The importance of using the CodeInsights monitoring tool to support teaching programming in the context of a pandemic

Nuno Gil Fonseca
Polytechnic of Coimbra
Technology and Management School of Oliveira do Hospital
Oliveira do Hospital, Portugal
Center for Informatics and Systems of the University of Coimbra
Coimbra, Portugal
nuno.fonseca@estgoh.ipc.pt

Luís Macedo
Center for Informatics and Systems of the University of Coimbra, Dep. of Informatics Eng., University of Coimbra, Coimbra, Portugal
macedo@dei.uc.pt

António José Mendes
Center for Informatics and Systems of the University of Coimbra, Dep. of Informatics Eng., University of Coimbra, Coimbra, Portugal
toze@dei.uc.pt

***Abstract*— This Innovative Practice Full Paper describes the use of a monitoring tool for teachers to assess students' performance and progress, improving their ability to make decisions and interventions in programming classes, in the context of the current COVID-19 pandemic.**

Considering that learning programming is not an easy task as well as the social, cultural, and educational diversity of the student population, we believe it is crucial that teachers have at their disposal up-to-date information on the learners' progress, skills, and difficulties to properly support them in gaining and maintaining a positive learning momentum.

Previously, we suggested the use of this monitoring tool to supplement the information teachers can obtain through direct observation in traditional face-to-face classes. However, in the context of the current pandemic, its use takes on new significance since, in most cases, face-to-face instruction has been suppressed, demanding new strategies to collect assessment data.

In this paper, we introduce some features of the system and explain how they can help teachers to support their students. Key findings from a field trial, in which the system was used for about a month and a half to support more than 60 students of an introductory programming course, are also presented. Due to COVID-19 lockdowns and stay-at-home orders, classes took place almost exclusively online via Zoom. During this time, the usage of the system enabled the teacher to monitor student progress regardless of when or where they were working.

In post-experiment interviews, the teacher who participated in this study stated that using the system was vital to deal with the challenges that distance learning entails. Similarly, student feedback was also very positive. Several students mentioned that they felt more confident while using the system, knowing that the teacher was able to track their work and give them personalized feedback whenever necessary.

Keywords*— *programming, monitoring tool, emergency remote education

I. INTRODUCTION

As our daily life is becoming more dependent on the use of technology, there has been a significant increase in the demand for information technology (IT) professionals [1]. Because of this demand, there has been a consistent growth in the number of IT-related formal and informal educational programs. At the same time, even programs not directly related to IT have been including computer science courses in their curricula, namely introductory programming courses. Unfortunately, these courses are among those that consistently show higher academic failure and drop-out rates [2]. Throughout the years, several causes for this outcome have been presented, such as: the students' background, gender, income, or motivation [3]. Pedagogical approaches, study methods and attitudes have also been pointed out as causes for difficulties [4][5]. Divjak et al. [6] stated that the difficulty of mandatory mathematics courses also had a significant weight in the students' attrition. Watson et al. [7] identified the high failure rates in the programming courses as the main reason for the high failure and drop-out rates in IT-related programs.

As several authors have mentioned, learning programming is not a straightforward task, and the students can experience difficulties during the learning process [8]-[10]. The main difficulties concern requiring high levels of abstraction and problem-solving capabilities, the complexity of the programming languages syntaxes, or the understanding of compilation errors. A recent study [11] funded by the US Office of Naval Research has shown that the level of knowledge of mathematics seems to be almost irrelevant when it comes to predicting the students' ability to learn programming. On the other hand, the authors observed a strong relationship between the students' ability to learn foreign languages and their aptitude to program.

In addition to these specific factors that affect programming performance, we must consider other wide-ranging factors, such as the students' different learning styles [12], working paces [13] or personalities [14].

For a successful programming learning experience, the students must practice on a series of interrelated concepts, namely by solving programming assignments. According to Robins [15], for a positive learning experience, students must acquire and maintain a positive learning momentum. The positive learning momentum occurs when a student thoroughly understands a particular topic, ensuring a better preparation and motivation to address the subsequent ones. In contrast, due to the relationship between the concepts, if a student is unable to understand and use a specific concept correctly, that will undoubtedly affect the comprehension and usage of the following ones.

Bloom [16], Raabe et al. [17], and Butler et al. [18] have stated that the guidance and assistance provided by the teachers are among the most significant factors that impact students' success, namely because their intervention is crucial to address the student's difficulties and thus maintain a

positive learning momentum. Therefore, to guarantee higher performance rates, it is essential that teachers can understand their abilities and difficulties in a timely manner, enabling them to adopt effective strategies.

