Formative evaluation and learning analytics for agile teaching

JJ Merelo-Guervós
jmerelo@ugr.es
Department of Computer Engineering, Automatics and Robotics and CITIC, University of Granada
Granada, Spain

ABSTRACT

The agile manifesto provides a framework for software development in which customer comes first. Here we will describe a teaching experience based on agile principles that puts students at the center of a strategy that, through the use of fit learning analytics, is able to optimize the potential of a class and its individual students. We used project-based learning and formative evaluation as agile teaching best practices, evaluating student progress based on learning objectives submitted and evaluated asynchronously. The time when every objective is submitted and the time taken to pass it are the essential data points that will be leveraged to evaluate class progress, and the impact of specific measures taken to improve it, in-class or from one course to the next. Measures taken through three years with the same methodology prove that agile teaching can work, but it needs measurements for diagnosis and subsequent interventions to reach its full potential.

CCS CONCEPTS

• **Software and its engineering → Agile software development**.

KEYWORDS

Software engineering, agile methodologies

ACM Reference Format:

JJ Merelo-Guervós. 2024. Formative evaluation and learning analytics for agile teaching. In *Proceedings of ITICSE'24.* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Teaching software engineering represents not only one, but a series of challenges to the teacher, who needs the student to acquire a series of marketable skills at the same time that this acquisition is assessed in order to obtain a grade; essentially, it needs to deliver a product (student with acquired competences). Software development also focuses on delivering products, and it currently employs an *agile mindset*, following the Agile Manifesto [2], which is a customer-centered philosophy that maintains that any change in code should add value to the customer, either directly or indirectly. Teaching mindsets of philosophies is a notch more complicated than teaching skills; evaluating mindsets is even more complex. But that is what an industry with its ever growing *need for talent* is asking for.

One of the ways of teaching a mindset is having that same mindset while teaching; that is, if you want to teach agile, be agile while teaching [3, 6]. One step into this mindset is simply to consider class experience a product for which students, collectively and individually, are the customers, and use industry-standard tools such as GitHub to produce class materials, create course-oriented milestones, as well as generate issues with the problems that need to be solved, content-wise or related to any software that you might have developed to support teaching.

You can, however, go a bit further and embrace agile in all stages of the teaching/learning process. Several researchers have proposed manifestos in agile teaching, so the first step would be to understand those manifestos and adopt them where possible. In our case, the manifesto created by Krehbiel et al. [6] seems the closest to our philosophy; this manifesto emphasizes adaptability to personal (and other) circumstances, teamwork, focusing on learning objectives and not on grades, autonomous learning by the student, practical working instead of theoretical teaching, and, over all, continuous improvement.

Manifestos are never enough either in development or teaching, and agile development encourages the adoption of best practices in every phase of development; but the last part of the manifesto which talks about continuous improvement, as well as the fact that we are talking about *best* practices, would need some way of measuring progress and how adopted practices affect the classroom. This is why we would like to add a new item to the manifesto: *learning analytics* [4] over anecdotal evidence or end-of-class failure/success rates.

Within this agile mindset, there are teaching methodologies that fit better; adopting these best practices is the first step towards agile teaching. We will focus on two of them that fit well with each other: project-based learning gives the student autonomy (which is one of the items in the manifesto) through the development of a project across one or several subjects [5]; formative evaluation [10] also fits the autonomous learning part of the manifesto, but also focuses on learning objectives, since it pursues helping the student through continuous feedback on their submitted work (that advances the project mentioned above). This feedback helps continuous improvement of the project they are working with, which is other of the items in the manifesto. Finally, *flipped learning* [7], that turns classrooms into working, and not student-listening-to-lectures spaces, is another best practices that blends well with the two mentioned above.

