Evaluation of Students Through the Commits Made on the GitHub Code Repositories

Alexandru ALEXANDRU[1], Alin ZAMFIROIU[2,3]
[1]Military Technical Academy "Ferdinand I"
[2]Bucharest University of Economic Studies
[3]National Institute for Research & Development in Informatics - ICI Bucharest
alexandru.alexandru.aa99@gmail.com, alin.zamfiroiu@csie.ase.ro

Evaluation of students in the time of the semester is made in general by homework and some tasks that the students should resolve them. For a technical course like programming or development is important to evaluate the perspective of the students that can work in coding. In this paper, we present an instrument that can help teachers in student assessment by providing support for work management and automated code evaluation. For programming subjects, it is very important for the teacher to follow the progress of all students during the semester. This would mean giving them a lot of homework and correcting them. This is very time-consuming. The tool proposed by us will make an automatic evaluation of all the commits made by the students on a GitHub repository, the teacher being able to see the evolution of all the students much faster.
Keywords**:** *Students, Evaluation, Code, Repositories*
DOI: 10.24818/issn14531305/26.4.2022.02

1 Introduction

The teacher should centralize all student responses, correct them, and then give the pupils the results using traditional evaluation techniques. If a teacher wants to evaluate each student's development over the course of a semester or academic year, all of the students' tests should be collected in one place, and a sheet with each student's results should be made. It takes a lot of time and is very challenging, especially if the teacher has a large class size. It is manageable for small student groups, but for large student groups, the teacher won't be able to administer as many tests and quizzes.

Based on the student's responses, the system may easily and automatically perform these analyses during the virtual examination. The teacher would rapidly be aware of each student's progress in this way [1], [2].

If the results of the prior exam are used as the basis for the following test, it is crucial. To do this, the teacher should assess each student's development and design a unique test for them.

It will be extremely simple to create that in a virtual evaluation using the question from the pool. The method for selecting the questions for each student based on the prior test, in order to enhance the student's learning process, is still a challenge.

In this paper, we propose a way of analyzing the progress of students from a technical field by analyzing the code committed to a repository in GitHub [3].

All students from a technical field university have to write code and teachers should evaluate this code for each phase of the project or for each homework. Also, more and more teachers start to put students to use different repositories where they should upload the code to make it available for teachers and also for team workers if the project is for a team of students. In this way, it will be more convenient for teachers if will exist a tool that can analyze these repositories from students automatically and suggest a grade for them. In this way, the teachers should only check the correctness of the obtained grade.

In [4] it is presented the way of evaluation of students by online quizzes, and in this way, the number of meetings is reduced, by using the way of evaluation through the code written and committed to a public repository this

number of the meetings is reduced to zero. So, the teachers shouldn't meet with students. Each student should send to the teachers the link to his repository where commits all changes and the teachers can evaluate the progress through the platform that analyzes the code from that repository.

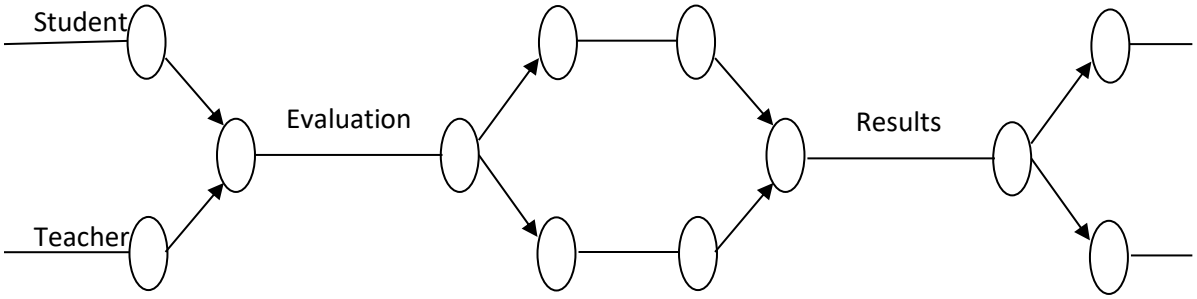

Fig. 1. Evaluation in a traditional way

Figure 1 shows the structure of meetings between professors and students in the traditional way of education. The teacher must meet with the student for the evaluation part and then for the part of providing the results.

