Using Flipped Classroom and Team-Based Learning in a First-Semester Programming Course: An Experience Report

Hrafn Loftsson
Department of Computer Science
Reykjavik University
Reykjavik, Iceland
hrafn@ru.is

Ásrún Matthíasdóttir
Department of Sport Science
Reykjavik University
Reykjavik, Iceland
asrun@ru.is

Abstract— **The use of the flipped classroom (FC) approach and team-based learning (TBL) has gained popularity in recent years by instructors in introductory programming courses (CS1), due to increased emphasis on student success and active learning. In this paper, we present an experience report about using FC and TBL in a CS1 course. We present the motivation for restructuring the course, the specific implementation, the results of two student surveys, and the outcome of several exams. We discuss the results, present what actions were taken during the course period, and what changes will be carried out in the future. The results from the surveys show that 47% of the students were pleased with the organization of the course, whereas up to 33% of the students were displeased (in particular the female students). About 60% of the students liked the TBL in class, but about half of the students felt that the course lacked traditional lecturing. Finally, it was surprising that 44% of the students never or seldom read the textbook before class, while 74% watched the videos.**

Keywords—flipped classroom, team-based learning, CS1, experience report

I. Introduction

Learning programming has been considered difficult for many students, which often leads to high failure and drop-out rates from programming courses [10]. A large study involving 161 introductory programming courses in 15 countries shows that the mean worldwide pass rate is 67.7%, and that it has not significantly improved over time [13]. Moreover, the study shows that there is no significant difference in the pass rates based on the programming language taught. Kunkle and Allen [5] state that theories have been put forth about the difficulties that novices experience in learning to program, but that several decades of research have not yielded definitive answers. Luxton-Reilly [7] indeed challenges the view that learning programming is hard and states that our expectations of what students should be able to do at the end of a first programming course are unrealistic.

Due to increased emphasis on student success, active learning, and effective communication between instructor and student, several CS departments have experimented with using the *flipped classroom* (FC) approach and *team-based learning* (TBL) in programming courses [1, 4, 6, 11, 12]. In FC, in-class and out-class activities are flipped: students are expected to study a specific course material outside the class and then be able to apply the knowledge to complete various activities related to the material during class under the guidance of an instructor.

One way of putting structure to FC is using TBL [3, 6]. In TBL, students work in teams to apply their knowledge as opposed to working individually on activities. One of the key components of TBL is that students must have immediate feedback. Moreover, in TBL it is common to assess the student's level of understanding on the assigned readings, both with an individual test and a group test [6].

Thus, the FC approach and TBL offers an opportunity to incorporate active learning methods in the classroom, while still covering necessary learning material. The main objective during class time is to test the understanding of concepts and apply them on increasingly complex problems. Some research has shown that these approaches can improve student's learning and performance in an introductory programming course [1, 6, 12].

Finelli et al. [2] state that previous research has shown that there are several barriers to adoption of active learning methods, e.g. concerns about student resistance, efficacy of the methods, teacher preparation time, and the ability of the teacher to cover the course syllabus. In particular, the authors propose stategies to reduce student resistance, which they define as "negative behavioral responses to active learning".

In this paper, we present an experience report about using FC and TBL in a first-semester introductory programming course (CS1) at Reykjavik University. We present the motivation for restructuring the course in Section II and the specific implementation in Section III. The results and discussion about two student surveys and exams are presented in Section IV and instructors' reactions in Section V. Finally, a conclusion is presented in Section VI.

II. Motivation

The BSc program in the Department of Computer Science (DCS) at Reykjavik University is a three year program. The vast majority of the students who enroll into the CS program are novice programmers.

Before the work reported in this paper, the first-semester, 12-week, programming course in the DCS had been taught using a traditional lecture and laboratory-based format. The instructor introduced the text material in lectures (3-4 lecture hours per week) and demonstrated programming concept while the students listened and sometimes participated in the programming part. In the labs (2 lecture hours per week), students were given programming exercises to practice the concepts given in the lectures. Additionally, several homework projects were given during the term, on which the students could work in a group of two. The C++ programming language had been used in the course for several years.