In this paper we will describe the experience carried out in the last three school years in a 4th year (7th semester) subject in the Computer Science degree at the University of A City in A Country the subject deals with A Subject Matter so the project needs to be

ITICSE'24, June 03–05, 2018, Milan, IT

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

working towards the deployment of an application of that kind. But alongside the description of the experience, we will try to answer the following research questions:

- **RQ1:** Can you leverage metrics to improve individual and collective learning?
- **RQ2:** Are there early indicators or class success or failure?
- **RQ3:** Can we measure the effect of flipped learning through the gathered software analytics?

The rest of the paper is organized as follows: next we will present the state of the art in learning analytics applied to evaluative formation and flipped learning. The class setup, as well as how data is collected will be presented in Section 3. The results of the analysis will be presented in Section 4 along with the answers to the research questions. Finally we will draw some conclusions in Section 5.

2 STATE OF THE ART

Learning analytics (LA) [4] have been applied to all kind of learning processes and subjects; this includes problem-based learning combined with formative assessment (FA); [1] mentions that learning analytics and formative assessment work well together since they both provide immediate feedback on the quality of the learning process; however, it can be argued that LA is an essential part of FA, since a deep understanding of the objectives achieved by students is needed in order to provide student-specific feedback. In general, however, LA will refer to additional, macro-measurements at a class level or analytics that compare student performance with other students or with previous years [11]. This last paper reveals an increasing interest in FA combined with LA, with computer science courses leading the way, but with other disciplines and subjects also using it to improve success rates. However, there are no general rules on what specific analytics should be used. In general, it is important to consider a learning (and possibly learner) model in order to be able to correctly measure learning outcomes and how they are reached. For instance, [8] focuses on *conceptions*, how the students understand (or not) certain specific and common ideas, thus their process is geared towards using common assessment tools to measure understanding of those conceptions. Student attitude and disposition towards learning can be as interesting as a model but impossible to measure without explicitly asking the student; that is why authors such as [9] propose a framework that combines traces with surveys in order to predict outcomes such as early drop out.

What seems to be missing is, however, a single set of learning process measurements that can be used across any platform, subject, or way of deployment of formative assessment strategies, and which can be used for individual assessment, as well as aggregated for class and course assessment. This is what we will try to propose and evaluate next, after exposing how our class has been set up in the next Section.

3 CLASS SETUP AND METHODOLOGY

Problem-based learning make students work on their own project and repository; this course will have 10 levels of project completion ("objectives"), numbered from 0 to 9. For every objective they will create a branch and a pull request from that branch to their main one; a pull request is a "request to merge" a set of changes; acceptance of that PR will imply merging into the main branch (and, in real world, usually deployment to production). This class setup matches the agile philosophy which prefers interaction and working software over "comprehensive documentation" (which, in a class environment, would mainly be "theoretical" classes and memory-based exams). At the same time, PRs are the main interaction medium in software development: it allows members of the team to perform quality (and other, such as compliance) checks on the code that is going to be used; this matches the requirements of formative evaluation too. The professor (and other students who have also overcome that specific objective) will review that PR, request changes if it falls short of reaching the objectives, and eventually accept it when it does.

Since these PRs happen in the students' repos, we need a single, centralized place as a *platform* to have an unified view of the whole class; this is carried out via a single, "class" repository, where the students submit a PR to a file that is different for each objective, triggering a workflow with a series of baseline tests of correctness of the PR submitted. When the student overcomes the objective, a mark is inserted into the same file by the professor. Automatic workflows then analyze these files to generate a data file that records these events and when they occur. This file is the single source of truth for the state of every objective for every student, and will be analyzed to study the progress of the class.

The student needs to pass objective number 5 to get a passing grade; every objective beyond that one works toward obtaining full marks. Objectives receive a different percentage of the total grade depending on the actual amount of work they need. The percentage was adjusted depending on the amount of time students needed from submission to passing the objective in the first year. It did not need further adjustments in the following years. *Bigger* objectives get 15% of the grade, while the smallest ones (for instance, the first one) can get only 5 %. All students passing a certain objective obtain the same grade; this is why formative evaluation is often called *grade-less*, since learning objectives are reached or not, and when reached, everyone gets the same grade. Different grades correspond only to the fact that different objectives are reached.