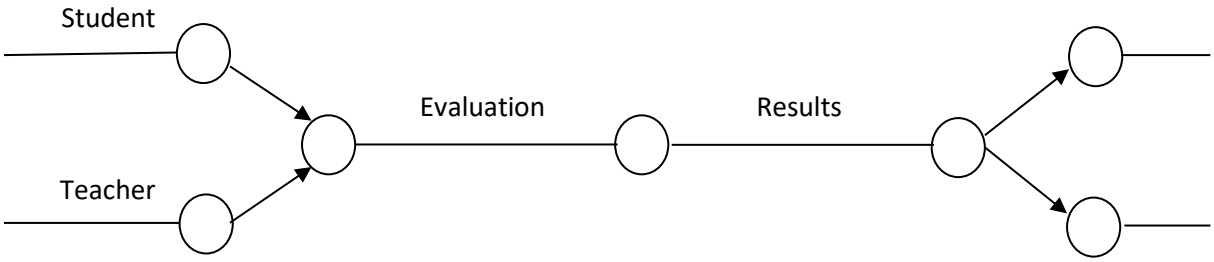

Fig. 2. Evaluation in an online way

Figure 2 shows the online teaching mode, in which the evaluation of grid tests is done automatically and thus the teacher meets with the student only once for the evaluation because the results are automatically displayed on the spot.

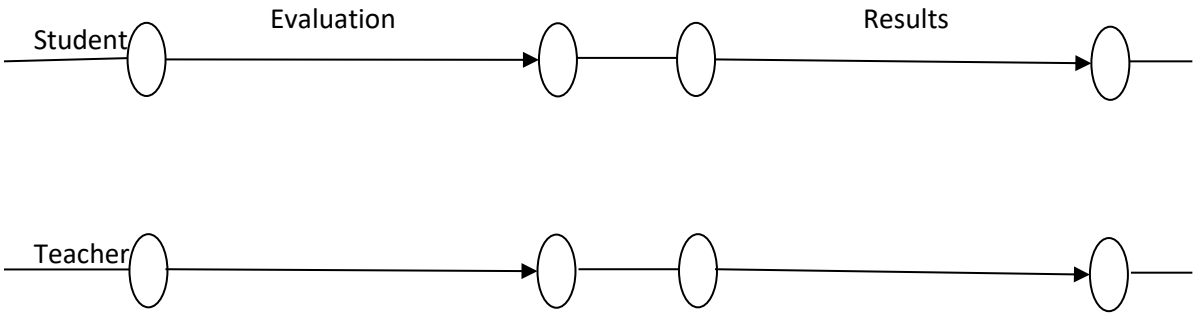

Fig. 3. Evaluation by analyzing the progress

Figure 3 shows the evaluation mode through the code repositories. In this way, progress is made automatically based on the code written by the student and loaded in that repo, without the need for a physical meeting with the teacher.

In this way, it can be observed that by analyzing the code on GitHub, the evaluation method of students is improved by reducing

the number of meetings between them and the teacher, the evaluation being carried out automatically.
According to [5], there are three main aspects regarding to the learner motivation: confidence, confusion and effort. These may be detected to some extent, by learner activity on different platforms.

2 Material and Methods

In this section, we describe the elements that we are using in the process of evaluation of stunts in an atomically way.
We create a prototype platform that will analyze the code committed in one repository of code. The platform will analyze the number of commits, the number of changes, and so on. This platform is presented in the next section. We have modified the platform in a way that it is able to connect to more public repositories from GitHub and make this evaluation for more repositories. In this way, we can evaluate more repositories from more students.
We have a group of 100 students from Bucharest University of Economic Studies, Faculty of Cybernetics, Statistics and Economic Informatics. They have been evaluated for a course by the code they write in the semester period. They have created on a repository that commits all the code written in the laboratories and for homework. The platform will get all these repositories links and will be made automatically the evaluation of all students.
For student evaluation, we created a system that provides a grade based on the extracted information. We generated a dataset, each characteristic having values in a predefined range. The characteristics we tracked are the number of total commits, the number of commits per month, the total number of changes made, the number of changes per commit, and the number of changes per month. To calculate the grade, we created a set of formulas (Equation 1), which can be configured according to the teacher's requirements (variables like x, y, z, q etc.).