In fall semester 2018, the DCS decided to use the FC approach and TBL in the CS1 course, in order to encourage the students to take more active part in their learning. By freeing class time for in-class activities, students would get more practice in solving programming problems and students interaction with the instructor would increase. At the same time, the programming language used in the course was changed from C++ to Python.

III. Implementation

At the start of the CS1 course, 325 students were registered. In order to facilitate the FC approach and TBL, the students were divided into seven sections. Five of the sections had about 55 students each, whereas the remaining 50 students were divided between a special evening class section and an off-campus section. Inside each section, students were randomly divided into a group of 5–6 students. Each section met two days a week in class, for four lecture hours each day.

One faculty member, the main instructor, was responsible for the overall organization of the course (syllabus, assessment, quizzes, projects, exams, etc.). Six instructors and seven teaching assistants had the role of tutoring in the sections (one instructor had two sections).

Our implementation of FC and TBL is based on implementations described in e.g. [1, 3, 4], and is as follows:

- In advance of most of the classes, students were expected to read a given chapter of the textbook [9] and watch 1–2 short YouTube videos related to the specific concepts. The videos were provided as supplementary material to the (more important) textbook chapters.
- At the beginning of each class, the students were given the chance to ask questions about the material and in some cases the instructor spent 5–10 minutes giving an overview of the main topics in the underlying chapter.
- After the question and overview session, in most of the classes students were given a short quiz, containing ten multiple choice questions which were directly linked to the given textbook material. After this individual quiz, students discussed the same quiz in their group and the team turned in a single copy as their collective answers.
- For the remainder of the class, students were given several short programming assignments for practicing the specific concepts. Students worked on these assignments in their groups, but each student needed to submit his/her solution at the end of the class. If a student was not able to finish all the assignments in class, he/she had the opportunity to submit a solution four hours (at the latest) after the class finished.
- In addition to the short programming assignments given in class, the students were given larger programming projects each week to be worked on at home, optionally in a group of two students.
- Each week a special open session was available in which students could receive additional help (from 2–3 teaching assistants) on any issue regarding the course.

Table I: The general organization of each class

Activity	**Description**
Discussion	Students ask questions and the instructor gives an overview of the material.
Individual quiz	Students answer 10 multiple-choice questions.
Group quiz	Students discuss the same quiz in groups and hand in a single copy as answers.
Programming assignments	Students work on the assignments in groups but each student submits his/her solution.

Table I shows the breakdown of activities performed in most of the classes during the course period.

The course assessment was the following:

- Quizzes (individual and group) in class: 10%.
- Short programming assignments in class: 15%.
- Homework programming projects: 15%.
- Two midterm exams (the higher one counted): 20%
- Final exam: 40%

Solutions to individual quizzes were automatically graded by Mimir Classroom. [1] For the group quizzes, IF-AT (Immediate Feedback Assessment Technique) scratch cards, which reveal the results to the student as they answer the questions (see [6]), were used. Solutions to the programming assignments in class were automatically graded by Mimir using predefined test cases. Thus, an effort was made to make sure that students received immediate feedback in class on their level of understanding.

Solutions to the homework programming projects were also run against test cases in Mimir, but the final grade was decided by a group of 6–7 teaching assistants, who based the grade on the code quality as well as the functionality. Solutions to exams (midterms, final and retake) were submitted by students in Mimir, but graded by the instructors in a similar manner as the homework assignments.

It should be evident from the description above that considerable effort and manpower was needed to administer and run the FC/TBL version of the CS1 course. Overall, about 20 individuals (instructors and teaching assistants) contributed to the course in some way or another. This is about double the number needed to administer and run the course when it had been taught in the traditional manner.

IV. Results and Discussion

In this section, we present and discuss the results of two online surveys carried out among students registered in the course. Furthermore, we discuss results from the various exams given during the course period.

[1] https://www.mimirhq.com/

Table II: Answers to the question "Are you generally pleased or displeased with the course"

Rating	Answer	Count	Ratio
5	Very pleased	39	20%
4	Rather pleased	52	27%
3	Moderate	40	20%
2	Rather displeased	36	18%
1	Very displeased	29	15%

Table III: Answers to the question "Are you generally well or poorly prepared for classes"

Rating	Answer	Count	Ratio
5	Very well	52	26%
4	Rather well	84	43%
3	Moderate	49	25%
2	Rather poorly	10	5%
1	Very poorly	1	1%

A. First survey

The first survey was administered by the Office of Teaching Affairs and was given to students in the 5th week of the course. Of the 325 registered students, 196 (60%) submitted answers. The survey consisted of 8 questions of which the most important two (for the purpose of this paper) are shown in Tables II and III (both questions are rated on a five level Likert scale).