Using GitHub as a platform and pull requests as the main vehicle helps the creation of other workflows and work patterns around these PRs. Every time a PR submitted to the "central" repository passes tests, five random reviewers are drawn from the pool of other students that have already cleared that objective. The student, then, receives feedback for every learning objective not only from the teacher, but also from their peers. Again, this fits the agile mindset, but when students need to verbalize the issues with a PR by a colleague, it helps them settle the knowledge they have acquired in the objective they already passed. Additionally, it helps them understand the need for a efficient communication, through PRs, with other members of the team, and to use effectively these tools, closing the gap between classes and actual work and following an agile mindset[1].

In general, we strive to make feedback as fast as possible, so that students can react to it in a timely way. The general rule is that we will react to requests for feedback (which are made either by

[1]These reviews are made for additional credit; reaching all the objectives includes only 70% of the final grade

a comment with mention in a pull request or using the specific mechanism GitHub has to request reviews) in less than 48h[2]. A fast feedback is essential for the student to still have whatever conceptions or assumptions fresh so that they can be modified.

This implies a certain (and rather irregular) workload, mainly in the early stages when students submit their pull requests at pretty much the same time. However, early objectives require less effort to review than later ones; and when these ones arrive, the teacher only has to chime in when one to three other students have already approved the submission. Although it is rather infrequent than no further suggestions have to been made, most frequent errors have already been ironed out by these reviews and request for changes. Additionally, every course a *frequent errors* specific for every objective is created. This is used by the students as a check list before submitting the review, but also, since it is created as submissions are reviewed, also by the teacher for quoting when these frequent errors show up in a pull request. In some cases specific errors that can be automatically checked show up (for instance, using capital letters in the names of files, something that is generally discouraged in repository-checked projects) they are incorporated into the automatic tests the PRs undergo when submitted to the central repository, so that they do not require additional rounds of reviewing by the teacher, releasing time needed for in-depth examination and feedback to the student; this is well within the agile mindset.

In general, workload for the teacher is bigger than for *summative* evaluation or traditional *flipped learning* classes. However, the system has been set up so that most of the time spent by the teacher is effectively spent in giving constructive feedback to the student, so that they can reach their learning objectives.

During face to face classes, time is spent either addressing issues with specific students, or making small-group explanations on the next submission to those that have just passed an objective. These whiteboard (never slide-based) expositions are adapted to specific circumstances of the student of group of students (early submitters, or late ones, for instance). In general, classes are working spaces where students and teacher interact with each other and work towards reaching their learning objectives.

In our country universitary system and for this subject, classes and evaluations are organized in this way:

- A class period, which takes 15 weeks, for a total of (up to) 60 face-to-face hours. This is equivalent to 6 ECTS credits.
- The *ordinary evaluation period*, which takes place up to 4 weeks later than the end of classes.
- The *extraordinary evaluation period*, which takes place up to 3 weeks after the ordinary evaluation period.

Whoever passes the minimum number of objectives during the ordinary period is evaluated during that period; if that is not the case, students can continue submitting the rest of their objectives up to the end of the extraordinary evaluation period. Data has been collected in this way for three consecutive school years: 2021-22, 2022-23 and 2023-24[3]. After the course is over, student GitHub nicks are anonymized with a one-way hash so that privacy is preserved[4]. These are publicly available at https://github.com/JJ/ivR.

Table 1: General data about the three courses

Course	2021-22	2022-23	2023-24[5]
Students	51	48	40
% no-shows	37.25%	66.66%	27%
% passed ordinary	39.21%	20.83%	52%
% passed extraordinary	23.53%	12.5%	N/A
% total passed	62.74%	33.33%	N/A

Next we will choose metrics and analyze the kind of insights we can obtain from them, and how they can be used to improve learning outcomes.