$$\text{grade\_1} = \frac{10 \times number\_of\_commits}{x}$$

$$\text{grade\_2} = \frac{number\_of\_commits\_per\_month \times y}{z \times number\_of\_commits}$$

$$\text{grade\_3} = \frac{number\_of\_changes}{q \times number\_of\_commits}$$

$$\text{grade\_4} = \frac{number\_of\_changes\_per\_commit \times l}{m}$$

$$\text{grade\_5} = \frac{number\_of\_changes\_per\_month \times n}{p \times număr\_modificări}$$

$$\text{final\_grade} = \frac{\sum_{k=1}^{5} grade\_k}{5}$$

Equation 1. Formulas to calculate grade for a student

3 Code Evolution Visualization Platform

The created platform follows the progress of a student using 2 methods: the first one provides an overview of the changes made at each stage of the project by viewing them in an interactive way and the second one extracts certain features that provide information about the student's contribution, for example, their weekly changes [6].

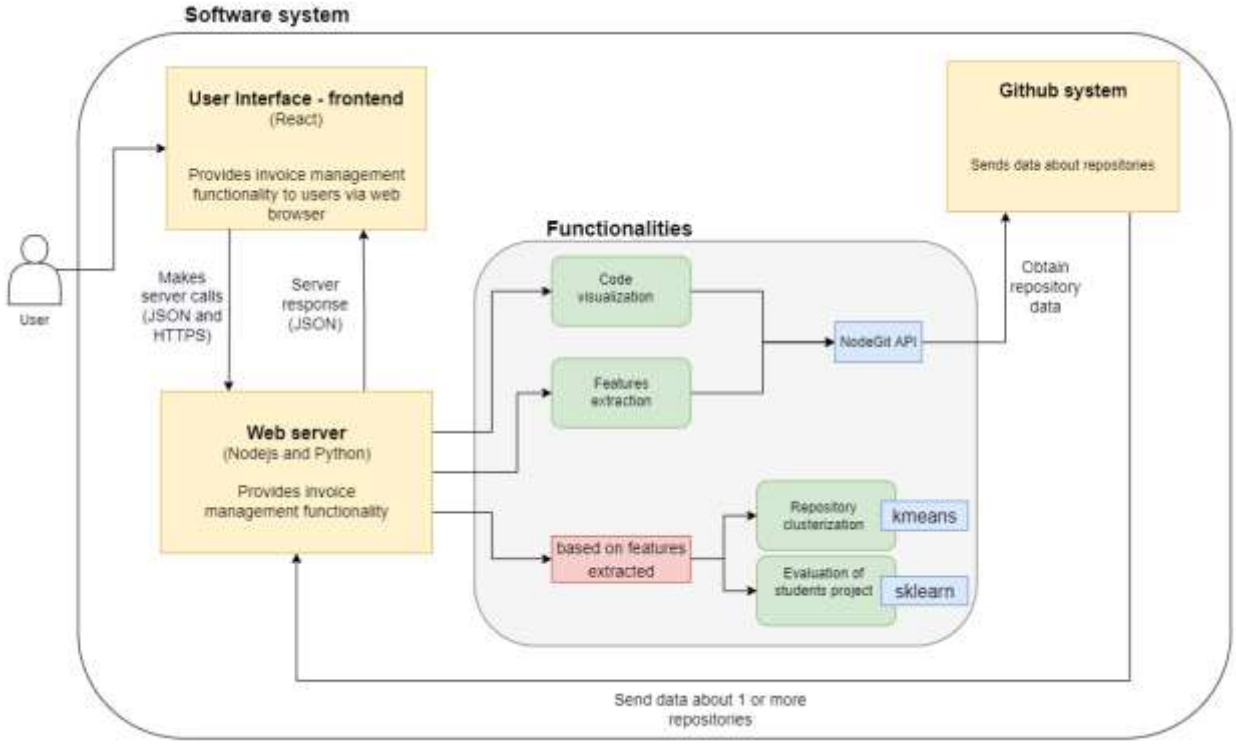

Fig. 4. System architecture

Figure 4 shows the architecture of the system, the application frontend, its communication with the server, the functionalities that it can solve at the request of the user and what tools it uses to provide the desired answer. It shows how the 2 methods described in the first paragraph are divided into different modules each one working on different task to resolve the request made by user [7], [8].

For the first method we created a transition using a stack of div, each one having an Editor type component, with an unique index. In order to be able to control what is displayed at a given moment, we used a slider that allows the selection of a state and specific buttons (forward, backward, stop and play). Initially all the div in that stack, except for the first one which is considered the start page, are not visible. When the slider is moved to a new value, all the div in that stack will have display property none, except for the selected value that remains with the display property on the block (that index is always considered, which is sent via an event handler). For a smooth transition (like one from a YouTube video) we used a method that calls a function and runs some code after specific intervals of time (the change of display property for a specific div mentioned above).

The content of each Editor component is obtained using the nodegit library from npm package. After entering the GitHub link, the server downloads repository locally, assign a unique name to that directory and creates a list of commits and a list of files present at the last commit. If a file is selected, using the previously created lists, the handle event function will iterate through each index, checks if that file existed at that time, fetches the code and adds it to a new list, which will be sent later as a response to the frontend. In that way, each Editor component will contain the file code from a specific commit during the development of the application.