Table II shows that 47% of the students were pleased with the course (gave the rating 5 or 4) whereas 33% of the students were displeased (gave the rating 2 or 1). The average rating was 3.17. This result came as a surprise to the instructors, because in previous runs of the CS1 course the rating distribution has not been as bi-modal as here is shown. Usually, the ratio of students being displeased with the course has been in the range 10–15% and a higher ratio of students has been pleased. What was even more surprising was the fact that the average rating given by the female students was only 2.72, compared to the average rating of 3.45 given by the male students. Later in this section, we argue for a plausible reason for this gender difference.

Table III shows that only 6% of the students felt that they were poorly prepared for classes. In Section IV.B, we will compare this to the result of a question regarding textbook reading before class.

In this first survey, students were able to comment on what were the advantages of the course and what could be improved. The following items were mentioned most often as advantages:

- The organization of the course / the flipped classroom
- The amount of useful and challenging programming problems
- Python
- Mimir Classroom
- The need to come prepared for class
- Quiz at the start of each class

The following items were mentioned most often that could be improved or mentioned as disadvantages:

- More teaching (as in traditional lecturing) needed
- Allow students to submit programming assignments later than four hours after class finishes
- Projects too hard / too much work
- Grades for homework projects should be returned sooner

When further analyzing the text responses it became clear that several of the female students were intimidated by some male students in the TBL work in class. Presumably, the reason is that in most cases the female students are novice programmers, whereas some of the male students enter the course with previous programming background (as shown in Section IV.B). This may be partly the reason why the average rating given by the female students is much lower than the average rating given by the male students.

Many students pointed out that more traditional teaching was needed, either in class or with video lectures. The enhancement made to the course (after this first survey) was that the instructor in each section started every class by discussing the material as well as solving the first programming assignment jointly with the students.

Many students felt that there was too much pressure and too little time for submitting solutions to the programming assignments given in class. Some of the students wanted to be able to return these assignments the day after. The instructors, however, were not willing to change this part, because they wanted the students to work on the assignments mainly in groups inside the class.

B. Second survey

The second survey, designed by the authors of this paper, was much more comprehensive than the first one. It consisted of 23 questions and given to students in the 10th week of the course. 178 (55%) of the registered students answered this survey. The first four questions were background questions identifying the participants‘ gender, age, semester, and programming skills:

- 114 (64%) of the participants were male students and 64 (36%) female students.
- The participants‘ average age was 24.4 years, ranging between 18 and 46 years.
- Most students, or 148 (83%), were first semester students.
- Most of the participants, or 119 (67%), rated their programming skills very little or little before they entered the course and only 24 (14%) students rated it as great or very great. The average rate (on a five level Likert scale) was 2.25 for male students and 1.52 for female students.

The remaining nineteen questions were geared towards the course organization, the teaching, study material, midterms, group work and use of systems. The results for selected

Table IV: Course organization and the teaching

Question	Totally agree and agree	Totally disagree and disagree
The organization of the course is good	47%	27%
The classes each week are useful to me	46%	29%
The course lacks traditional lectures	49%	30%
Communication with instructors in class help me to study	61%	16%
I like the organization of the quizzes	45%	28%

Table V: Study material and group work

Question	Totally agree and agree	Totally disagree and disagree
The textbook helped me in my studies	25%	45%
I usually read the textbook before class	35%	44%
The videos helped me in my studies	58%	20%
I usually watch the videos before class	74%	13%
The discussion with fellow students in class helped me in my studies	59%	23%
I like to work in a group with fellow students	62%	17%

questions are presented in Tables IV and V. In what follows, we interpret and discuss the results.[2]

The first question shown in Table IV and the question in Table II are similar in nature. Answers to these two questions show an almost similar bi-modal distribution and that a substantial part of the students (27–33%) were not pleased with the course.