4 RESULTS

Some basic numbers of the three courses over which we have been using this methodology is shown in Table 1, which shows that a methodology, by itself, is not enough to guarantee good results. The first year it was used the overall result was relatively good (more than 50% passed), but almost 1/4th of the class needed additional time to pass. The next year, overall, was a disaster, with only 1/3rd of the class passing, and of those who did, more than 1/3rd needed to do so on additional time[6]. The causes of this are still being analyzed (and are beyond the scope of this paper), but what is clear is that specific changes need to be made from one year to the next so that whatever students fall in class every year are able to succeed. Please note that no-shows + passed add up to 100%, since technically those that do not pass are not submitting their objectives, they are "not showing up" to be evaluated.

These end-of-course metrics will need to be supported by real-time analytics for the teacher to be able to intervene on class while there is still time to do it. The metrics used in the learning analytics system should, ideally, be obtained directly from the student activity, without needing additional surveys. They should also be directly related to learning outcomes, and, if possible, learning attitude, possibly indirectly in this case. This is why in this case we are gathering two data points:

- The time in which a student submits their PR for an objective.
- The time in which the teacher marks the objective as *passed* for the student.

In the first case, it is computed directly from the date the PR to the centralized repository is merged, that is, when the objective submitted clears the initial tests; a GitHub workflow computes it automatically and submits it to a data file that is also registered in the repository. It follows our requirements: it is obtained directly from the student activity, directly related to learning outcomes, namely, when the students feel they are ready at least to understand initial feedback from the teacher. Their attitude is reflected mainly in the time between clearing the previous objective and that submission,

[2]Exception being some weekends, and official holidays
[3]Incomplete at the time of this writing
[4]Even if students are warned to use whatever GitHub nick they want, including an anonymous, class-only one. The only thing needed for the teacher is to be able to clearly identify which student corresponds to every nick
[6]Provisional results for this year are, however, quite promising, with the number of persons who have passed already surpassing 21-22

although that is clearly influenced by the difficulty of the objective itself.

In the second case, there might be a small delay between the moment the objective is accepted and the mark is done in the file and submitted to the repository, but this delay is usually a few minutes, and except for errors, less than one hour. The scale of this metric, or the difference between both times, is counted by days, so the small bias this might introduce is not really important. How many days the student needs, and its relationship with the time needed by other students in their class or previous years, is really representative of attitude.

In order to be able to normalize and compare different courses, dates are counted from the day the course starts. Raw data is shown in Figure 1.

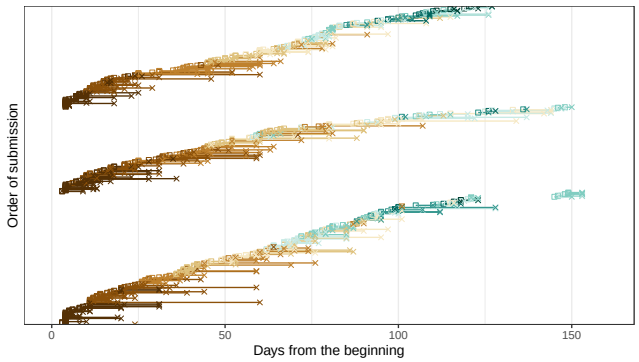

Figure 1: Segment plot showing the number of days from the beginning of the course when objectives (indicated by color) have been submitted, with the segment end indicating when they have been approved. From bottom to top, this has been done for course 2021-22, 22-23 and 23-24 (still ongoing).

This visualization already gives you an idea of the time needed to pass every objective: there is an abundance of long segments in the brown color, which is used for objectives number 2 and 4, the most complicated. The slope that the left hand side ends of the segments form also shows the rhythm of submission, how it slumps during certain periods and then becomes steeper. We need, however, additional processing to gather actionable insights from this data.

For instance, some specific measure of individual performance. The one that pops out most immediately is simply the difference between the time of submission and the time of approval, a measure of the effort the students put into learning the specific concepts and skills required for every objective.

