APPLYING "FLIPPED CLASSROOM" METHODOLOGY IN COMPUTER SCIENCE COURSES

John J. Cupak[*]
Lecturer, Department of Mathematics & Computer Sciences, Rivier University
and
Vladimir V. Riabov, Ph.D.[**]
Professor & Department Coordinator, Department of Mathematics & Computer Sciences, Rivier University

Keywords: *"Flipped Classroom", quizzes, notebooks, computer labs, mini projects*

Abstract

The "Flipped Classroom" provides students the opportunity to review material and take notes before class time, but must be coupled with quizzes to assess the student's understanding and review of student notebooks to ensure material is actually reviewed before class. This methodology allows the instructor and students concentrate entirely on the in-class activities, such as reviews of case studies, lab exercises, and mini project developments. The authors share their experience of motivating and stimulating the students to earn benefits from this approach.

1. Introduction

Computer science courses are information-intensive and skill-based. Much of the course time is spent by the students learning the vocabulary of the course as well as practicing the skills required to master the course requirements. Learning the terminology of the course requires not just being able to regurgitate the definition of terms, but also to understand the concepts and application of the terms as applied to the context of the course. Skills for the course are typically practiced and demonstrated through weekly exercises and assignments.

Rivier University programming courses (e.g., "Computer Science Fundamentals", "Java Programming", "Algorithms", and "Professional Seminar") are fourteen-weeks long, and other courses (e.g., "Operating Systems", "Object-Oriented System Analysis and Design", "Web Development", "Networking Technologies", "Computer Security", and "Software Quality Assurance") are "hybrid" seven-weeks long. In both types of the courses, students are expected to complete out-of-class assignments. Prior to the 2014-2015 academic year, the courses were conducted in typical lecture format with post-class assignments. Lectures consisted of presentations of slides supplemented with "chalk-talk" explanations and examples. In class, it was observed that students took notes and their subsequent performance on homework assignments, quizzes, and exams revealed that they were grasping the material and were able to perform as expected.

At the beginning of the 2014-2015 academic year, Rivier University experienced a large influx of international students for whom English was not their native language. In addition, most of these students would come to class without textbooks, note paper, or writing implements. The majority of the students would sit and listen to the lecture without taking notes, and were unable to demonstrate their understanding of course terminology on quizzes or complete homework assignments correctly.

In an attempt to overcome the apparent inability of the students to understand the course terminology or to complete the homework assignments without errors, Professor Cupak decided to explore the "Flipped Classroom" format [1] in his CS597 "Multimedia and Web Development" 7-week

ISSN 1559-9388 (online version), ISSN 1559-9396 (CD-ROM version).

hybrid course. Course lecture and reference material for Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and for a cross-platform JavaScript library jQuery [4] was posted on the Canvas™ Learning Management System [2] for each class meeting. Additional reference material, such as "quick guides" were provided online via Canvas™ and the instructors' websites, as well as handed out in class. Students were expected to review the material before class and arrive in class prepared to engage in discussions and to be able to demonstrate their knowledge when called upon. Short, random, multiple-choice quizzes were conducted at the beginning of each class meeting to assess the students' understanding of key concepts. During the class meeting time, topical examples were provided, and in-class exercises were assigned and worked on before assigning homework.

A similar "Flipped Classroom" approach was also applied by Dr. C. Thomas Wilkes in his CS505 "Computer Science Fundamentals" and CS554 "Operating Systems" classes. Being the alumni of Georgia Tech, he used the pedagogical approach and the CS online course concepts promoted there via the Online Master of Science in Computer Science program [3].

2. Assessment of Students' Preparedness for the Classes

Due to the limitations of the Canvas™ LMS, there was no way to determine if the students had actually reviewed the material before each class in the CS597 "Multimedia and Web Development" 7-week hybrid course other than their quiz grades. While it appeared that the students were getting good quiz grades, a "review" during the week-6 class in preparation for the final exam during the week-7 class revealed that most of the students were unable to recall key concepts or provide correct solutions to problems. Furthermore, during the final exam in which the students were required to create a web page when given the requirements, over half of the students turned in identical solutions that had been copied from one student's exam paper.

Given the opportunity to conduct a 14-week CS557 "Algorithms" course, Prof. Cupak introduced "discovery learning" in every class in which students must recall and provide answers to problems posed during class. Initially, the students were hesitant to respond. This was attributed to their less-than-perfect ability to communicate English. However, the quiz scores were abysmally low. Students were not able to provide correct response to key information, and often provided answers which had no relationship to the questions asked.