The implementation of the second method can be divided into 2 directions: features extraction for a set of repositories and students evaluation based on that information. For feature extraction we used a function that takes each link from a given list, downloads the repository and goes through each commit separately. At the time of browsing certain properties will be calculated: total number of commits, number of commits made each

month in order to determine frequency contribution during application development. To obtain the changes made (lines added, deleted or modified) for each commit we calculated the difference between it and the parent commit. Each diff (the result calculated before) is split into packets (each packet representing a file) and these in turn are split into chunks of code that represent the changes made between commits for a certain file. Using these extracted features, we can calculate other statistics that would be useful, such as the changes made on each commit or the changes made each month.

Another functionality that will help to evaluate students is to detect if there are some similarities between their repositories. We created a system that groups repositories based on featured extracted earlier. To achieve this for each repository a feature vector is created. Having this vector we calculated the Euclidean distance (see Equation 2, where p and q are 2 points in the Euclidean space, and q_i and p_i are 2 vectors that represent those points) between each repository, in order to build a matrix of distances where the line (1, 2, 3, ... , n) represents the respective repository and the column the distance with the others. Based on the matrix created before, using k-means algorithm we were able to determine the labels (the students that have similar time contributions during the application development), setting a number of cluster that we want to obtain.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Equation 2. Formula for Euclidean distance

4 Results

To see if the solution proposed by these 2 methods is reliable and does not require long execution times for a repository that has a large number of files and commits or for a set of repositories that must be evaluated, we calculated the response time that the user receives from the moment the request is initiated.