In programming classes, students are frequently assigned with several "one-function" assignments to practice on a topic, in the classroom or as homework. Often, these assignments' solutions are not evaluated by the professor due to the high number of students and course works, which represents a relevant loss of information about the students' performance. As a result, students may struggle to understand a particular concept, which prevents them from proceeding. In addition, inhibition factors, such as anxiety in exposing their doubts before their colleagues, prevent the teacher from being aware of their difficulties. Even when students express their doubts, it can be hard for the teacher to understand their real difficulties and what they have already attempted to solve the issue in hand.

This lack of information about the students' aptitudes and struggles may sometimes lead the teachers to solve the problem themselves without adequately identifying and clarifying the problematic key aspects. The main problem regards the non-identification and clarification of the students' difficulties in a timely manner, which may eventually lead to drastic consequences in terms of their learning process.

According to data published by UNESCO [19], at a given time, due to the COVID-19 pandemic, more than 170 countries around the world implemented nation-wide educational institutions closure, which led to the massive adoption of what has been described as "emergency remote education" [20]. While proper eLearning courses have been available for several years and have specific characteristics [21] [22], the time constrains and the uncertainty that resulted from the pandemic did not offer enough time for teachers to fully adapt their "traditional" courses to the online format. In general terms, this was done using fully synchronous, fully asynchronous, or hybrid approaches, each of them with its own advantages and disadvantages. [23].

As highlighted by Crick et al. [24], computer science teachers were, as expected, amongst those that felt more prepared for this rapid transition to online classes. Nevertheless, in this study, several teachers mentioned that online teaching of some subjects, such as programming, was challenging.

The teachers already faced significant challenges to follow the students' performance in the context of face-to-face classes, in which the teachers can walk by the classroom and observe their progress. Obviously, this issue assumes a new relevance in the context of the COVID-19 restrictions, limiting the proximity between students and teacher, who experienced additional difficulties in monitoring the students' progress and struggles in programming tasks.

We have been developing and testing a monitoring tool with auto-grading features, aimed at providing teachers with a real-time insight into the performance of each individual student as well as the overall class performance, regardless of when and where the students are working. We have shown that the use of such monitoring tools was important to increase the teacher's access to information regarding their students' performance and progress in in-person classes [25]. Given the current circumstances, we had the opportunity to test the monitoring tool in the context of an introductory programming course that occurred remotely due to a lockdown.

In this paper we address some key features of the monitoring tool and analyze how they enable teachers to better support their students in a distance learning scenario. Subsequently, we display the main results from the previously mentioned experiment. Finally, we present some conclusions and provide some hints for future improvements.

II. Background

As previously stated, several authors have argued that students' performance is highly coupled with the teacher's capacity to detect their difficulties and react within useful time. Although possible, this proximity relationship between teachers and students may be hard to achieve.

The need to obtain information from the students to support the teaching process is well-known. Finding strategies to increase this proximity relationship has driven researchers to explore the advantages of using technology. In the programming field, we have been witnessing the proliferation of automatic grading/assessment systems. These systems were developed mainly to automatically assess/grade the students' assignments and provide them with some feedback. This way, the teachers are exempted from this highly important, but somehow monotonous and tedious, task [26].

One example of such systems is Retina [27], a Java tool that enables teachers to browse through the students' source code and have access to some statistics about their performance. TestMyCode [28] is another automatic assessment tool that collects snapshots from the students' code and allows teachers to browse through them and provide feedback when requested. Spinoza [29] also offers auto-grading features and some real-time information about the students' performance. After the final solutions are submitted, the students can run, debug, and comment on their colleagues' submissions. OK [30] is another example of an automatic grading tool that also collects code snapshots from the students and offers the teacher some basic statistics about their work, allowing to monitor performance. The Virtual Programming Lab (VPL) plugin for Moodle is an additional popular auto-grader tool [31].

Several commercial auto-grading systems were proposed in the last years, as well. Examples are CodeGrade [32], Voccareum [33], Codio [34], Gradescope [35], or CodeLab [36]. These systems are mostly focused on supporting the teacher in the grading component of their work, as well as in providing some manual or automatic feedback for the students. Other systems, like SnapViz [37], CodeBrowser [38] or ArAI [39], can be used by the teachers to browse/analyze datasets of code snapshots collected from the students. While these off-line systems have proven to be useful for identifying problematic topics and predict both the risk of dropping out and the overall performance of the students, they are used *a posteriori*, preventing a timely intervention.