Figure 2 shows the histogram of this difference for every objective in every course; differences between the different courses go beyond the different number of students; the first two courses had approximately the same number of students, while this year had less However, 21-22 and 23-24 show the same distribution, with the mode being student overcoming the objective in 1 week or less, while 22-23 shows that most students took between 1 and 2 weeks to pass objectives. This already shows the descriptive power of this metric: we already know that the results during 22-23 were not

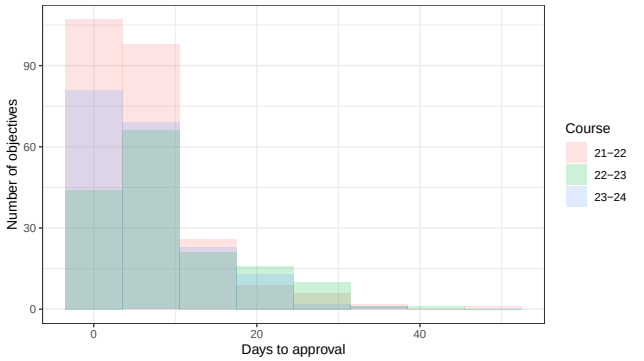

Figure 2: Histogram of the difference between the time of submission and the time of approval for every objective. Bins are 7 days wide

good, and this is correlated with the fact that they needed more time to pass objectives.

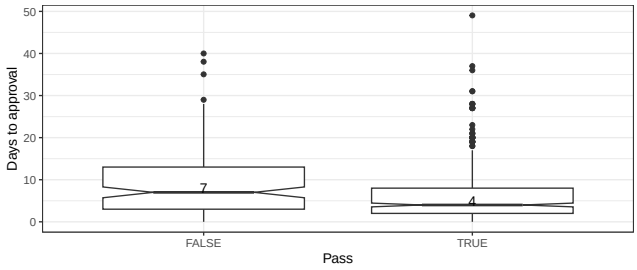

Figure 3: Boxplot of the difference between the time of submission and the time of approval for every objective, grouped by whether the student passed or not.

This metric, however, does not distinguish between students who passed and those who fell short of objective number 5, the minimum requirement. Will time-to-pass be a good predictor of success? This is represented in Figure 3, that shows a boxplot of the number of days to pass an objective (from submission) for those who failed the course (FALSE) and those who passed (TRUE). Median differs, as shown, in 3 days, and the difference is statistically significant (p-value < 0.001), so this metric is a good candidate for being an early indicator of success. However, this is an average over all objectives and over all the class, with objectives with more submissions weighing more. It is also an average over all courses, with path-dependent differences between courses (because it will depend on the student that submits first, and how they help the rest of the student with reviews). We show the boxplot for the different courses in Figure 4, taking into account only students who have passed. In this case, the difference between courses 21-22 and 23-24 is not statistically significant, but the difference between 22-23 and the other two is (p-value < 0.001).

Even when we are taking into account those that passed, on average a *suboptimal* course will take *more* time to succeed in passing an objective. Obviously, more time for every objective will

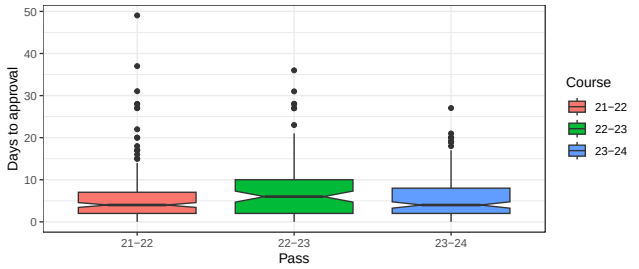

Figure 4: Boxplot of time needed to pass objectives by course

mean that there will be less time for the rest of the remaining objectives. While a better-than-average course will have a high success rate, students will be able to reach further in terms of objectives than others that do not.