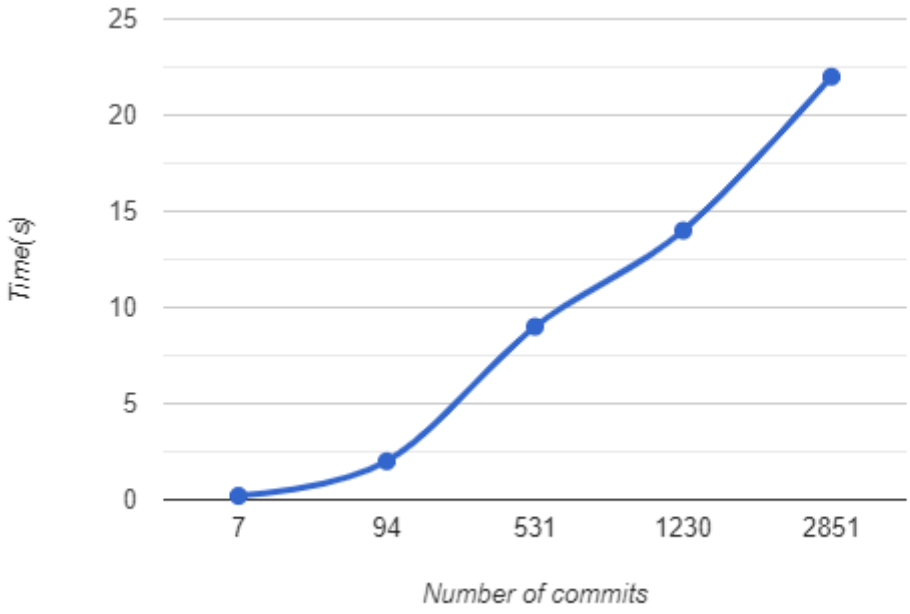

Fig. 5. Time to download a repository and parse all the files

The graph (Figure 5) shows the times in which the platform can parse a repository. We tested on 5 repositories with different sizes (7, 94, 531, 1230 and 2851 commits, respectively).

To this time is added the time for parsing the source code of a file when it is chosen, but this operation increases the time by a maximum of 3 seconds (sometimes this time is not even noticeable).

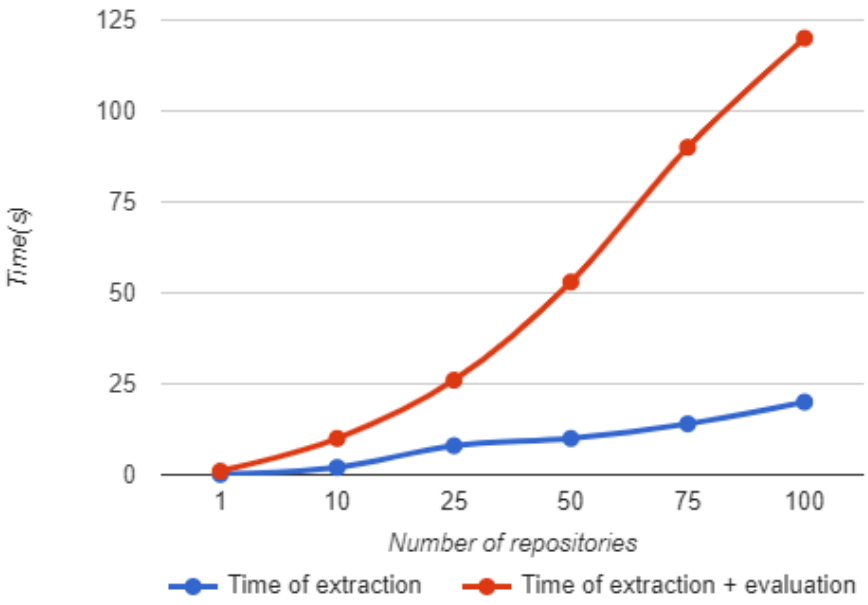

Figure 6. Features extraction time

In the graph (Figure 6) we calculated extraction times for 1, 10, 25, 50, 75 and 100 repositories. As can be seen, the time to extract the features of that set and the final grade based on them is much higher than if we did not calculate that grade as well (for 100 repositories the time increased by about 120 seconds). This is due to the time to predict the grade using the created model. However, the times that the platform can produce are very good, as this is used for teaching purposes to evaluate a group of maximum 20-25 students.

