were given some instructions and they were asked to write a C++ code. A total of 4-6 short coding problems were given, with moderate levels of difficulty in comparison to the challenge-based exams' application problems that were more comprehensive in nature.

Table 1. Average scores of students in each assessment, and the final course grade. All values are in %. Values in the parentheses indicate the median scores.

Assessment Mode	Test 1	Test 2	Final Exam	Final Grade
Challenge-based	80 (87)	65 (68)	58 (54)	68 (70)
Competency-based	68 (68)	62 (64)	73 (75)	71 (71)

3 Results and Discussion

The study was conducted with a total of 207 students. 80 students from this group appeared for the challenge-based assessments and the rest appeared for the competency-based assessments. The content taught to all the students were identical, and the students performed the same labs and assignments in preparing for these exams.

The performance of the students in the two cohorts are summarized in Table 1. As seen in this table, in the cohort that took the challenge-based assessments, the performance of the students continuously deteriorated from Term test 1 to the final exam. On the other hand, the performance of the students appearing for the competency-based assessments had reasonably moderate fluctuations through the course.

This trend in the results can be explained as follows: In the challenge-based testing, the problems get increasingly complex as the term progresses. Hence, being the easiest, we see a good performance of the students in the first test. As we move to Term test 2 and the final exam, the performance of the students drops since the level of difficulty increases due to the complexity of the topics. It must be noted that the higher level of difficulty is largely due to the fact that in the challenge-based exams, when students are required to solve just 2-3 application problems, the questions are designed to assess them on an array of topics in a given question.

On the other hand, in the competency-based testing, the student performance remains fairly consistent across the different assessments. The first term test focuses on the preliminary concepts that are relatively easy whereas the second term test focuses on more advanced concepts. Hence, it is natural for the average scores to decrease. However, we notice that unlike the challenge-based testing, in the competency-based format, the student performance is down only by about 6% and one could argue that this not statistically very significant. The final exam is comprehensive and focuses on all topics and the students are largely comfortable with the contents of the material. Hence, they perform very well in the final exam, rebounding to an average score of 73%.

As seen in Table 1, the overall course grade is not very different between the two formats. However, it must be noted that the 68% average in the challenge-based testing is largely due to the very high score of 80% in the first Term test. To obtain more clarity on this seemingly insignificant difference of 2% in the final course grade across the two testing formats, we investigated at the distribution of the grades in the two cohorts (Figure 1). As seen in Figure 1, in the cohort appearing for the challenge-based assessments, 59% of the students get an A grade in Term test 1 and this percentage shrinks to 26% by the final assessment, i.e., the final exam. Further, with the complexity of the course increasing through the term, the number of students failing the assessment increases from 9% in Term test 1 to an alarming 42% in the final exam. The overall failure rate in the course is large at 15%. Also, about 26% of the students pass the course with a grade of D, indicating that they are not appear to be very well prepared for the future courses that require strong programming skills.

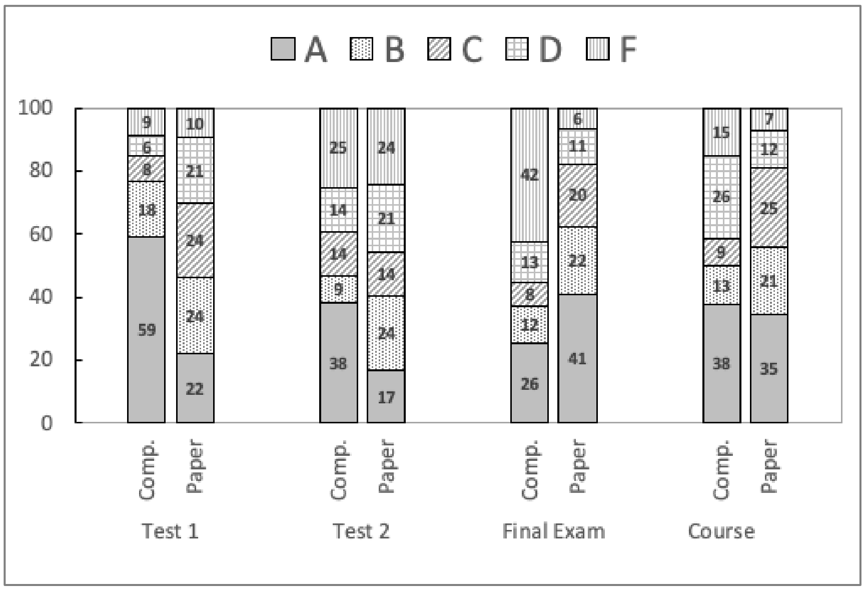

Fig. 1. Distribution of the grades in the three assessments administered to the two cohorts. Cohort-A appeared for the challenge-based assessments, and Cohort-B appeared for the competency-based assessments. All the numbers are in %