This metric refers, in general, to autonomous learning; formative evaluation favors that mode of achieving learning objectives, but in order to have a clearer picture of how learning takes place, we need to factor in interaction with the teacher and other students *in class*. Problem-based learning favors flipped learning, eschewing lectures with unidirectional communication from teacher to students; this is why we need to find out how this time spent in class actually translates into learning. First we will compare submission during the class period (first 15 weeks) to the rest.

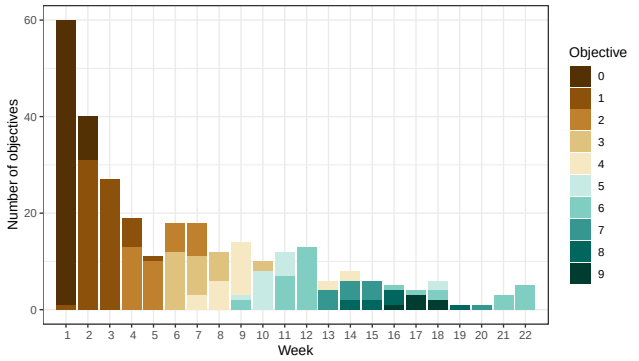

Figure 5: Barplot of submission per week, with colors indicating objective number.

Figure 5 shows a barplot of the number of objectives, with color indicating the actual number of objective. We can see what is the usual period of submission for every objective (objective 0 finishing in the second week, for instance, objective 1 in the 5th week, objective 2 in the 7th and so on). Submissions take a dive after the 15th week, but this is due to many factors, including the fact that it is a vacation period and that at least some of the students stop submitting after reaching the minimum number of objectives required for passing. We also see a dip in the 5th week, where there is a national holiday that, in the past years, has implied essentially a week without classes. This dip repeats itself in the 8th week (another national holiday) and the 13th week (essentially a week-long national holiday). Submissions hardly recover after that week; in many cases this is due to the students reaching the minimum requirement in week 11 and dropping out. But even if the student wants to continue, it is complicated for them due to the workload in other classes, which usually concentrate on the last week of face to face classes. With this data, we can already see that attending class contributes to the submission of objectives: less is achieved in weeks without classes. But we need to delve a bit deeper into this data, since it might be other factors, like help from other students, or simply the fact that during weeks with classes the student allocates more time to work (in this and other courses).

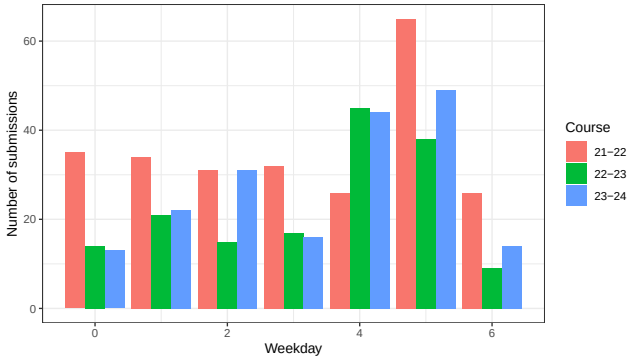

Figure 6: Barplot of submissions by day of the week.

Figure 6 shows submissions by day of the week, starting by Sunday (day 0) to Saturday (day 6). Classes take place on days 4 and 5, Thursday and Friday. Thursdays take place in a big room with all students attending (it is called the "Theory" class regularly); Fridays are split equally in two groups, and take place in a different class with a different configuration. We need to analyze the situation for every course, since behavior is different. During the first course, 21-22, Thursday was devoted mainly to lectures, instead of individual work. There are *less* submissions on that day than in any other day of the week, even less than Saturdays; this probably implies that lectures are *substracting* value to students, instead of adding value to them; Fridays, anyway, registered the top number of submissions, so that class was effectively adding value to students. Next course, 22-23, changed how these classes took place with lectures almost eliminated after a few weeks: submissions on Thursdays and Fridays are much higher than the rest of the days. Surprisingly enough, Fridays are lower than on Thursdays. Paradoxically, this was due to the fact that students *expected* lectures on Thursdays, and then attended class that day. Class attendance was dismal on Fridays, but even so, more submissions were made on that day. Situation this course, 23-24, is close to ideal: submission rise on the days leading up to the class (with a dip on Wednesday, which could be explained by the abundance of holidays on that day this year) and they reach its optimum on Thursdays and Fridays, with a relatively small difference. In general, it can be said that the presence of the teacher and other students adds value to them, and this value is inversely proportional to the time devoted to unidirectional lectures.

