A Flipped Classroom Experience Teaching Software Engineering

Nicolás Martín Paez
Department of Science and Technology
Universidad Nacional de Tres de Febrero
Saenz Peña, Buenos Aires, Argentina
nicopaez@computer.org

***Abstract*— New teaching approaches like the flipped classroom are an interesting alternative to educate new generations but they represent new challenges for teachers. This paper describes our experience re-designing our classes and study materials in order to adopt a flipped classroom approach combined with some other non-traditional teaching techniques. This experience is focused on the Software Engineering course at Universidad Nacional de Tres de Febrero. In this paper we share details of our strategy, the positive results we obtained and the concerns we still need to address.**

Keywords-component; experience report; flipped classroom; education; agile; software engineering

I. INTRODUCTION

Facebook, Instagram and WhatsApp are some the applications that run on our student's smartphones. And those Smartphones are like extensions of the body, the students never stay away from them long. Best-case scenario, they mute them. This is a significant challenge for teachers trying to engage them meaningful learning experiences. Furthermore, the more students they have in the classroom, the greater the challenge.

The flipped classroom [1] and the teaching "from the back of the room" [2] approaches may help to overcome this "attention challenge". Both approaches promote learner-centered activities, making the classes more interactive.

This paper describes our experience at Universidad Nacional de Tres de Febrero using these approaches to teach Software Engineering. The paper is organized in three main sections: Section II describes our context, Section III explains in detail what we are doing and Section IV describes our results and perspectives for future work.

II. CONTEXT

Universidad Nacional de Tres de Febrero is a small, young university located very near to Buenos Aires City. The Computer Engineering Program started in 2008. Software Engineering topics are split into 3 main courses: the first one is focused on Requirements, the second on Design and Architecture, and the third on all the remaining concerns of Software Engineering such as Planning, Estimation and Processes. This third course also acts as an integrator because students apply everything they have learned in the previous courses. This paper focuses on the strategy we use in this third course, which is part of the fourth year of the program, which lasts 5 years. Students taking it have already developed strong programming skills. This course is organized in one weekly 4-hour class. The amount of students per semester is approximately 15, the classroom is large for that amount of students and it has a fixed projector. The seats are individual and are not fixed, which means they can be rearranged at any time.

About 50% of the students are already working in the software industry. This poses another challenge because they have already been exposed to the industry whilst their classmates find them completely new.

III. TEACHING STRATEGY

With just one class a week it is difficult to compete with the invasion of social networks and other smartphone notifications. It is even harder to keep students interested during a 4-hour class with traditional teaching dynamics based on teachers conveying information. This situation led us to search for a different approach for our course. After doing some research and having participated in some Massive Online Open Courses and also in some other blended courses, we ended up with some ideas to experiment.

We decided to try the flipped classroom approach [1] which proposes a reversal of the traditional class flow. In the typical class flow the "in-class" time is dedicated to a lecture where the teachers explain the concepts and then distribute some work/exercises which students are to complete at home. The flipped approach proposes the students "carry out" the lecture at home (for example watching a video or reading an article) and then the "in-class" time is used for interactive activities such exercises and discussions. This approach has already been used in Software Engineering courses [3][4] and as stated by Kerr [5], several studies have reported high student satisfaction and increased performance in flipped classroom environments.

At the same time, we decided to organize the "in-class" time using "from the back of the room" teaching techniques. These brain-friendly techniques are specially designed for adult training and are aimed at generating immersive, collaborative and active learning experiences [2].

978-1-5386-2795-2/17 $31.00 © 2017 IEEE
DOI 10.1109/SECM.2017.6

Finally, we also wanted to use some practices like Coding Dojos [6] and Coding Katas [7] to teach the practice of more technical concerns such as Test-Driven Development and Refactoring.

Introducing all these new techniques required us to rethink the whole course structure, which we re-organized into several chapters, each of them including a set of resources: videos, readings, exercises, interactive activities and quizzes. Every week the students are required to complete a set of mandatory assignments, some of them individual and some working in groups.

In the first class of the course we explain to the students how the course is organized and we are very explicit about the time they will they need to work at home. In addition to the 4 hours of the weekly class, we estimate students will require between 4 to 6 hours working at home. This dedication is stable along the whole course.

