Iyer, S. et al. (Eds.) (2022). Proceedings of the 30th International Conference on Computers in Education.
Asia-Pacific Society for Computers in Education

Agile Learning as a Solution for Class Absence

Maria Francesca ROIG-MAIMÓ[a*], Ramon MAS-SANSÓ[a]
[a]*Universitat de les Illes Balears, Spain*
*xisca.roig@uib.es

Abstract: In this work, we have presented the results of the application of an agile learning methodology in a Computer Science Laboratory undergraduate course with the goal of decreasing the truancy rates. We have taught a Computer Science Laboratory course of a Telematics Engineering Degree over nine years. The first two academic years we followed a project-based approach. Although we had successfully used this methodology for two years, we noticed a significant increase in the students' absence in class. As an attempt to decrease that truancy rate, we adopted an agile learning methodology for the following seven years. Our primary hypothesis being that if a daily basis work is needed in order to deliver sprints on time, then class assistance should be a requirement to avoid falling behind. As we already expected, students easily went ahead in the subject and their satisfaction, commitment and involvement were greater. Moreover, we realized that truancy rates were lowered in a significant way.

Keywords: agile learning, computer science laboratory, agile methodologies, truancy rate, class absence

1. Introduction

The traditional learning approach, normally modelled as a one-way communication from the lecturer to the student, keeps playing an important role in high education institutions. However, in engineering degrees, due to its inherent practical nature, the project-based learning has been introduced as an alternative methodology to the conventional lectures.

Project-based learning tries to achieve optimal results by putting the student at the center of the learning process, emphasizing his autonomy and recognizing each particular learning style (Thomas 2000). When applied to the teaching of software development, project-based learning can be easily related to the sequential waterfall process, possibly due to its contemporaneity (Matković & Tumbas 2010). In such a model, the students are first proposed a project and then they are asked to analyze the requirements, design the product, implement the software and test the software to finally provide a running version. Each of the stages depends on the correct fulfillment of the previous one.

In 2013-14 we started to teach a Computer Science Laboratory (CSL) course of a Telematics Engineering Degree. The main goal of CSL is the strengthening of the programming skills attained throughout two previous programming courses. Therefore, the proposed program focuses mainly on the development of a computer application in all its stages so that programming skills are reinforced. Software and project management related competences are fulfilled in the curricula of the previous three years, so we mainly must promote the programming aspects in our program. We pursue the achievement of such learning outcome through the analysis, design, coding and validation of a mobile application. With this scenario, applying a project-based learning methodology looked like the most consequent.

The first two academic years of teaching this course (2013-14 and 2014-15) we followed a project-based approach. Although we had successfully used this linear approach for two years, we realized that in such a sequential process the students started the development stage very late. And, if we consider that this stage is the main road to achieve the course's goals, it might not be a suitable approach as a lot of time was devoted to previous steps. We also noticed a significant increase in the student's absence in class, possibly because, as they exactly knew what to do by the end of the semester, they thought they could do it without any supervision at their own pace. Assistance to class was not required but the main problem was that deviations from the project due to wrong design or implementation decisions were often detected too late. This sometimes led to students' demotivation, as they hat to rethink important items in their projects.

As a result from these concerns, we decided to shift to an agile learning methodology approach (from 2015-16 to 2021-22). The term agile learning, in contraposition to the cascade nature of project-

based learning, embodies the principles of agile programming included in the Manifesto for agile software development that was published in 2001 (Beck et al. 2001). This model, at first sight, could better fit the purpose of our course. In agile learning, the principles of agile software development are used in the context of learning. Agile software development uses an incremental model based on the continuous planning and team collaboration that follows the cycle of software development in an incremental way. The cycles of planning, design, building, testing and delivering a project are no longer large sequential stages that need to be completed before starting the new one but are incrementally updated to rapidly deliver working software to the customer.

In this work, we present the results of our experimental work on agile learning in a Computer Science Laboratory undergraduate course and compare them to our previous project-based approach. We particularly address the research question of whether the application of an agile learning methodology can decrease the truancy rates since a daily basis work is needed in order to deliver sprints on time.

In section 2 we first review the literature and in section 3 we compare both approaches and detail our implementation of the methodology. In subsequent sections we show and discuss the results obtained.

2. Previous work

Since the publication in the early years of the XXth century of The Project Method (Kilpatrick 1918) and its ulterior redefinitions as project-based learning (Guo et al. 2020), there have been many attempts to provide students with professional skills using a real problem-solve approach at different levels of education (Barron et al. 1988, Bell 2010, Kokotsaki et al. 2016, Morrison et al. 2020, Sharma et al. 2020, Tsybulsky et al. 2020).