Figure 7 shows the number of objectives approved by day of the week. The situation is similar to the previous one, except for the anomalous number of approvals on Sunday the first year. However,

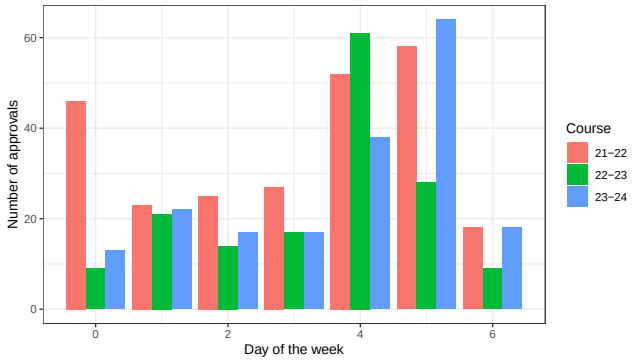

Figure 7: Barplot of number of approvals by day of the week.

while the first and last year the number of approvals was higher on Friday than on Thursday, the second year there were half as many approvals on Friday than on Thursday. Taking into account the average number of days to approval, this in general implied that students waited for Thursday to understand feedback given by teacher (and other students), and devoted class to it, managing a pass possibly by the end of the class or even when returning home. Very few left it for Friday, since attendance on Fridays was very low. In general, however, this shows that face to face personal explanations in class are a good complement to written reviews in the pull request, and it really adds value to students, helping them overcome the objective. This frequent interaction is also one of the principles in the Agile Manifesto. Together with focusing on activities that add value to the student, like personal or very small group introductions to every objective and face to face feedback on obstacles to achieving the objective, it shows how an agile mindset can be successfully applied in the classroom.

5 DISCUSSION AND CONCLUSIONS

In this paper we have described our experience in the use of an agile mindset in computer science education, including using performance indicators from the activity to assess (and eventually improve) course-wide, as well as individual performance. With the use of learning analytics, in this paper we propose two simple data points that are able to capture most of the activity, and to a certain point the attitude, of the student, as long as the teacher devotes enough time to providing feedback and eventually approving the learning objectives provided by the student.

The data points that we have proposed are two: submission and approval time for every objective, and the main metric is the difference between them, which reflects the time needed by the student to assimilate, and thus pass, objectives. By using collective analytics and proposing specific actions, we were able to improve course performance from last year (22-23) to this one (23-24). This metric also captured the main differences between courses, mainly time needed to pass every objective from the time of submission. Although it is soon to tell, a good methodology leads not only to better results, but also results that are indistinguishable statistically between courses, as shown in Figure 4. This constitutes a positive response to RQ1: metrics, if collected and chosen with precision, can capture the main differences between courses, and also be used to improve course performance: measuring time-to-pass every objective individually can be used to target individual students for special attention.

Research question 2 can be answered by looking at the same metric in early objectives. Even if submissions are made early, or soon after the previous objective is cleared, managing to get an objective approved in more days than the average, on or beyond the 75% percentile, will raise a flag and indicate a student that will need special attention. The same happens for the whole class; if the median starts to shift beyond 6 days, some intervention will be needed to get a significant amount of students passing the subject. Of course, too many students dropping out of class and stopping making submissions will also be a cause for failure, but the leverage the professor might have in those case is extremely limited. Too much time to pass an objective, however, would indicate that the student has the attitude, if not the aptitude to pass it, so an intervention should have a certain impact.