Taking advantage of the classroom facilities, during the classes we re-arrange the seats to form a circle which we think reinforces the idea that we are a team of peers and that we, students and teachers, are all learning.

Another major change we implemented was in the evaluation strategy. We don't use exams to evaluate students, we use a multi-factor evaluation strategy instead that is based on 3 main dimensions: the weekly assignments, the "in-class" participation and the final group assignment. The weekly assignments provide students with frequent feedback on their performance.

All these changes represented a big improvement in the student´s learning experience, giving them a deeper insight into course´s different topics. Yet implementing these changes required significant effort on behalf of the teachers. This included the initial study of these new pedagogical approaches, the re-design of all classes and their study material and also the weekly assignment follow-ups to provide personal feedback to each student.

A. Flipped classroom materials

The flipped classroom approach led us to create new course materials and update existing ones. At the same time to share materials and also to extend the interaction with students beyond the "in-class" time, we started using a virtual classroom. Given our university didn't have this kind of tool, we selected a platform called Eliademy [8] which provides all the classic tools expected of any virtual classroom:

- Hierarchical structure to organize classes
- File Sharing
- Forum
- Grading book
- Quizzes
- Notifications of course updates

Eliademy is offered as a service so we do not have to worry about uptime, backup and any other operational concerns.

Regarding the new materials we generated, the videos played a key role in the dynamics of our course. We have two different kinds of videos: lecture videos and "how-to" videos. Lecture videos are basically recorded lectures; they present a topic that will be discussed in the next class. They act as an introduction to the topic. The "how-to" videos shows how to use specific tools. For example, as shown in figure 1, we have videos that show how to use Git. We also have videos about Jenkins and Cucumber, among others. Most of the lecture videos are produced by us. The "how-to" videos are a mix of third-party publicly available videos we collected from the web and some others created by us. In all cases we tried to keep the duration of the videos under 12 minutes. Those we create are recorded and produced with Camtasia Studio [9]. We then publish them on YouTube and we link them back to the virtual classroom. The videos are very welcomed by the students because they can watch them at any time using their smartphones and given the videos are short, they don't need to dedicate a lot of their time to them. Students may watch a video while travelling in the train and they are able to watch it again later if they want to review something.

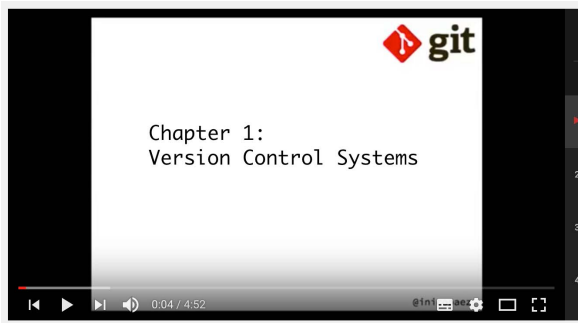

Figure 1. Video that explains the usage of Git tool[1]

Other types of resources we use are papers, book chapters or blog posts. Each of these readings has some key points that we expect the students to understand. Of course that just asking the students to read the documents does not guarantee that they will discover those key points. In the past, we have had difficulties with students not being able to identify the key points from the readings. So to guide them and transform the passive reading activity into an active challenge, each reading has now been associated to a quiz in the virtual classroom which must be completed. Each quiz consists of a set of multiple-choice questions (typically between 3 and 5). Figure 2 shows an example of a multiple-choice question. The fact that they are multiple-choice allows us to configure the virtual classroom platform to provide feedback to the students instantaneously, as soon as they complete the quiz. Students confirm that quizzes help them to focus when reading and at the same time we suspect that having to complete these quizzes "forces" them to read. In the past, even when reading the documents were mandatory, sometimes students didn't read them at all. It is important to mention that even when a quiz has at maximum 5 questions, we actually create at least 8. This way, each time the quiz is opened it contains different randomly selected questions. The possible answers are also shown in a random

1 The text in all figures has been translated to English for this paper but all the original materials are in Spanish.

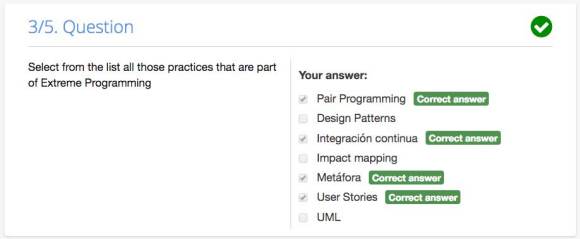

Figure 2. Multiple-choice question for the Extreme Programming Quiz

order. This makes it more difficult in case the students want to copy. Creating the quizzes required some time but it is a one-time effort and it helps us to save lot of time because grading quizzes is handled by the platform (this is only possible because questions are multi-choice). Both, readings and videos are used in two different ways: some are used before the class and others are used afterwards. This depends on the specific topic and the specific way the corresponding class is designed.