Although Lang (2017) proposes the term agile learning to name the application of the processes and principles of agile development to learning, such a methodology has formerly been widely applied to divers education environments such as software engineering (Dewi et al. 2014, Scharf et all. 2013, Arcamone et al. 2013), interdisciplinary education (Gestwicki et al 2016), serious games (Lynch et al. 2011) or discrete mathematics (Duvall et all. 2017). These works show a growing interest in applying agile learning methodologies to allow students to work together in an energetic, targeted, and effective way in an educational context (Salza et al. 2019).

In his work, Lang states some of the advantages of agile learning. First, agile learning combines learning and application of learning as when new concepts are added, they are immediately applied in practice. And second, that due to the interactive nature of the methodology using sprints, students fail more and fail faster. He also states some disadvantages like that agile learning takes longer than project-based learning and that for students it is easier to fall behind.

Similar approaches and results can be found in the literature exploring the advantages of using an agile learning-based approach in computer science courses (Ju et al. 2020; Salleh et al. 2010; Roig-Maimó & Mas-Sansó 2022). In this work, we depict the importance of agile learning as a solution to the student class absence. As far as we know, there are no such results based on long term analysis (nine years) of class attendance when comparing both methodologies applied to programming reinforcing for non-computer scientist engineering students.

3. Implementation: project-based learning vs agile learning

The Computer Science Laboratory (CSL) course is a mandatory subject taught in the first semester of the last year of the Telematics Engineering Degree at our university. The number of students has important deviations depending on the academic year, with a mean of 15.5 students per year and a standard deviation of 4.2 along the CSL course evolution (see Fig. 1).

As previously mentioned, the main goal of the course is the strengthening of the programming skills attained throughout two previous programming courses. We seek the achievement of such learning outcome through the development of an Android mobile application. Android is chosen because the students are used to the JAVA language from the previous programming courses, so no additional time must be devoted to the learning of a new programming language.

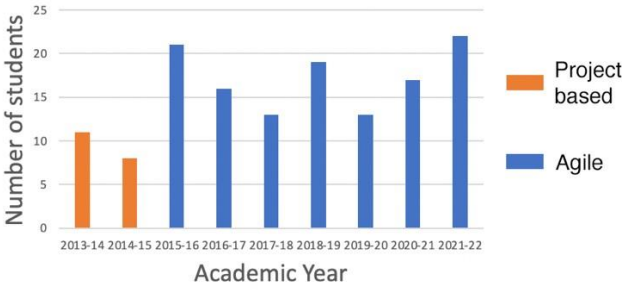

Figure 1: Students per academic year

The first two academic years (2013-14 and 2014-15) were we used a project-based methodology; students followed the traditional stages of software development in a cascade approach. At the beginning of the semester, the students were proposed a common programming project and they had to work out a solution. They first analyzed the requirements, then they fully designed the application, and then they started the project implementation and testing. Of course, they received the lecturers support all along the process.

The seven following academic years (from 2015-16 to 2021-22), were we used an agile learning methodology approach, we acted as the project customer, so we could change design criteria on the fly to force retrospective actions or to change initial requirements. When a retrospective analysis detected a requirements' change or that some modifications to the final product were needed, the students were asked to review the new application status. At the beginning of the semester, the students freely chose a team partner to work together all along the course.

4. The settings of the course following an agile learning methodology

In the first lecture of our agile learning methodology approach, we explained the main goals of the course and we introduced the agile learning methodology that we were going to use. We proposed a complete project that, as formerly stated, consisted in the development of a game application on an Android platform.

In that first lecture, we also provided the students with a proper view of the semester schedule. The semester duration of the course (15 weeks) was divided in a fixed number of sprints. A week had two class sessions of two hours each and six hours of autonomous work (which corresponds to 6 ECTS credits). Each sprint took one or two weeks, depending on its complexity, and resulted in the delivery of new functionalities to the final product. A sprint is one of the most characteristic trends in the Agile philosophy and it consists of a short term during which a certain task or set of tasks must be completed and the results obtained must be delivered. In the CSL course, a project was organized as a series of sprints that the students should go through.

The proposal of a sprint consisted of a written document specifying the title, the goal, the requirements and the delivery date of the result. The application requirements were presented as user stories to the students.

When a sprint finished, the students were asked to deliver their work and a new sprint was proposed. We reviewed the delivered work and gave feedback to the students before the new sprint began, so they had time to sort out detected problems.

Face-to-face class time was devoted to student guidance and support, and to the explaining of existing misunderstood concepts.

The frequent deliveries ensured fast and on time detection of programming errors or project concepts. Students' deviations were easily detected and, therefore, support efforts could be organized in order to correct such situations.

During the sprint's proposal, we often introduced specific points to be improved, forcing additional retrospective reflections.

By the end of the semester the students were asked to deliver the final product and a report documenting their work. The report consisted of a paper-like document where the students should describe what they did, the main troubles they faced and how they solved them. We also included a section where they could express their opinion about the course development and finally a self-assessment of their work.