To promote the desired proximity between teachers and students, some authors propose the use of "virtual/remote programming labs", in which teachers see the screens of all the students in real time, and can control their computers, if permission is granted. One example of the usage of such systems in the context of the current pandemic was presented by Garcia, et al. [40]. While this approach can allow a closer monitoring by the teachers, it is very invasive. Furthermore,

since the teachers see a "video stream" of the students' screen during classes, it is hard to access and interact, asynchronously, with the source code written by the students. Teachers also do not have access to data concerning the work done by the students out-side the classes.

Although the above-mentioned systems can assist students and teachers, in general, they do not offer many analytics about the students' performance. Some of them are highly coupled to a specific set of assignments, programming language, or development environment. Some others are not very flexible concerning the students' solutions or require much work from the teachers to prepare the assignments. In most cases, they are intended to be used in final submissions rather than during their building processes. In a context in which the teachers have limited presential contact with the students, we believe that it is crucial to have access to the information as soon as possible, particularly when there is still opportunity to intervene.

III. CodeInsights

Unlike most of the previously presented systems, which are mainly focused on automatically grading the final submissions made by the students, relieving the teachers from this task, CodeInsights was idealized, from the first moment, as a monitoring tool intended to provide teachers with useful information about the students' performance while solving the assignments, without compromising their autonomy.

When using CodeInsights to support programming classes, the students have access to a web-based application in which they can check the list of assignments provided by the teachers, read details on the assignment, and obtain information about their last attempt for each assignment (if any has been made). To submit their solutions for each assignment, the students can install one plug-in in the development environment of their choice (*e.g.*, Eclipse, IntelliJ, PyCharm, PHPStorm, VSCode or Atom) or use the web-based code editor provided by the system. In any case, each time they compile/run the code, a submission is automatically sent to the system to be processed. The code will be tested with a predefined set of test cases, and information about the tests results will be presented to the students along with some feedback on the most common reasons for failure in that specific assignment (in case of total or partial failure).

At the same time, the resulting information is also immediately available for the teachers, including in a web-based application. This way, the teachers could easily identify: (i) in which assignments the students struggled the most; (ii) the most common mistakes; (iii) students that were falling behind in their assignments, or, on the contrary, those fast finisher students to whom additional advanced works could be assigned, to keep them implicated in the learning process.

Using the information obtained from the system, the teacher can make different types of interventions, such as: adjust the teaching pace; provide additional examples, resources or assignments; as well as undertake more individualized and personalized interventions.

While a course may have numerous students, the system was designed to be used mainly to support the "hands-on" component of courses (*i.e.*, the programming labs) in a context where there are several lab classes, each one assigned to a different teacher and with a much smaller number of students. This way, teachers will only have access to information about the classes assigned to them (combined or individually).

Considering that, in the lockdown scenario, teachers and students are not allowed to meet in person and classes occur via online meeting platforms, immediate personalized feedback can be provided in these platforms' "private rooms". For out-of-class communication, a mechanism to send feedback on a certain attempt directly from our system is provided. In this case, the student will receive an email with the feedback and the corresponding source code.

While several reasons for plagiarism can be indicated [41], some more plausible than other, as Dawson [42] mentioned, it can be due to lack of understanding, eventually because of accumulated unaddressed doubts. As Eaton et al. [43] indicated, regarding plagiarism, teachers should adopt pro-active formative measures, instead of punitive ones. If plagiarism is already a critical subject regarding in-person assessment, it poses a higher challenge in online classes, in which the teacher's "control" is more limited. In fact, as Mokad et al. demonstrate, plagiarism increased considerably during the pandemic [44].

To address this issue, CodeInsights performs a simple real-time plagiarism check, which can be used by the teachers to support formative interventions [45]. For instance, this feature can be used to identify the assignments in which more students may be plagiarizing, which may indicate some issue regarding all students. On the other hand, it can be used to identify individual students that may be plagiarizing, and, by further analysis of the information provided by the system, identify the source of the difficulties in order to address them. Taking into consideration that the information is collected from the students in real time, it makes perfect sense to provide the teacher with a "real-time" perspective of their students' work. For this purpose, our system provides a "live" view of the students' work, in the form of a matrix, updated every *n* seconds (the default value for *n* is 15). One portion of an example of this matrix is presented in Fig. 1.

Fig. 1. Portion of an example of a live feed matrix.