Table 1. Quiz Scores

Scores	Quiz 02	Quiz 03	Quiz 04	Quiz 06	Quiz 08	Quiz 09
>= 5	3	4	1	6	5	6
< 5	5	4	7	2	3	2

Quizzes given at the beginning of each class were designed to evaluate the students' understanding of the material before instruction began and was used to by Prof. Cupak to focus the lecture on topics poorly understood. By the end of the third quiz (Quiz 04), it was observed that the students were having problems understanding the material before class. One student asked if the quiz could be given "at the end of the class." Instead, Prof. Cupak provided information on using the "Cornell Method" for taking notes, and informed the students that a review of the students' notebooks during the closed-book quiz at the beginning of each remaining class would be counted as part of their grade. With the incentive to take notes on the "Flipped Classroom" material before class, the quiz grades improved, as shown in Table 1.

The CS557 “Algorithms” mid-term examination consisted of five parts on terminology, documentation, and analysis. With the exception of one student who neglected to answer a question and another student who was unable to estimate the running time for an algorithm, the grades ganged from 76 percent to 92 percent. It was evident that the requirement for students to have a notebook with their own notes improved their ability to understand the exam questions and answer them.

3. Running In-class Labs

The "Flipped Classroom" methodology allows the instructor and students concentrate entirely on the in-class activities, such as reviews of case studies, lab exercises, and mini project developments. Every class starts with the “warm-up” exercises that introduce an unusual non-trivial topic that will be covered in-depth in the class later. After these "warm-up" exercises, the instructor offers a discussion on the main topic, asks students for a feedback on lecture materials, and their arguments on selecting a competitive strategy for the problem analysis and development. These discussions help students get the main point of the class session and become focusing on it in class activities. A several examples of the "warm-up" exercises can be found in [5].

The main goal of labs in the “Software Quality Assurance” course is to introduce software quality metrics and help students build their individual skills of code analysis, testing, and redesign to improve code quality and enable possible reuse in other projects.

The first lab deals with implementation of the structured testing methodology offered by McCabe [6, 7]. The approach is based on graph theoretical complexity-measuring techniques in code studies and control of program complexity. Using the experimental results of Miller [8], McCabe suggests that code modules approach zero defects when the module cyclomatic complexity is less than 10. During online lectures (prior to classes), the instructor provided an overview of the graph-based complexity metrics and the results of his systematic metric analyses of software for two industrial networking projects [9]. In class, following the lab assignments, students explored the McCabe ™ IQ tool and used it to perform metric analyses of several codes by applying cyclomatic complexity (*v*), essential complexity (*ev*), module design complexity (*iv*), system design complexity, and system integration complexity metrics [7] in order to understand the level of complexity of a code module’s decision logic, the code’s unstructured constructs, a module’s design structure, and the amount of interaction between modules in a program, as well as to estimate the number of unit and integration tests necessary to guard against errors.

The second lab was designed to introduce students to the comparative analyses of algorithm implementations in different languages (FORTRAN, C, C++, Java and some others). Before class, students read several chapters from the Halstead’s book “Elements of Software Science” [10] on the code analysis strategies. Following Halstead’s procedures [10], students identified all operators and operands, their frequencies, and estimated the program length, vocabulary size, volume, difficulty and program levels, the effort and time amounts to implement and understand the program, and the number of delivered bugs (possible errors), *B*. They compared their findings with values calculated by using SQA tools (McCabe™ IQ, Java Source Metric™, CCCC™, Source Monitor™, and COCOMO-II™), and found that the results are sensitive to the programming language type (procedural or object-oriented). In particular, students found that efforts to implement and understand the program were higher for procedural languages (FORTRAN and C) than for the object-oriented language (Java), even for simple algorithms, like Euclid’s algorithm for calculating the Greatest Common Divisor. They also found that large C/C++ source files [9] contain more actual errors than the number of delivered bugs (*B*) estimated by using Halstead’s approach [10].

The third lab was developed to help students identify clusters of object-oriented metrics that would better describe the major characteristics of object-oriented systems (properties of classes, polymorphism, encapsulation, inheritance, coupling, and cohesion) implemented in computer code written in C++ and Java. Before this class, students read and analyzed the case studies reviewed by Rosenberg [11]. Running the lab, students identified some specific object-oriented metrics (Weighted Methods per Class, Response from a Class, Lack of Cohesion between Methods, and Coupling between Objects) that are the important factors for making a decision about the code module/class re-usability [12].

The fourth, and last, lab assigned to students required to identify major factors that forced programmers to change the code [9] in the project redesign efforts. After analysis of the project software (about 300,000 lines of C-code), 271 modules of the old Code Release 1.2 [9] were recommended for redesign. The re-engineering efforts resulted in the deletion of 16 old modules and in the addition of 7 new modules for the new Code Release 1.3. Analyzing the deleted modules, students found that 7 deleted modules were unreliable ($v > 10$) and 6 deleted modules were unmaintainable ($ev > 4$). Also, 19% of the deleted code was both unreliable and unmaintainable. Moreover, all new modules were reliable ($v < 10$) and maintainable ($ev < 4$).