Table IV shows that half of the students (49%) felt that the course lacked traditional lectures and this result is consistent with the most common criticism in the first survey. This table also shows that the majority of the students felt that communications with the instructor in class was beneficial, which is indeed one of the advantages of using the FC approach.

The most striking result for the instructors in this survey can be seen in the answers to the first two questions (regarding the textbook) in Table V. Only 25% of the students felt that the textbook was of help in their studies and only 35% usually read the textbook before class. On the other hand, Table V shows that most students prepared for classes by watching the videos, which were considered by the instructors to be a supplementary material for the textbook. Recall that a small ratio (6%) of students felt that they were poorly prepared for classes, according to the result presented in Table III. It is thus clear that many students do not see reading the textbook as part of the preparation for class.

Finally, Table V shows that the majority of the students liked to communicate with fellow students and the group work in class, which, in our mind, supports the continuing use of TBL in the course.

Table VI: Exam results

Exam	Students	Average grade	Failure rate
Midterm 1	281 (86.5%)	7.1	19.9%
Midterm 2	227 (69.8%)	6.3	36.1%
Final	279 (85.8%)	4.4	55.6%
Retake	133 (40.9%)	5.4	41.3%

C. *Exam results*

In this course, students were able to take four exams, i.e. two midterms, a final exam, and a retake exam. All the exams were "open book", i.e. students were allowed to use the textbook, slides, notes, and solutions to assignments in the exam. Grades are given on a 0–10 scale, and a grade below 5 is a failing grade.

The first midterm was given in the 4th week of the course. The material for the exam were basic programming concepts like variables, types, operators, assignment statements, expressions, if-statements, and loops. The second midterm was given in the 8th week of the course. In addition to the material covered in the first exam, the second one included the following concepts: functions and top-down refinement, scope, file I/O, exception handling, lists and tuples. At the time of final exam, the following concepts had been added: dictionaries, sets, (large) program development, and classes.

Results of the four exams are shown in Table VI. Of the 325 students registered at the start, 281 (86.5%) showed up for the first midterm. We can thus assume that about 14% students had dropped out of the course by week 4. The number of students participating in the second midterm dropped down to 227. The drop is due to the fact that only the higher grade of the two midterm counted toward the final grade and many students that had received a high grade on the first midterm decided to skip the second one.

The failure rate of about 56% in the final exam was the highest the DCS had seen in many years. This was disappointing to the instructors – the failure rate in the final exam in previous running of the course had been in the range 33–50%.[3] For further comparison, the failure rates in the other first-semester 12-week courses (taught using a traditional lecture and laboratory-based format) pursued by the students, Discrete Mathematics, Software Analysis and Design, and Computer Architecture, was 23.3%, 20.7%, and 17.8%, respectively.

When grading the final exam, it became evident that the most common problem the failing students had was the inability to apply functional decomposition, i.e. break a problem description into individual tasks and implement functions for those tasks. Many of the homework programming projects indeed practiced this skill as well as several of the programming assignments in class. In our opinion, greater emphasis needs to be put on this skill in future running of the course. Furthermore, we believe that the lack

[2] In another paper [8], we apply linear regression, on the data obtained in the second survey, in order to find out what best explains the students' positive learning experience.

[3] The final (Python) exam in this course was comparable to the final (C++) exams given in the previous years. The grading schemes used in the Python exam and the C++ exams were also similar.

of textbook reading (see Section IV.B) plays a role in the high failure rate. Note, however, that we are not able to confirm this belief, due to the anonymity of our surveys.

On the other hand, it should be mentioned that 49 students (17.6%) did very well on the final exam, i.e. obtained a grade higher or equal to 9.0. 85% of these students were not novice programmers, according to the results of a special survey given to these students.

The number of students that showed up in the retake exam[4] was 133, of which 128 had taken the final exam. The number of students that had failed the final exam was 155, of which 27 did thus not show up in the retake exam.

Overall, 199 students, or 61.2% out of the 325 students registered at the start, passed the course. This is a bit lower pass rate than the average worldwide pass rate of 67.7% in CS1 courses (see Section I).