In this matrix, each line corresponds to an individual student (the names were obfuscated), and each small colored rectangle represents the last attempt made by that student in the previous *n* seconds. The colors indicate: (i) if the source code has compilation errors (red); (i) no compilation errors, but the code failed in some test cases (yellow); (iii) no compilation errors, but the code failed in all test cases (orange); (iv) the code passed in all test cases (green); (v) invalid code (purple). Just by looking at this matrix, the teacher can have an overview of how the students are performing. However, hovering the cursor on each rectangle

will provide additional information, such as the corresponding assignment or a summary of the results of running the code with the test cases. If the teachers wish to view more details for a specific attempt, they can simply click on the desired rectangle. One example of the page with the details of an assignment is shown in Fig. 2.

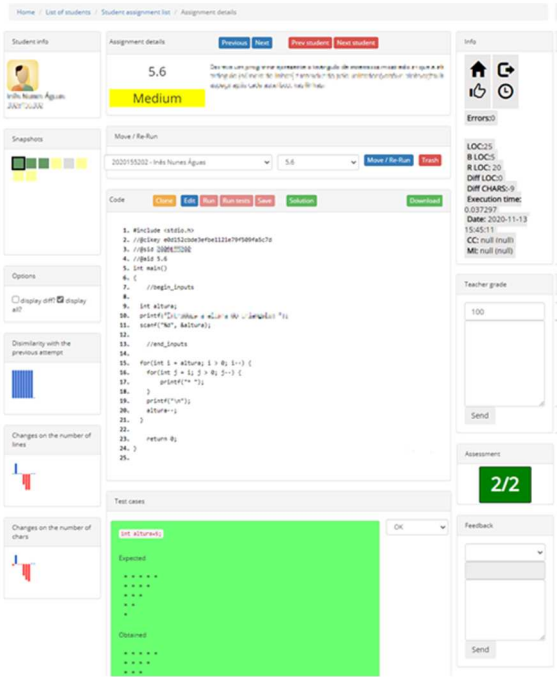

Fig. 2. Partial view of the page with details on the attempts made for an assignment.

In this page, the teachers can see the details of that attempt and navigate through all the other attempts for that assignment. For each attempt they can observe the source code (with or without the differences from the previous attempt highlighted), the results for the test cases and a series of metrics. They can also edit and test a temporary version of the source code for each attempt, without ever leaving the system. The teachers are also able to easily navigate to the other assignments attempted by a particular student as well as switch between students.

Besides the live feed, which is ideal for a view of what is happening at that moment, the system also offers an overview of the work performed by the students from all classes (or a particular one) in the form of another matrix, in which each square represents the "state" of the last attempt submitted by the students for each assignment. One segment of this matrix can be observed in Fig. 3.

Fig. 3. Portion of the summary matrix.

These two matrices can be used at the same time, in different browser tabs, for instance, allowing the teacher to have an insight on the students' works, even in a distance learning context.

Although CodeInsights was conceived do collect data on all the attempts made by a student to solve a certain assignment, for a variety of reasons, the students may only submit their last attempts (the one that they consider to be correct). These cases are easily identified, since the number of attempts made for an assignment is presented in different listings and visualizations.

When it happens, even though some data is lost, the teachers could still use that last attempt to address aspects to be improved. If the code has compilation errors or if the code does not pass the execution against the test cases, then it certainly has some problems that the teacher could help identify and overcome. On the other hand, if the code performs well in the tests, there could still be issues to be improved, such as the complexity of the solution. For this purpose, the system collects, for instance, data on the execution time.

Obviously, the system does not aim at removing the students' autonomy neither stimulating the teacher's intervention at the first compilation error. On the contrary, students are encouraged to try to solve the assignments on their own, even if it requires them to try other possible solutions and deal with the possible errors that this process may generate. Nonetheless, in some cases, the students may be very close to the correct solution and may not be capable to fully solve a specific assignment, even if they have indeed tried. In these cases, we hold that the teacher's timely intervention is crucial.

While the information is available to the teacher as graphical visualizations, the system also provides a configurable notification mechanism. Using this mechanism, the teacher can automatically be notified (directly in the system and by email) about the occurrence of various issues that might deserve his/her attention. Among these issues we find notifications concerning students with an abnormal number of submissions for a particular assignment, students that are struggling to keep the pace of their colleagues, or assignments with an unusual number of students unable to correctly solve them. This automatic notification system may be particularly relevant when dealing with a high number of students.

IV. Evaluation

The system was evaluated in the context of an introductory programming course in which the C language was used. The teacher had used the system in previous years, so he already knew how he could benefit from its use in a traditional context. Then, he expected that the use of the monitoring tool would be of even greater use in the new scenario.