4. Developing Mini-Projects in Class

The "Flipped Classroom" approach allows the instructor create a class environment for supporting effectively project developments. Instructors in the CS552 "Object-Oriented System Analysis & Design" core course encourage students to read the selected chapters and overview the case studies from the textbook [13] before every class, and concentrate mostly on the application of the Unified Modeling Language (UML) methods in the mini-projects developed by the students in the computer-lab classrooms.

The instructors start using a modified approach towards teaching the course concepts by utilizing Cloud-based UML drawing tools – Lucidchart™ and Cacoo™ – rather than traditional tools like ArgoUML™ and Visual Paradigm™. The rationale for such a move was to give greater line-of-sight visibility into the progression of students' work on team projects, and to reduce the amount of cheating (or "copying") that had been prevalent in earlier editions of the course. Student teams were required to share their Cloud workspaces with the instructor, so he/she was able to see the progression of their work. There was "no magic" – students knew that elements from other tools or students' previous submissions could not be inserted into their diagrams. Further, the checkpoint reviews allowed the instructor to give direct feedback to each project team every week so that they could focus on understanding the reasons of their mistakes. Finally, the checkpoint reviews gave the instructor insight into topics that he/she needed to review a second time for the class.

After two semesters of utilizing Cloud UML drawing tools and integrating weekly team checkpoints into the CS552 curriculum, Dr. K. Bagley, the CS552 course instructor, believes that his teaching effectiveness and his students' learning performance (and retention) has improved. Plagiarism and cheating has also decreased in his CS552 "Object-Oriented Design" class, simply because it is more difficult to copy-paste existing material. Finally, the Cloud-based solutions are "ready-made" – the zero-footprint installation means that no student has any excuses for not having or accessing the software from any location. No more "my computer didn't work" (or other "the dog ate my homework") stories apply here since both the tool and the associated UML artifacts produced are in the Cloud.

The similar approach was used by other instructors in developing new types of lab and mini-project assignments that motivate students to work creatively and avoid cheating and plagiarism. For example, unique code samples were offered by Dr. Riabov for the complexity code analyses in his CS680

“Software Quality Assurance” course [14]; and non-traditional capstone project topics have been offered by instructors to students in the CS699 “Professional Seminar in CS” course. The number of misconduct cases dropped by the factor of 8-12 in all these courses.

5. Working on Capstone Projects

The effectiveness of the “Flipped Classroom” methodology is mostly pronounced in students’ works on their individual capstone projects. Every student in the last core course, CS699 “Professional Seminar in CS”, makes in-class presentations of their reviews of two articles from the peer-reviewed academic journals (available, for example, from the ACM and IEEE Digital Libraries) that inspire the students to explore modern computing technologies and develop their capstone projects. The articles are available for all students prior to the presentations, and students’ contributions to the in-class discussions on the article contents are encouraged and graded.

Following the guidelines on the research project development [15], students work on various project stages, including the Project Plan, Feasibility Analysis, Functional Analysis, Functional Test Plan, System Design, System Integration Test Plan, System Implementation (Code Programming), Module/Unit Test Plan, System Prototyping, and Testing. Working on their capstone projects, students utilize knowledge and skills acquired during the studies in the Computer Science program. All the software tools that students used for labs and mini-projects in various CS classes are available to them via the Virtual Desktop Infrastructure (for using in the computer-lab classrooms or remotely from the home computers). The course instructors assist the students, who need help, during the classes or during their office hours. Finally, at end of the course, every student present a demo of the developed computing system and submit a project report with full documentation (burnt on a CD). Striving to “go an ‘extra mile’”, students prepare high-quality project reports for including them into the job/internship-search portfolios, for promotion in the company, for presenting the papers at the conferences, and publishing the project summaries in academic journals.

6. Conclusion

In conclusion, the "Flipped Classroom" reduces the classroom "lectures" and requires that students review the material before class. Reviewing and grading notebooks by course instructors ensures that students are looking at and taking notes from the pre-class material. Quizzes conducted at the beginning of each class help identify material less completely understood and allow the course instructor the flexibility to tailor the class to meet the needs of students. Finally, experience with international students has led us to the conclusion that the material must be repeated often before the students will absorb and understand it.