On the other hand, in the competency-based exams, the number of students getting an A grade increases from 22% in the Term test 1 to about 41% in the final exam. The percentage of students failing the second Term test is very high at approximately 25% in both formats. However, the positive effects of the constructivist theory employed through the course is better reflected in the competency-based testing where we observe a sharp decline in the failure rate of the students (6%). As mentioned before, in the challenge-based tests, students are required to design 2-3 applications and in each applications the students are assessed on numerous concepts. As a result, when the students are unable to demonstrate the concept, it reflects on the entire application, and the resulting design of the application is very poor, resulting in a low score. Much worse, the stress induced due to a non-functional application creeps into the design of the other questions as well, and the student is unable to adequately demonstrate all the learning. On the other hand, in the competency-based format, we are able to distinctly examine the student for various competencies through a large variety of questions. As a result of this decoupling and assessment of various competencies through several distinct and focused questions, if the students are not good at one or two concepts, it can be uniquely identified, and it does not reflect on the other questions. With this, the outcome is that the number of students failing the course in the competency-based format is nearly half at 7%. In fact, unlike the challenge-based assessments where nearly 41% of the class was not doing well (the students getting F and D grades are 15% and 26%, respectively), this percentage is just 19% (the students getting F and D grades are 7% and 12%, respectively) in the competency-based assessments.

In summary, we observed that challenge-based exams incorporate the assessments of various skill sets and course learning outcomes, but they do not fully capture the learning of an individual student, especially in the final exam. A major issue is that students who are struggling with even just one or two concepts are unable to design a functional applications. As a result, the students often feel defeated and submit incomplete solutions for evaluation. The competency-based format overcomes this challenge by spreading the assessment of various competencies across a large variety of questions. The students who are missing just a few concepts are still able to demonstrate their learning of other concepts that are independently tested through other questions.

Two other issues that is prevalent in challenge-based assessment are: (1) The scheduling of exams is dependent on the availability of the computer labs and poses a major constraint on design of exam timetable. Often, this means that the students appear for this exam at the end of the examination period when they are mentally exhausted after appearing for exams of other courses. While we do not have a direct evidence, we believe this is also an important contributor to the poor performance of the students in the final exam. On the other hand, with the competency-based format, exams are easily schedule since there are no constraints on examination environment. (2) Malfunctioning of a computer during an exam in the challenge-based assessment poses a major hurdle. While this is more of an exception than a norm, this results in an enormous stress on the student who often has to retake the exam, significantly impacting the performance of the student in that assessment. Further, in such circumstances, to account for the disruption, the student needs to be accommodated with extra time to complete the assessment. This stretches the examination duration for the student that is not ideal. On the other hand, the competency-based exams are devoid of such technical issues.

4 Summary and Conclusion

In this work, we examine two assessment formats in a Level-1 undergraduate engineering course on C++ programming. The first format is a challenge-based assessment in which over three exams, the students are required to design a set of C++-based applications. In this, each application is intended to evaluate their learning of several concepts simultaneously. The second format is competency-based exams in which over the three exams, the competencies are tested through a much larger variety of shorter questions. Each question tests either a single concept or a combination of 2-3 concepts. The study was conducted with a total of 207 students, 80 of whom appeared for the challenge-based assessments and the rest appeared for the competency-based assessments.

The performance of the students, as reflected in the two Term tests and a comprehensive final exam, deteriorated with the challenge-based exams. Approximately 40%

of the students failed in the final exam in this format (c.f. Figure 1). In the competency-based exams, the performance of the students remained fairly consistent with just 6% of the students failing the final exam. While the overall course average was nearly the same in the two formats (c.f. Table 1), we found that the performance of students appearing for the challenge-based exams very significantly poor compared to the competency-based exams. Nearly 14% of the students failed the course and an additional 26% received a grade of D. indicating that they are probably not well prepared for the upper-level courses. On the other hand, in the competency-based exams, only 7% of the students failed the course and about 12% students passed the course with a grade of D.