The evaluation occurred during approximately eight weeks (between 14/10/2020 and 7/12/2020), and the system was used to support the teacher, and a total of 61 students, in the real context of their activities. Some remaining students, although enrolled, did not actually attend classes or participate in any activity related to this course.

During this period, the teacher provided six different worksheets, each of them related to a main topic, such as basic use of variables, conditions, loops, arrays, or file

manipulation. In total, these worksheets contained approximately 35 different assignments that the students were supposed to solve during lessons as well as outside this period. For all these assignments, the system processed a total of nearly 40,000 valid submissions. The distribution of these submissions through time is visible in Fig. 4., in which the x-axis represents time, and the y-axis represents each individual student. Each blue dot corresponds to an attempt.

In Fig. 4. it is possible to observe several aspects worthy of notice. For instance, the block marked with "A" corresponds to a group of students that were accepted to enroll that specific program in a late phase, which means that, when they began working on the first worksheet, most of their colleagues were already working on another one. Therefore, while the rest of the colleagues were working on new assignments, they needed to do additional work post-lesson to avoid falling behind. Besides these students, it is also possible to observe that, while there was some accumulation of attempts during classes, several attempts occur in another timeframe, especially in what regards two worksheets with some assignments accounted for the final grade (B).

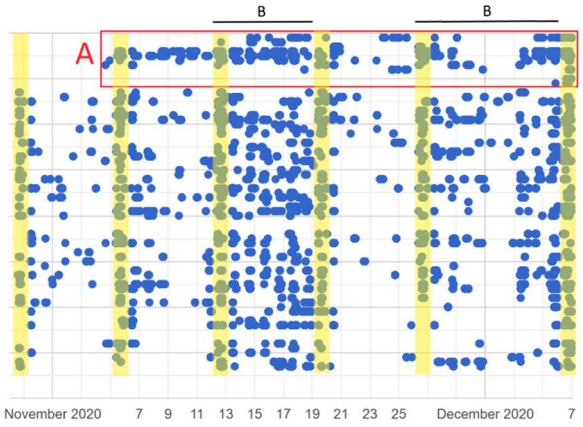

Fig. 4. Distribution of the students' attempts through time.

Another aspect worth of notice concerns the distribution of attempts during the day. These students had two moments in which the teacher was with them remotely during approximately a two-hour period each time. Although approximately 60% of the attempts were received during that period, a considerable amount was not, as depicted in Fig. 5.

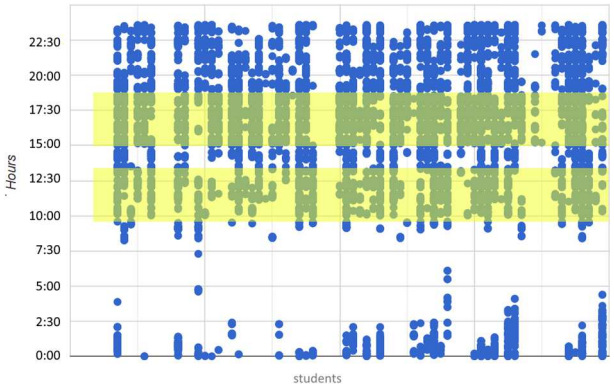

Fig. 5. Distribution of the students' attempts throughout the day.

Although the students were working in various periods of the day, understandably the teacher was not available to assist them in all those periods. Nevertheless, whenever the teacher accessed the system, he was able to check on what the students had been doing in these periods and take some measures if needed.

Even though the researchers had permanent communication with the teacher throughout the experiment, when it ended, there was a Zoom meeting to evaluate the system usefulness. In this meeting, the teacher expressed that the system was very useful in the online teaching context, particularly, during the first weeks, when the students had more difficulties in understanding the basic programming notions and syntax. By using the system, he was able to identify common mistakes and misconceptions, which allowed him to intervene or provide feedback to the students without downloading files or using a code editor, since everything was made within the system, even the manual grading of some assignments.

While he was "in class" with the students, he was connected to Zoom[1] and had a browser tab permanently open with the "live feed". Whenever he identified some issue that should be addressed for all students, he used Zoom to address it, for instance, presenting extra examples, or using the code written by a given student to explain what should (or should not) be done in a specific situation.

When students needed assistance, the teacher had immediate access to the last attempt of the student in a specific exercise. He could then use Zoom to give live support to the student, either by using a public or private room. The teacher also mentioned that it was common to receive e-mails from the students out-of-class, asking if he could check the attempts for a certain assignment. In this case, the teacher would use the feedback tool provided by the system to assist the student. He also mentioned that he frequently checked how the students were performing out-of-class and, when he found any important issue, he took adequate measures.