5. Results and discussion

To analyze how the agile learning methodology has affected the teaching of our course along the years, we have collected qualitative and quantitative data:

- The qualitative data is extracted from the opinion section of the final reports of the students enrolled in the academic years where we applied the agile learning methodology approach (from 2015-16 to 2021-22). We have collected more than 350 qualitative comments from more than 70 different teams.
- The quantitative data is formed by the students' assistance and the students' assessments. It is obtained using the academic information from the academic students records over the complete longitudinal study (from 2013-14 to 2021-22).

5.1 Qualitative data: comments from the students reports

In this section, we present a summary of the results, extracting the ones that appeared the most. They are aligned with the results obtained in previous work:

- About working in pairs, a worth-mentioning aspect is that the students have felt very comfortable working in pairs, and they think it has allowed them to progress faster in the project. The importance of team working received a lot of comments.
- About working in sprints, the teams also point out that working with time spans of one or two weeks has been a very positive aspect, because they easily detected and corrected the designing and programming failures early and they could keep progressing without many problems.
- About consolidation of concepts, some students report that they have deeply understood basic concepts as object-oriented programming and top-down design. An easy adaptation to the agile learning methodology has also been widely reported.
- The feeling of satisfaction is also present in an important share of comments.

5.2 Quantitative data: students' assessments and assistance

Figure 3a depicts the graphical results of comparing what is the expected qualification of the student's pairs from their report (see orange color) in front of the actual assessment of their project (see blue color). We have used the data from all the academic courses, i.e., including the data of both approaches: project-based (2013-14 and 2014-15) and agile learning (from 2015-16 to 2021-22) bounded by a red vertical line in figure 3a.

As we can observe, the students' perceptions (self-assessment) are not too far from the assessment they get (actual assessment). However, we can see a noticeable overestimation in the first two years, which corresponds to the project-based approach methodology. That's an expected result because they don't get any feedback till the final assessment; while in the agile approach, they get feedback on their results after each sprint.

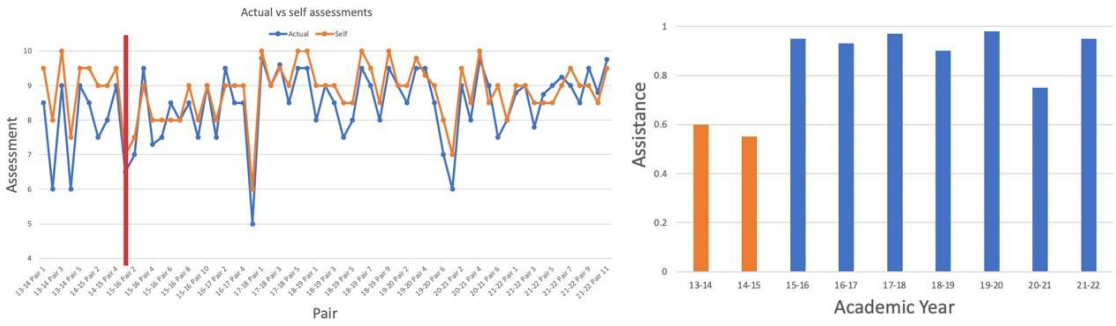

Figure 3: a) Perceived versus actual assessments and b) annual class assistance

Figure 3b shows the comparison of the assistance to class rate for both methodologies: the traditional project-based (see orange color) and the agile-based (see blue color) approaches. We should notice the data of the academic year 2020-21, which coincided with COVID pandemic, and consequently affected the assistance rate, which can be perceived as a decrease of around 20%. However, as we clearly observed, during the first two years, when a project-based learning approach was used, students' assistance to class was around 0.6, a 40% lower than the mean of 0.95 when using agile learning. The difference was statistically significant ($F_{1,7} = 33.35$, $p < .001$). The fact that short term deliveries are expected at the end of each sprint, motivates the teams' attendance as they can get immediate support and validation to their design and programming decisions. Moreover, we have observed that when somebody was absent from class, his teammate was attending the class. Rarely both team members were absent.

The data analyzed for Figure 3b encourage us to the application of agile learning approaches in order to motivate the students' assistance and, therefore, to decrease truancy rates.

6. Conclusion

In this work, we have presented the results of the application of an agile learning methodology in a Computer Science Laboratory undergraduate course with the goal of decreasing the truancy rates.

We have taught a Computer Science Laboratory course of a Telematics Engineering Degree over nine years. Although we had successfully used a traditional project-based methodology approach for two years, we noticed a significant increase in the students' absence in class. As an attempt to decrease that truancy rate, we adopted an agile learning methodology for the following seven years. Our primary hypothesis being that if a daily basis work is needed in order to deliver sprints on time, then class assistance should be a requirement to avoid falling behind.