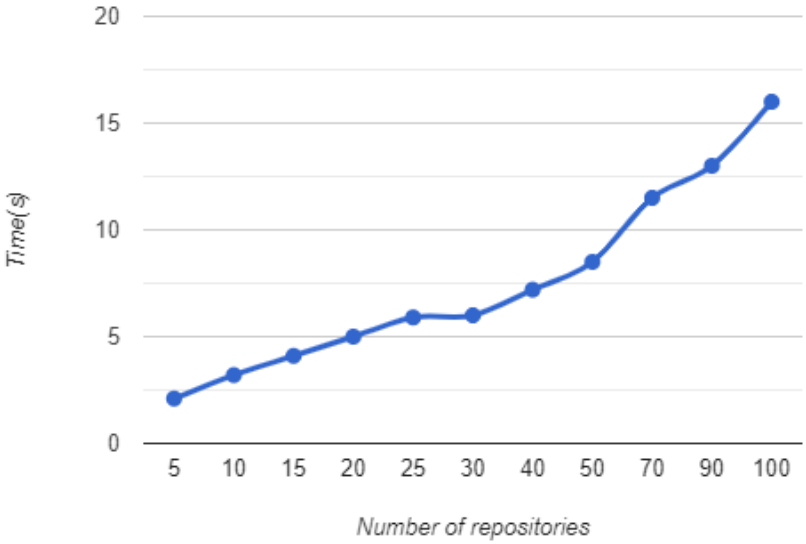

Figure 7. Clustering algorithm time

In the Figure 7 are represented the times in which the platform can perform the clustering of the repositories based on the extracted characteristics. We performed this test with 5, 10, 15, 20, 25, 30, 40, 50, 70, 90 and 100 repositories as input to the clustering algorithm. The number of labels is 4 and the number of iterations of the algorithm is 12.

5 Conclusion

There are some limitations to our analysis, the prediction grade and clustering repositories based on features extracted can be upgraded for a better evaluation of students. Currently, k-means algorithm doesn't provide always a corect answer and for future work, we will try to use different algorithms like DBSCAN (Density-Based Spatial Clustering of Applications with Noise) or other ones that can fit perfectly on our dataset. For prediction grade, we need to create a bigger dataset and to see what feature we actually need for a correct grade prediction.

References

[1] Dall, C., & Nieh, J. (2014, March). Teaching operating systems using code review. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 549-554).

[2] Cipriano, B. P., Fachada, N., & Alves, P. (2022). Drop Project: An automatic assessment tool for programming assignments. SoftwareX, 18, 101079.

[3] Krusche, S., Berisha, M., & Bruegge, B. (2016, May). Teaching code review management using branch based workflows. In *Proceedings of the 38th International Conference on Software Engineering Companion* (pp. 384-393).

[4] Boncea, R., Zamfiroiu, A., & Mitan, E. (2018). Proposing algorithm to improve student evaluation process. In *EDULEARN18 Proceedings* (pp. 5799-5805). IATED

[5] Qu, L., & Johnson, W. L. (2005, May). Detecting the learner's motivational states in an interactive learning environment. In Proceedings of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology (pp. 547-554). IOS Press

[6] Parizi, R. M., Spoletini, P., & Singh, A. (2018, October). Measuring team members' contributions in software engineering projects using git-driven technology. In *2018 IEEE Frontiers in Education Conference (FIE)* (pp. 1-5). IEEE.

[7] Guerrero-Higueras, Á. M., Matellán-Olivera, V., Costales, G. E., Fernández-Llamas, C., Rodriguez-Sedano, F. J., & Conde, M. Á. (2018). Model for evaluating student performance through their interaction with version control systems. *Proceedings of the Learning Analytics Summer Institute Spain*

[8] Wang, S., Jager, L. R., Kammers, K., Hadavand, A., & Leek, J. T. (2021). Linking open-source code commits and MOOC grades to evaluate massive online open peer review. *arXiv preprint arXiv:2104.12555*.

Alexandru ALEXANDRU has graduated the Military Technical Academy "Ferdinand I", Faculty of Computer Science in 2022. Currently he started the Master program Internet System Engineering at Politehnica University of Bucharest and works in Ministry of National Defence as software developer.

Alin ZAMFIROIU has graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 2009. In 2011 he has graduated the Economic Informatics Master program organized by the Bucharest University of Economic Studies and in 2014 he finished his PhD research in Economic Informatics at the Bucharest University of Economic Studies. Currently he works like a Senior Researcher at "National Institute for Research & Development in Informatics, Bucharest". He has published as author and co-author of journal articles and scientific presentations at conferences.