Concerning the real-time plagiarism check mechanism, the teacher reported that, particularly at the beginning of the experiment, he identified some students that were consistently submitting plagiarized solutions for the assignments, sometimes containing compilation errors or solutions that were not working as expected. According to the teacher, they were mainly students who did not attend classes frequently, and even when he provided feedback via the alternative means available, were not able to improve their solutions. Overall, the system achieved the goal of assisting the teacher in the process of identifying students with difficulties.

Since this teacher had used the system previously, he did not report any difficulties in using it. Nevertheless, he made some suggestions for improvements that were implemented, such as providing scaffolding code for the students, as well as some minor changes in the user interface.

According to the teacher's point of view, the system played an important role in helping him to monitor and assist the students, particularly in this situation where he could not meet them in person.

After the end of the experiment, in order to access students' opinions about the usage of the system, we have built a survey on Google Forms which was sent to all the subjects. The survey was available for one week and, during that period,

[1] Any other similar videoconferencing tool could be used.

a total of 25 responses was received. Due to the homogeneity of the group of students enrolling the course, in terms of age (around 18-20 years old) and gender (only six of the 25 students were female), as well as its anonymous character, a demographic categorization of the results was not considered relevant.

We began by asking the students how frequently they have attended online classes. Ten students (40%) answered "always", ten students (40%) answered "often", one student (4%) answered "sometimes", and finally four of them (16%) answered "rarely".

We proceeded by asking how frequently they have solved the assignments proposed by the teacher. Ten students (40%) answered "always", thirteen (52%) answered "frequently", and two students (8%) answered "rarely".

In a later question, we inquired students on the importance of using a monitoring tool in the current context. Seventeen students (68%) answered "very important", three students (12%) answered "important", four students answered "moderately important" (16%), and, finally, one student (4%) answered "not at all important".

To explore these responses, we additionally asked them to substantiate their answer. A total of 17 answers were received. Among the most positive ones we can underline answers such as: "*It's good to get a sense of where each student stands and provide personalized assistance as soon as needed*"; "*Because this way the teacher can follow our work and help us from a distance*"; "*Given the current situation we are in, I think this computer support is important*"; "*It gives a certain obligation to the student to practice and to the teacher a way to control what the student puts in, given the absence of face-to-face classes.*"; "*This type of system has become fundamental to the effectiveness of the teacher's clarifications of doubts*" or "*The teacher is more aware of the student's progress and can monitor what the student is doing.*"

As expected, for some students the experience was not entirely positive: "*Very sensitive to errors! Sometimes the program was correct, and it appeared that it was not*" or "*Sometimes all it took was a little different formatting than expected, and the exercise was marked wrong*". Being aware of these students' remarks about the system, efforts have been made to minimize the key issues. Nevertheless, since we are relying on comparing the expected and obtained results for each test case, we need to have some commitment from the students to follow the expected formatting, which in most cases really occurred.

As the classes took place remotely and, as we showed previously, an important part of the student's work was done autonomously outside classes, we asked the students if having their code tested automatically benefited their learning process. From the 25 students who answered the questionnaire, 23 (92%) answered positively and only two students (8%) answered the opposite. As we have mentioned before, the system does not aim at reducing the student's autonomy and increasing their dependency on the teachers. Therefore, testing the students code automatically and providing them some feedback when it is not behaving as expected is certainly a measure in that direction. In any case, as most students have recognized, it is also very reassuring knowing that the teacher was vigilant and could intervene even in situations in which the code has passed the tests, because, for instance, the solution was not the most efficient.

We continued asking about the perception that they had about the usage of the system by the teacher. Most students, 88%, answered positively, while the other 12% answered that they did not feel that the teacher was using the system. It is important to mention that there is a strong correlation between the answers to the last questions. In fact, students that did not perceive the usage of the system by the teacher were also those who answered that they rarely attended classes or attempted to solve the proposed assignments.

Subsequently, we asked the students to provide specific examples of the teacher's usage. We received 13 answers with more or less the same content, so we highlight some of them: "*During the lessons, the teacher could see the students' progress and helped them*"; "*Whenever we started the class, he always mentioned the mistakes previously made, because he had seen them in the system*"; "*When noticed that a student was not solving the exercises correctly, the teacher provided help.*"; or "*Some students were struggling, but the teacher, through the monitoring system, could notice this and helped the students or explained the subject/exercise better.*". These answers end up meeting our expectations on how the teacher used the system.