V. Instructor's Reactions

As presented in Sections IV.A and IV.B, a significant part of the students, or 27–33%, were displeased with the organization of the course. This ratio is higher than the DCS has experienced in previous (traditional) running of the course. Furthermore, as presented in Section IV.C, the failure rate in the final exam was the highest the DCS had seen in many years. Both of these issues came as a surprise to the instructors, because they had anticipated that, by switching to FC and TBL, the students would be at least as satisfied with the course as in earlier years, and would be able to do at least as well on the final exam.

The six instructors of the course were generally satisfied with the course organization and all of them want to continue with using FC and TBL in CS1. In particular, they liked the effective communication with students in class and the tutoring related to the various programming assignments. There are, however, several issues the instructors would like to improve when running the course again using FC and TBL (these improvement are related to part of the criticism that surfaced in the surveys discussed in Sections IV.A and IV.B):

- **Lack of lecturing.** The most common criticism by students was the lack of traditional lecturing. The instructors had not anticipated this criticism, but it is understandable given the fact that 44% of the students either never or seldom read the textbook before class (see Table V). When teaching this course next time, the instructors will make it even clearer that reading the textbook before class is of prime importance, but, additionally, they plan to spend more time in class to discuss the main material covered in the text. Admittedly, this criticism regarding lack of traditional lecturing should have been anticipated as it is, for example, discussed in [6]. On the other hand, it should be pointed out that in the previous running of the course, using traditional lecturing, the attendance levels in lectures gradually decreased as the course progressed.
- **Pressure of submission**. The second most common criticism that appeared in the first survey relates to the pressure many students felt in the submission of the programming assignments in class (recall that these assignments counted towards the course grade). Many students felt that they did not have enough time to finish the assignments (even with four more hours after class). It has not yet been decided if or how the instructors react to this issue. One option might be to make the class programming assignments not count towards the course grade, but that indeed might discourage the students to work on the assignments in class and leave after the quizzes.
- **Projects are too hard**. The third most common criticism found in the first survey was that many students felt that the programming projects were too hard. Furthermore, after the final exam, many students felt that the instructors had established unrealistic expectations regarding the students' programming skills. While the instructors do not agree with this view, it is consistent with the view expressed in [7], and it will be discussed further in the DCS.
- **Students with previous programming background**. As mentioned in Section IV.B, 67% of the students in the course were novice programmers. Most of the students with programming background are male students and, according to our first survey, a substantial number of the female students (who were novice programmers) felt intimidated by these male students during TBL. Another "problem" with the students with programming background is that they feel bored in the first weeks of the course, because the material is too elementary for them. The instructors had hoped that the students with programming background would help, and indeed teach, their teammates, particularly in the programming assignments in class. This seemed to be working out in the very first few weeks, but, as the time passed, many of them lost interest in helping their teammates. The instructors would like to "solve" these problems, when teaching the CS1 course next time, by making a special section for the students with previous programming background, as done by various other institutions [14]. In this special section, it will thus be possible to present these students with more challenging programming problems, and, hopefully, get rid of the intimidation problem.

VI. Conclusion

In this paper, we presented an experience report about using FC and TBL in a CS1 course. We presented the motivation for restructuring the course, the details of the implementation, the results of two surveys carried out, and the outcome of four exams. We discussed the results in detail, presented what actions were taken during the course period and what changes are intended to be made to it when it will be offered in the future.

The surveys showed that up to 33% of the students were displeased with the course – in particular the female students. About 60% of the students liked the TBL in class, but about

[4] All students are able to take the retake exam. However, generally, the students showing up in the retake exam are the ones who either failed or missed the final exam.

half of the students felt that the course lacked traditional lectures. It was surprising that 44% of the students never or seldom read the textbook before class, whereas 74% watched the videos. The instructors were disappointed with the failure rate of about 56% in the final exam, which is the highest compared to previous years. Nevertheless, the instructors were generally pleased with the experience of using FC and TBL in the course and intend to apply these teaching methods again in an improved version of the same course next year.