At the end of every academic year, we collected qualitative and quantitative data.

From the qualitative data extracted from the final reports of the students we can stand out some positive aspects that are coherent with those previously pointed out in the literature. The students have felt very comfortable working in pairs; team working has been largely and positively commented. The time spans provided by the division of the project in sprints is another positive aspect as the students can detect and correct design and programming flaws in early stages of the project. They claim an easy adaptation to this learning model.

From the quantitative data formed by the students' assistance, it is remarkable that the truancy rate had significatively decreased, getting an assistance rate of 95%. This result led us to conclude that applying an agile learning approach is a good strategy to motivate the students' assistance and, therefore, to decrease truancy rates.

Acknowledgements

The authors acknowledge the Project EXPLainable Artificial INtelligence systems for health and well-beING (EXPLAINING) funded by PID2019-104829RA-I00 / MCIN/ AEI / 10.13039/501100011033. We also thank the University of the Balearic Islands, the Higher Polytechnic School (EPS) and the Department of Mathematics and Computer Science for their support.

References

Barron, B. J., Schwartz, D. L., Vye, N. J., Moore, A., Petrosino, A., Zech, L., & Bransford, J. D. (1998). Doing with understanding: Lessons from research on problem-and project-based learning. *Journal of the learning sciences*, 7(3-4), 271-311.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). Manifesto for agile software development.

Bell, S. (2010). Project-based learning for the 21st century: Skills for the future. *The clearing house*: *A Journal of Educational Strategies, Issues and Ideas*, *83*(2), 39-43

Dewi, D. A., & Muniandy, M. (2014, September). The agility of agile methodology for teaching and learning activities. In 2014 8th. *Malaysian Software Engineering Conference (MySEC*) (pp. 255-259). IEEE.

Duvall, S., Hutchings, D., & Kleckner, M. (2017). Changing perceptions of discrete mathematics through scrum-based course management practices. *Journal of Computing Sciences in Colleges*, 33(2), 182-189

Guo, P., Saab, N., Post, L. S., & Admiraal, W. (2020). A review of project-based learning in higher education: Student outcomes and measures. *International Journal of Educational Research, 102*, 101586.

Gestwicki, P., & McNely, B. (2016). Interdisciplinary projects in the academic studio. *ACM Transactions on Computing Education (TOCE)*, 16(2), 1-24.

Ju, A., Hemani, A., Dimitriadis, Y., & Fox, A. (2020, February). What Agile Processes Should We Use in Software Engineering Course Projects? In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 643-649).

Kilpatrick, W. (1918). The project method. *Teachers college record*, *19*(4), 319-335.

Kokotsaki, D., Menzies, V., & Wiggins, A. (2016). Project-based learning: A review of the literature. *Improving schools*, 19(3), 267-277.

Lang, G. (2017). Agile learning: Sprinting through the semester. *Information Systems Education Journal*, 15(3), 14.

Matković, P., & Tumbas, P. (2010). A comparative overview of the evolution of software development models. *International Journal of Industrial Engineering and Management*, *1*(4), 163.

Morrison, J., Frost, J., Gotch, C., McDuffie, A. R., Austin, B., & French, B. (2021). Teachers' role in students' learning at a project-based STEM high school: Implications for teacher education. *International Journal of Science and Mathematics Education*, *19*(6), 1103-1123.

Roig-Maimó, M.F., & Mas-Sansó, R. (2022). Applying Agile Learning on Computer Science Laboratory: A Case Study. In T. Bastiaens (Ed.), *Proceedings of EdMedia + Innovate Learning* (pp. 455-464). New York City, NY, United States: Association for the Advancement of Computing in Education (AACE).

Salleh, N., Mendes, E., & Grundy, J. (2010). Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering*, *37*(4), 509-525.

Salza, P., Musmarra, P., & Ferrucci, F. (2019). Agile methodologies in education: A review. *Agile and lean concepts for teaching and learning*, 25-45.

Sharma, A., Dutt, H., Sai, C. N. V., & Naik, S. M. (2020). Impact of project-based learning methodology in engineering. *Procedia Computer Science*, *172*, 922-926.

Scharf, A., & Koch, A. (2013, May). Scrum in a software engineering course: An in-depth praxis report. In 2013 *26th International Conference on Software Engineering Education and Training (CSEE&T)* (pp. 159-168). IEEE.

Thomas, JW. (2000). A review of research on project-based learning.

Tsybulsky, D. and Muchnik-Rozanov, Y. (2019). The development of student- teachers' professional identity while team-teaching science classes using a project-based learning approach: A multi-level analysis. *Teaching and Teacher Education*. 79, 48–59.

Werner, L., Arcamone, D., & Ross, B. (2012). Using Scrum in a quarter-length undergraduate software engineering course. *Journal of Computing Sciences in Colleges*, 27(4), 140-150.