The "Flipped Classroom" methodology allows the instructor and students concentrate entirely on the in-class activities, such as reviews of case studies, lab exercises, and mini project developments. This approach contributes to the success of students’ efforts in development of high-quality capstone projects that become a valuable part of their job/internship-search portfolios. ■

References

[1] Brame, C. Flipping the Classroom. Vanderbilt University Center for Teaching, 2013. Retrieved November 5, 2017, from: http://cft.vanderbilt.edu/guides-sub-pages/flipping-the-classroom/

[2] Canvas: Instructure, Producer, 2013. Retrieved November 5, 2017, from https://www.canvaslms.com

[3] George Tech Online Programs. Online Master of Science in Computer Science, 2017. Retrieved November 5, 2017, from http://www.omscs.gatech.edu/

[4] jQuery, 2017. Retrieved March 29, 2017, from http://jquery.com/

[5] Riabov, V. V. Challenging Projects and Virtual Labs in Web-enhanced Networking Technology Classes. In *Journal of Computing Sciences in Colleges* 21(6), 88-99, June 2006.

[6] McCabe, T. J. A Complexity Measure. In *IEEE Transactions on Software Engineering* SE-2(4), 308-320, December 1976.

[7] Watson, A. H., McCabe, T. J. Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric, *NIST Special Publication*, No. 500-235. Gaithersburg, MD: National Institute of Standards and Technology, September 1996.

[8] Miller, G. The Magical Number of Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. In *Psychological Review*, 63 (2), 81-97, 1956.

[9] Riabov, V. V. Networking Software Studies with the Structured Testing Methodology. In *Computer Science and Information Systems*. Athens, Greece: ATINER, 2005, pp. 261-276.

[10] Halstead, M. H. *Elements of Software Science*. New York: North Holland, 1977.

[11] Rosenberg, L.H. Applying and Interpreting Object-Oriented Metrics. In *Proceedings of the Tenth Annual Software Technology Conference, April 18-23, 1998.* Salt Lake City, UT, 1998.

[12] Riabov, V. V. Tools and Methodologies for Teaching Online Computer-Science Courses in LMS Environment. In: *Learning Management Systems and Instructional Design: Best Practices in Online Education*, edited by Yefim Kats, IGI Global, 2013, pp. 144-171.

[13] Dennis, A., Wixom, B. H., & Tegarden, D. *System Analysis & Design, UML Version 2.0: An Object-Oriented Approach*, (4th ed.). Hoboken, NJ: John Wiley & Sons, Inc., 2013.

[14] Riabov, V. V. Methodologies and Tools for the Software Quality Assurance Course. In *Journal of Computing Sciences in Colleges* 26(6), 86-92, June 2011.

[15] Booth, W. C., Colomb, G. G., & Williams, J. M. *The Craft of Research*, (4th ed.). Chicago, IL & London, UK: The University of Chicago Press, 2013.

[*] **PROF. JOHN J. CUPAK** has been teaching Computer Science courses at Rivier University since spring 2014. He is a member of the Institute of Electrical and Electronic Engineers (IEEE) and the Association for Computing Machinery (ACM). He has a B.A. in Biology with a mathematics minor from the Utica College of Syracuse University, and a M.S. in Computer Science from the State University of New York at Albany (SUNYA). Prof. Cupak has worked as a software engineer for over 40 years, and brings his experience in computer-system design, development, documentation, and testing to his courses. An avid educator, he researched and implemented the first "Flipped Classroom" concept in the Department of Mathematics and Computer Science at Rivier University. He notes that "Teaching is a learning experience - for the student and the teacher as well."

[**] **Dr. VLADIMIR V. RIABOV**, Professor of Computer Science and Department Coordinator at Rivier University, teaches Algorithms, Networking Technologies, Computer Security, Software Engineering, Software Quality Assurance, Object-Oriented System Design, System Simulation and Modeling, Numerical Methods, Introduction to Computing, and Professional Seminar in Computer Science. He received a Ph.D. in Applied Mathematics and Physics from Moscow Institute of Physics and Technology and M.S. in Computer Information Systems from Southern New Hampshire University. Vladimir published about 130 articles in encyclopedias, handbooks, journals, and international and national conference proceedings, including *The Internet Encyclopedia*, *The Handbook of Information Security*, *The Handbook of Computer Networks*, *International Journal of Computers and Structures*, *Journal of Spacecraft and Rockets*, *Journal of Aircraft*, *Journal of Thermophysics and Heat Transfer*, *Congress Proceedings of International Council of the Aeronautical Sciences*, *International Symposia on Rarefied Gas Dynamics and Shock Waves*, *International Conferences on Computer Science and Information Systems*, *International Conferences on Technology in Collegiate Mathematics*, *Conferences of American Institute of Aeronautics and Astronautics*, and others. He is a senior member of ACM and AIAA, and a member of IEEE and MAA.