It should be clear from our description of the implementation of the course that various other changes were made to it, in addition to the adoption of FC and TBL. For example, a switch was made to Python from C++, additional lecture hours were introduced, and more people were involved. One may argue that some of these issues affected the outcome. In our opinion, it is unlikely that the switch to Python from C++ had any significant affect. In Section I, we referred to a large study that has shown that the difference in pass rates is not based on the programming language being taught. The two other changes mentioned above, i.e. additional lecture hours and more people involved, were needed to facilitate the implementation of the FC. One would think that additional lecture hours would be beneficial to the students, but, currently, we are not in a position to verify that. Having different instructors involved in different sections of a course may result in different performances between students in the different sections. Indeed, in the final exam we noticed that one of the sections "outperformed" the others by some margin. However, it is possible, for example, that a higher ratio of students with previous programming background were part of this particular section.

From the experience gained, questions have arisen about the teacher's role and responsibility. In our case, the DCS decided to change the implementation of a course using recognized pedagogical approaches, i.e. making the students take an active part in the learning process and work on projects in teams during classes. The teachers were pleased with the change, but a significant part of the students was displeased. Moreover, the failure rate was very high. How should the teachers react to this situation? Should they continue and run the course again using the same format, should they give up and go back the traditional lecture and laboratory-based format, or should they consider the students complaints and improve the implementation? If we agree with the last option, new questions arise. How much of the complaints from students should be taken into account, and how will the improvements affect the learning outcomes and the attendance in classes?

REFERENCES

[1] Saleh Alhazbi. 2016. Using flipped classroom approach to teach computer programming. In *Proceedings of the IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE).* Bangkok, Thailand.

[2] Cynthia J. Finelli, Kevin Nguyen, Matthew DeMonbrun, Maura Borrego, Michael Prince, Jenefer Husman, Charles Henderson, Prateek Shekhar, and Cynthia K. Waters. 2018. Reducing Student Resistance to Active Learning: Strategies for Instructors. *Journal of College Science Teaching*, 47, 5 (2018), 80–91.

[3] Krisztina V. Jakobsen and Megan Knetemann. 2017. Putting Structure to Flipped Classrooms Using Team-Based Learning. *International Journal of Teaching and Learning in Higher Education* 29*,* 1 (2017), 175–185.

[4] Antti Knutas, Antti Herala, Erno Vanhala and Jouni Ikonen. 2016. The Flipped Classroom Method: Lessons Learned from Flipping Two Programming Courses. In *Proceedings of the 17th International Conference on Computer Systems and Technologies*. Palermo, Italy.

[5] Wanda M. Kunkle and Robert B. Allen. 2016. The Impact of Different Teaching Approaches and Languages on Student Learning of Introductory Programming Concepts. *Transactions on Computing Education* 16, 1 (2016), 3:1–3:26.

[6] Patricia Lasserre. 2009. Adaptation of Team-based Learning on a First Term Programming Class. In *Proceedings of the Conference on Innovation and Technology in Computer Science Education (ITiCSE '09).* Paris, France.

[7] Andrew Luxton-Reilly. 2016. Learning to Program is Easy. In *Proceedings of the Conference on Innovation and Technology in Computer Science Education (ITiCSE '16).* Arequipa, Peru.

[8] Ásrún Matthíasdóttir and Hrafn Loftsson. 2019. Flipped Learning in a Programming Course: Students' attitudes. In *Proceedings of the 15th International CDIO Conference*. Aarhus, Denmark.

[9] William F. Punch and Richard Enbody. 2017. *The Practice of Computing Using Python* (3rd. ed.). Pearson Education, New York, NY.

[10] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education* 13, 2 (2003), 137–172.

[11] Jason H. Sharp. 2014. Journey Toward a Flipped C# Programming Class: An Experience Report. In *Proceedings of the Information Systems Educators Conference* . Baltimore, MD, USA

[12] Manoj D. Souza and Paul Rodriques. 2015. Investigating the Effectiveness of the Flipped Classroom in an Introductory Programming Course. *The New Educational Review* 40, 2 (2015), 129–139.

[13] Christopher Watson and Frederick W.B. Li. 2014. Failure Rates in Introductory Programming Revisited. In *Proceedings of the Conference on Innovation and Technology in Computer Science Education (ITiCSE '14)*. Uppsala, Sweden.

[14] Chris Wilcox and Albert Lionelle. 2018. Quantifying the Benefits of Prior Programming Experience in an Introductory Computer Science Course. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. Baltimore, Maryland, USA.