A major disadvantage of challenge-based testing is that due to the complex nature of the questions, deliberately designed to assess several concepts in a single application, a student who is not well prepared with even one or two concepts will find the exam very challenging. In such cases, students often submit incomplete and dysfunctional applications that adversely impact their grade. Other disadvantages with the challenge-based assessments include scheduling hurdles due to the constraint of requiring a computer lab environment, and potential technical problems during the assessment period in the form of abrupt shut down of a computer, a non-responsive computer etc. that will negatively impact the students' performance. On the other hand, in the competency-based format, the student learning is evaluated by decoupling and assessing a concept of two almost distinctly, covering all the concepts via an variety of questions. This format also has short programming questions that can test the students' ability to combine 2-3 concepts to design a small program. As a result, if a student lacks a sound understanding of one or two concepts, they are uniquely identified, without impacting the evaluation of the other concepts that happen via other questions. Put differently, we can assess the student learning in a more comprehensive and fair manner. Also, this format is devoid of challenges such as scheduling constraints or technical disruptions that we face in challenge-based testing.

5 Acknowledgement

The authors are grateful to the reviewers for their time in evaluating the manuscript and providing valuable suggestions to improve the quality of this work.

6 References

[1] Hendrik H., and Hamzah A. (2021) Flipped Classroom in Programming Course: A Systematic Literature Review. Int. J. Emerging Technologies in Learning, 16(2):220-236. https://doi.org/10.3991/ijet.v16i02.15229

[2] Ivanovic M., Xinogalos S, and Komlenov Z. (2011) Usage of Technology Enhanced Education Tools for Delivering Programming Courses. Int. J. Emerging Technologies in Learning, 6(4): 23-30. https://doi.org/10.3991/ijet.v6i4.1796

[3] Alammary A (2019) Blended learning models for introductory programming courses: A systematic review. PLOS ONE 14(9): e0221765. https://doi.org/10.1371/journal.pone.0221765

[4] Alturki, R. A. (2016). Measuring and Improving Student Performance in an Introductory Programming Course. Informatics in Education.15(2):183–204. https://doi.org/10.15388/infedu.2016.10

[5] Gomes, A, and Mendes AJ, editors. (2007). Learning to program-difficulties and solutions. International Conference on Engineering Education–ICEE; 2007.

[6] Çakıroğlu Ü. (2014). Analyzing the effect of learning styles and study habits of distance learners on learning performances: A case of an introductory programming course. The International Review of Research in Open and Distributed Learning. 15(4). https://doi.org/10.19173/irrodl.v15i4.1840

[7] Gomes, A, and Mendes, A, editors. (2014). A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. Frontiers in Education Conference (FIE), 2014 IEEE; 2014: IEEE. https://doi.org/10.1109/fie.2014.7044086

[8] Capon, N. and Kuhn, D. (2004). What's So Good About Problem-Based Learning? Cogn. Instr., 22 (1), pp. 61–79. https://doi.org/10.1207/s1532690xci2201_3

[9] Dochy, F., Segers, M., den Bossche, P. V. and Gijbels, D. (2003). Effects of problem-based learning: A meta-analysis, Learn. Instr., 13 (5), pp. 533–568. https://doi.org/10.1016/s0959-4752(02)00025-7

[10] Centea, D. and Srinivasan, S. (2016). A Comprehensive Assessment Strategy for a PBL Environment, Int. J. Innov. Res. Educ. Sci., 3 (6), pp. 364–372.

[11] Wage, K. E., Buck, J. R., Wright, C. H. G. and Welch, T. B. (2005).The signals and systems concept inventory, IEEE Trans. Educ., 48 (3), pp. 448–461. https://doi.org/10.1109/te.2005.849746

[12] Buck, J. R. and Wage, K. E. (2005). Active and cooperative learning in signal processing courses, IEEE Signal Process. Mag., 22 (2), pp. 76–81. https://doi.org/10.1109/msp.2005.1406489

[13] Freeman, S. , O'Connor, E. , Parks, J. W., Cunningham, M., Hurley, D., Haak, D., Dirks, C. and Wenderoth, M. P. (2007). Prescribed active learning increases performance in introductory biology, CBE Life Sci. Educ., 6 (2), pp. 132–139. https://doi.org/10.1187/cbe.06-09-0194

[14] Hoellwarth, C., Moelter, M. J. and Knight, R. D. (2005). A direct comparison of conceptual learning and problem solving ability in traditional and studio style classrooms, Am. J. Phys., 73 (5), pp. 459–462. https://doi.org/10.1119/1.1862633

[15] Sidhu, G. and Srinivasan, S. (2018). An Intervention-Based Active-Learning Strategy To Enhance Student Performance in Mathematics. Int. J. Pedagog. Teach. Educ., 2 (6), pp. 277–288.