In previous usages of the system, we noticed that some students had issues concerning having all their attempts sent to the teachers, before finishing them. So, we asked these students if that made them uncomfortable. Most students, 72%, answered that they were not uncomfortable, while 28% answered affirmatively. Afterwards, we asked the students to substantiate their answers. Among the received answers, we highlight the following: "*In part, it is a little uncomfortable, but on the other hand we can have immediate help*"; "*I think it was too much pressure to have the teacher check in real time what we were doing but I know the goal was to help*", "*I answered affirmatively, because I felt pressured to do the best I could*" and "*The goal is to do the best we can, and the fact that the teacher is able to access the code has only given more opportunities for better results*". These answers are consistent with some from previous experiments. We noticed that the issue was not related to privacy, but rather to some psychological pressure regarding the fact that they were being watched, or a concern about their image before the teacher, based on their performance (unfinished code that, as such, may still contain errors).

Relating to the same topic, we asked the students if they had taken some measure to overcome the discomfort that they felt, and 72% of the students answered negatively. Most of the remaining students answered that they used a different IDE until they had a final solution, and only then submitted the solution in the system. Once more, this answer is also consistent with previous experiments. This is a situation that we would like to prevent by emphasizing the objectives of the system to the students.

Finally, we asked the students to evaluate if the system improved their learning process, by rating from 1 to 5. An equal number of 11 students (44%) attributed a grade of 4 and 5, which means that 88% of the students felt that the system's usage contributed to their success. On the other hand, two students graded the usage of the system with a neutral grade of three and one student attributed a grade of two. Once again, the "less positive" grades were given by those students who did not attend classes or attempted to solve the proposed assignments frequently.

V. Conclusions and Future Work

Overall, the system has helped both teacher and students in this uncertain pandemic context and most students understood the importance of using the system.

As previously noticed, using the system was essential to the students' success, by allowing the teacher to monitor more accurately their progress, and to enabling him to make interventions whenever he felt necessary. Nevertheless, we have identified some aspects that can be improved. One of them, as aforementioned, involves improving the effectiveness of the comparison of the obtained and expected results for the test cases. Another example concerns the need to rely on third-party applications, such as Zoom or email, whenever the students want to request personalized assistance from the teacher. For this purpose, we are currently developing a feature that allows the students to request help directly in the system, by using a chat inspired approach. We are also working on a feature that allows the teacher to invite students to live code editing on one of their attempts.

After this experiment ended, the COVID-19 pandemic and the resultant lockdowns persisted. Consequently, online distance learning was again the only possible option, corroborating the usefulness of the system in this setting, which was, once more, successfully used to support teachers in charge for other two programming courses in different universities.

Whether the pandemic and the lockdowns continue or not, this experiment provided relevant insights on how to improve the learning process in remote learning through using the CodeInsights monitoring tool to support teaching programming. We have carried out a series of experiments in the past [25], also in the context of face-to-face classes, in which the system has also proved to be very useful for both teachers and students in that context. In conclusion, regardless of its usage in distance (voluntary or not) or in more traditional face-to-face classes, the system has proven to be efficient in assisting teachers and students.

Until now, CodeInsights has been tested in controlled environments and is not yet available for the public. While the authors plan to make the system available for anyone shortly, in the meantime, any instructor interested on using it in their classes can contact the authors directly. Nonetheless, independently of the system used, our research shows that collecting data from the intermediary attempts made by the students can indeed provide useful information to be used by the teachers to assist their students.

Aknowledgements

The authors would like to thank the teacher and the students who participated in the evaluation presented by this paper.

References

[1] T. Hüsing, W. B. Korte and E. Dashja, "Trends and forecasts for the european ICT professional and digital leadership labour markets (2015-2020)", 2015.

[2] M. N. Giannakos, I. O. Pappas, L. Jaccheri and D. G. Sampson, "Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness," Education and Information Technologies, vol. Volume 22, p. 2365–2382, 2017.

[3] K. Kori, P. Margus, Ä. Leijen and E. Tõnisson, "The Role of Programming Experience in ICT Students' Learning Motivation and Academic Achievement," International Journal of Information and Education Technology, Vols. Vol. 6, No. 5, pp. 331-337, 2016.

[4] A. Gomes, M.J. Marcelino, F. Correia and A.J. Mendes, "Study Methods in Introductory Programming Courses," in EDUCON 2020 - Proceedings of the IEEE Global Engineering Education Conference, 2020.

[5] A. Lishinski and A. Yadav, "Motivation, Attitudes, and Dispositions", in The Cambridge Handbook of Computing Education Research, Cambridge University Press, 2019, pp. 801–826.