Let's consider the topic "User Stories" to analyze how we articulate classroom activities with home activities. Firstly, we ask students to read an article and watch a video at home. Then the next class we perform a User Story Mapping activity whose results are shown in figure 3. Finally, after the class, we ask the students to complete a quiz about User Stories.

B. Teaching by example

The curriculum of this course includes agile practices and some of those agile practices are applicable in several contexts beyond software development. A good example of this is retrospectives, which helps to detect process improvement opportunities. Throughout the courses we teach retrospectives techniques and we also use them. During the semester we perform two retrospectives activities in order to detect improvement opportunities. This gives the students an explicit opportunity to provide feedback. At the same time, we as teachers have the opportunity to make adjustments to the way we are running the course. In addition to this, students not only read about retrospectives techniques but they are also part of the course retrospectives which provides them with a better understanding of such techniques. Teachers and students share the same opinion regarding retrospectives: they are very valuable and the effort they require is minimal, it is just about 1 hour of "in-class" time.

Figure 3. Some students showing the result of a Story Mapping activity

Something that is worth mentioning is that sometimes the output of the retrospectives requires some modifications in the course plan which may imply some additional effort on the teachers' behalf. Figure 4 shows the result of a retrospective activity where students identify different dimensions of the course and rank them using the glad, sad and bad faces.

In addition to retrospectives, we run a "lessons learned" activity at the end of the semester to get overall feedback on the course for the next semester. This activity may not be valuable for students but it is very valuable for teachers in order to be able to enhance the course for the next year.

Upon finalization of the course, we invite two visitors to share their experiences working in the software industry. We try to mix in visitors from different contexts to provide a more realistic perspective, typically one of them from a start-up or small company and the other from an international corporation. These visits are very welcomed by students and are great motivation for them to see how what they study in our course is aligned with what the visitors share.

C. Continuous practice

We try to run the course as a project in the sense that we expect students to work every day and keep a sustainable pace. This may sound trivial but some students are used to just attending classes and only sit down to study a couple of days before the exam. For those kinds of students our proposal turns out to be very challenging.

Being able to keep a sustainable pace is one of the concerns of some Software Engineering techniques. To help students master these techniques we ask them to practice by planning and estimating their weekly assignments. Once they have completed them, we also ask them to track the actual times required and compare it with their own estimates. At the beginning of the course their estimates are usually very inaccurate but towards the end of the course, they tend to come reasonably close. To implement this estimate-plan-track dynamic each student uses an online spreadsheet that is

Figure 4. Result of a retrospective activity

shared with the teachers. As a result, we can see how they evolve in their estimation skills. Students have reported that doing this has increasingly helped them to organize themselves, not only for our course but also in other courses.

D. Real world tools

Software Engineering practices require tools of two main types: management tools (such as bug tracker) and development tools (such as programming language or continuous integration system). In both cases, we use tools that are commonly used in the software industry which help to narrow the gap between academy and industry. At this moment our stack of tools is very similar to one the proposed by Fox [10] and it includes:

- Programming language: Ruby
- Specification: Gherkin/Cucumber
- Build Tool: Rake
- Version Control System: Git (provided by GitHub)
- Continuous Integration/Deployment server: Jenkins (provided by CloudBees) and Travis
- Cloud hosting: Heroku
- Planning/Tracking tool: Trello or Meister Task

Making all these tools work together may require some specific knowledge and experience and we don't want our students to spend hours doing troubleshooting to make them work. So as to avoid these matters, we provide them with project templates. This way they can concentrate on solving the assignments without having to deal with the specific issues of each tool/framework. At the same time, to simplify environment setup, last semester we started using Vagrant [11], which really simplifies installations and configurations of tools by creating Virtual Machines.