[16] Srinivasan, S. and Centea, D. (2018). An Active Learning Strategy for Programming Courses, in Proceedings of 2018 International Conference on Interactive Mobile Communication, Tech- nologies and Learning (IMCL2018), pp. 21–30.

[17] Hunter, A.-B., Laursen, S. L. and Seymour, E.(2007). Becoming a scientist: The role of undergraduate research in students' cognitive, personal, and professional development, Sci. Educ., 91 (1), pp. 36–74. https://doi.org/10.1002/sce.20173

[18] Srinivasan S, Rajabzadeh AR and Centea D. (2020). A Project-Centric Learning Strategy in Biotechnology: in "The Impact of the 4th Industrial Revolution on Engineering Education. ICL 2019. Advances in Intelligent Systems and Computing.": pp 830-838. Springer Nature, Switzerland. https://doi.org/10.1007/978-3-030-40274-7_80

[19] Bogoslowski, S., Geng, F., Gao, Z., Rajabzadeh, A. and Srinivasan, S. (2021). Integrated Thinking - A Cross-disciplinary Project-Based Engineering Education: in "Visions and Concepts for Education 4.0", eds M. E. Auer and D. Centea, ICBL 2020, AISC 1314, pp 1-7, Published by Springer Nature, Switzerland. https://doi.org/10.1007/978-3-030-67209-6_28

[20] Roselli, R. J. and Brophy, S. P. (2006). Effectiveness of Challenge-Based Instruction in Biomechanics, J. Eng. Educ., 95 (4), pp. 311–324. https://doi.org/10.1002/j.2168-9830.2006.tb00906.x
[21] Lewis, S. E. and Lewis, J. E. (2005). Departing from Lectures: An Evaluation of a Peer-Led Guided Inquiry Alternative, J. Chem. Educ., 82 (1), pp. 135. https://doi.org/10.1021/ed082p135
[22] Alammary, A., Carbone, A., and Sheard, J., editors. (2016). Blended Learning in Higher Education: Delivery Methods Selection. ECIS; 2016.
[23] Dziuban, C., Graham, C.R., Moskal, P.D., Norberg, A., Sicilia, N. (2018). Blended learning: the new normal and emerging technologies. International Journal of Educational Technology in Higher Education. 15(1), pp.3. https://doi.org/10.1186/s41239-017-0087-5
[24] Merriam, S. and Caffarella, R. (1999). Learning in adulthood: A comprehensive guide. San Francisco: Jossey-Bass.
[25] Srinivasan, S. and Muhammad, N. (2020). Implementation of a Course in Computational Modeling of Biological Systems in an Undergraduate Engineering Program. International Journal of Engineering Education Vol. 36, No. 3, pp. 857–864.
[26] Nan I., Kau B., and Rugelj J (2008) Pair Programming as a Moderin Method of Teaching Computer Science. Int. J. Emerging Technologies in Learning, 3: 1-5.
[27] Knowles, M. (1984). The adult learner: A neglected species, 3rd ed. Huston: Gulf.

7 Authors

Gaganpreet Sidhu has a PhD in Materials Science from Ryerson University, Canada. She is currently a Postdoctoral Fellow and a part-time instructor at McMaster University’s Faculty of Engineering. Her pedagogical research interests include learning pedagogies, technology in education, and curriculum development.

Seshasai Srinivasan has a PhD in Computational Science and Engineering from Michigan Technological University, USA. He is currently the chair of Software Engineering Technology program at McMaster University’s Faculty of Engineering. Prior to this, he has held a Research Scientist and a part-time instructor position at the Department of Mechanical and Industrial Engineering at Ryerson University, a postdoctoral position at the Laboratory of Food Process Engineering at Swiss Federal Institute of Technology (ETH-Zurich) in Switzerland and a Research Associate position at the Engine Research Center at the University of Wisconsin-Madison. His pedagogical research interests include cognitive psychology in teaching and learning, active learning, problem-based learning, enhanced learning tools and methodologies, microinstructors in social media and curriculum development.

Nasim Muhammad has a PhD in Applied Mathematics from University of Guelph, Canada. He has more than 32 years of experience teaching various courses in the field of Mathematics and Computer Science in a variety of classroom settings. Nasim is currently teaching math and programming courses in the School of Engineering Practice and Technology at McMaster University’s Faculty of Engineering. His pedagogical research is focused on: developing teaching and learning techniques to enhance students’ learning, classroom dynamics, cognitive psychology, impact of leading-edge technologies in mathematics, and curriculum development.

Article submitted 2021-04-06. Resubmitted 2021-05-05. Final acceptance 2021-05-06. Final version published as submitted by the authors.