The answer to RQ3 can be seen mainly in Figures 5, 6 and 7, that show that weekly classes have a clear impact on submissions as well as passes; weeks without classes show overall lower numbers of submissions. This proves that agile teaching, with frequent interactions with the *client* (= student), and using data to assess and address specific needs, can be used to improve individual student, as well as class performance, adding value to the student.

This learning analytics system, which was deployed from the first year (with small variations), allowed us to make different interventions, not all of them successful. To avoid the high dropout rate in the first objectives, we started a hackathon in the first week of class of 22-23. Unfortunately, this intervention did not succeed, and that was clearly reflected in the analytics. Starting a hackathon to pass objectives 2 and 3 (and dropping lectures) did have a limited success, and helped some students to get back on track. In the first days of this course 23-24 we introduced another gamification experience after dropping the hackathon; this achieved its short-term objectives, as the analytics show, and apparently some long-term objectives too.

In general we can conclude that an agile mindset of serving the customer (the student) and adapting to specific class circumstances, as revealed by the learning analytics and its comparison, will boost student success and efficient acquisition of engineering best practices and skills. Besides, the learning analytics system is totally independent of the platform that is used for assessing student's submissions. As long as the submission time and when it is passed is known with precision, the same measurements can be applied; these measures can be downloaded from a LMS or simply noted in a spreadsheet.

Since every class is going to be different, in-class measurements such as the time the student needs from submission to objective success as a measurement of *product quality* is an actionable diagnostic that can help teachers apply individual-level or collective measures to improve learning outcomes in an agile way.

REFERENCES

[1] Naif Radi Aljohani and Hugh C Davis. 2013. Learning analytics and formative assessment to provide immediate detailed feedback using a student centered mobile dashboard. In *2013 Seventh international conference on next generation*

mobile apps, services and technologies. IEEE, 262–267.

[2] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. 2001. The agile manifesto.

[3] Andy Hon Wai Chun. 2004. The agile teaching/learning methodology and its e-learning platform. In *Advances in Web-Based Learning–ICWL 2004: Third International Conference, Beijing, China, August 8-11, 2004. Proceedings 3*. Springer, 11–18.

[4] Doug Clow. 2013. An overview of learning analytics. *Teaching in Higher Education* 18, 6 (2013), 683–695.

[5] Joseph S Krajcik and Phyllis C Blumenfeld. 2006. *Project-based learning*. na.

[6] Timothy C Krehbiel, Peter A Salzarulo, Michelle L Cosmah, John Forren, Gerald Gannod, Douglas Havelka, Andrea R Hulshult, and Jeffrey Merhout. 2017. Agile Manifesto for Teaching and Learning. *Journal of Effective Teaching* 17, 2 (2017), 90–111.

[7] Fernando M Otero-Saborido, Antonio J Sánchez-Oliver, Moisés Grimaldi-Puyana, and José Álvarez-García. 2018. Flipped learning and formative evaluation in higher education. *Education+ Training* 60, 5 (2018), 421–430.

[8] Judith Stanja, Wolfgang Gritz, Johannes Krugel, Anett Hoppe, and Sarah Dannemann. 2023. Formative assessment strategies for students' conceptions—The potential of learning analytics. *British Journal of Educational Technology* 54, 1 (2023), 58–75.

[9] Dirk Tempelaar, Quan Nguyen, and Bart Rienties. 2020. *Learning feedback based on dispositional learning analytics*. Springer, 69–89.

[10] Martin Tessmer. 1994. Formative evaluation alternatives. *Performance Improvement Quarterly* 7, 1 (1994), 3–18.

[11] Ke ZHANG, Ramazan YILMAZ, Ahmet Berk USTUN, and Fatma Gizem KARAOĞLAN YILMAZ. 2023. Learning analytics in formative assessment: A systematic literature review. *Journal of Measurement and Evaluation in Education and Psychology* 14, Özel Sayı (2023), 359–381.