Project templates and Vagrant configurations are great for students because they reduce the time taken up by installation and configuration tasks. On the other hand, teachers need to invest some time to prepare and test them. In our specific case we were already familiar with Vagrant and Ruby so creating all these artifacts didn't take too much time.

IV. Results and Perspectives

We have worked with this strategy in 2 instances of the course, and we got very good results and very positive feedback from the students. Previous instances of this course were taught by other teachers that also used a non-traditional approach. Unfortunately, we do not have metrics to compare those instances with our current approach.

In our two instances of the course, all students passed. In the first instance of the course only some of the techniques described in this paper where applied. In the second instance of the course we incorporated the feedback and lessons learned from the first instance and the result was what has been described here. Table 1 suggests better learning from the students in the second instance versus the first.

The videos were recognized as excellent materials by the students. They suggested us to add some more videos about topics that are currently studied with readings. Having the videos published in YouTube allows students to make comments on the videos and it also provides the teachers with some statistics.

	First instance	***Second instance***
Student's final mark	7.9	9
Student's course evaluation	7.5	8.4

TABLE I. Comparison of two intances of the course

Regarding the flipped classroom and the general teaching strategy of the course, students considered it more effective and engaging than the traditional lecture-based approach which they have experimented in other courses. It is worth mentioning that this feedback was collected in two different informal ways: in the retrospectives performed during the course and with one specific question in the general evaluation survey filled-in at the end of the course. Some results of that survey are shown in the last row of table 1.

Beyond the positive results, there is an item that didn't do so well and that requires improvement: grading of weekly assignments. Even when the virtual classroom manages quiz grading automatically, there are still many assignments that require human intervention (mainly technical assignments that include programming). These imply specific feedback so the students can understand and learn. In both instances of the course we had important delays in grading those types of assignments. These setbacks were because the marking activity took much longer than the expected when we first designed the assignments.

We understand our approach as a particular implementation that merges techniques and dynamics previously developed by others. We just identified those techniques and customized them for our particular context, but we think the conceptual approach is absolutely replicable in other contexts with a similar number of students. As mentioned before, we have approximately 15 students per semester and we are two teachers. We believe our approach could work well in a course of around 20 students. Larger courses may require some changes in classroom activities and would also mean more effort from teachers to mark weekly assignments.

We will continue with this strategy in 2017, applying some minor adjustments based on our 2016 experience and we will also experiment with some gamification techniques [12] to evaluate students. Additionally, for future changes, we will implement an experimental approach in order to be able to measure the effectiveness of the changes.

References

[1] J- Bergmann, Flip your classroom: Reach every student in every class every day, International Society for Technology in Education, 2012

[2] S. Bowman, Training from the back of the room!: 65 ways to Step aside and let them learn, Pfeiffer, 2008

[3] G. Gannod, J. Burge and M. Helmick, "Using the inverted classroom to teach software engineering," *2008 ACM/IEEE 30th International Conference on Software Engineering*, Leipzig, 2008, pp. 777-786. doi: 10.1145/1368088.1368198

[4] E. Choi, "Applying inverted classroom to software engineering education." *International Journal of e-Education, e-Business, e-Management and e-Learning* 3.2, Singapore, 2013, pp.121-125.

[5] B. Kerr, "The flipped classroom in engineering education: A survey of the research," *2015 International Conference on Interactive Collaborative Learning (ICL)*, Florence, 2015, pp. 815-818. doi: 10.1109/ICL.2015.7318133

[6] E. Bache, The Coding Dojo Hangbook, Emily Bache, 2013

[7] Coding Kata, [online] http://www.codekatas.org/ verified on 2017-01-20

[8] Eliademy, [online] https://eliademy.com/ verified on 2017-01-02

[9] Camtasia Studio, [online] https://www.techsmith.com/camtasia.html, verified on 2017-01-02

[10] A. Fox, D. A. Patterson, R. Ilson, S. Joseph, K. Walcott-Justice, and R. Williams, "Software Engineering Curriculum Technology Transfer: Lessons learned from MOOCs and SPOCs," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-17, March 2014.

[11] Vagrant, [online] https://www.vagrantup.com/, verified on 2017-01-02

[12] K. Kapp, The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education, Pfeiffer, 2012