[6] B. Divjak, O. Mirela and V. V. Hains, "Sustainable student retention and gender issues in mathematics for ICT study," International Journal of Mathematical Education in Science and Technology, vol. 41, pp. 293-310, 2010

[7] C. Watson and F. W. Li, "Failure rates in introductory programming revisited", in Proceedings of the 2014 conference on Innovation & technology in computer science education, Uppsala, Sweden, 2014.

[8] T. Jenkins, "On the difficulty of learning to program," in 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences, 2002

[9] A. Gomes, and A. J. Mendes, "Learning to program-difficulties and solutions," International Conference on Engineering Education–ICEE. Vol. 7, 2007.

[10] Y. Qian and J. Lehman. "Students' misconceptions and other difficulties in introductory programming: A literature review." ACM Transactions on Computing Education (TOCE) 18.1, 1-24, 2017.

[11] C.S. Prat, T.M. Madhyastha, M.J. Mottarella, and C. Kuo, "Relating Natural Language Aptitude to Individual Differences in Learning Programming Languages". Sci Rep 10, 3817 (2020).

[12] M. Maia, D. Serey and J. Figueiredo, "Learning styles in programming education: A systematic mapping study", in IEEE Frontiers in Education Conference (FIE), Indianapolis, IN, USA, 2017.

[13] J. Carter, S. White, K. Fraser, S. Kurkovsky, C. McCreesh and M. Wieck, "ITiCSE 2010 working group report motivating our top students", in ITiCSE-WGR '10 Proceedings of the 2010 ITiCSE working group reports, Ankara, Turkey, 2010.

[14] G. R. Pike, "Students' Personality Types, Intended Majors, and College Expectations: Further Evidence Concerning Psychological and Sociological Interpretations of Holland's Theory", Research in Higher Education, vol. 47, p. 801–822, 2006.

[15] A. Robins, "Learning edge momentum: A new account of outcomes in CS1," Computer Science Education - Vol. 20, No. 1, p. 37–71, 2010.

[16] B. S. Bloom, "The 2 sigma problem: the search for methods of group instruction as effective as one-to-one tutoring", Educational Researcher, vol. 13, pp. 4-16, 1994.

[17] A. Raabe and J. Silva, "Um Ambiente para Atendimento as Dificuldades de Aprendizagem de Algoritmos," in XXV Congresso da Sociedade Brasileira de Computação, São Leopoldo/RS, 2005.

[18] M. Butler and M. Michael , "Learning challenges faced by novice programming students studying high level and low feedback concepts", in ascilite 2007, Singapore, 2007.

[19] UNESCO, "Education: From disruption to recovery" [Online]. Available: https://en.unesco.org/covid19/educationresponse. [Accessed: 19-Jun-2021].

[20] A. Bozkurt, I. Jung, J. Xiao, V. Vladimirschi, R. Schuwer, G. Egorov, M. Paskevicius,. "A global outlook to the interruption of education due to COVID-19 pandemic: Navigating in a time of uncertainty and crisis", Asian Journal of Distance Education, 15(1), 1- 126, 2020.

[21] B. Means, M. Bakia, and R. Murphy, "Learning Online: What Research Tells Us about Whether, When and How", New York: Routledge, 2014.

[22] C. Hodges, S. Moore, B. Lockee, T. Trust, and A. Bond,. "The difference between emergency remote teaching and online learning". Educause review, 27, 1-12, 2020.

[23] K. Khotimah, "Exploring Online Learning Experiences During the Covid-19 Pandemic". In Proceedings of the International Joint Conference on Arts and Humanities (IJCAH 2020), p. 68-72, 2020.

[24] T. Crick, C. Knight, R. Watermeyer and J. Goodall, "The Impact of COVID-19 and "Emergency Remote Teaching" on the UK Computer Science Education Community", In United Kingdom & Ireland Computing Education Research conference. (UKICER '20). Association for Computing Machinery, New York, NY, USA, 31–37, 2020.

[25] N. G. Fonseca, L. Macedo, M. J. P. Marcelino and A. J. Mendes, “Augmenting the teacher’s perspective on programming student’s performance via permanent monitoring”, in 48th Annual Frontiers in Education (FIE’18) Conference, San Jose, California, USA, 2018.

[26] C. Wilcox, “The role of automation in undergraduate computer science education", In Proceedings of the 46th ACM Technical Symposium on Computer Science Education, ACM, 90–95, 2015.

[27] C. Murphy, G. Kaiser, K. Loveland and S. Hasan, "Retina: helping students and instructors based on observed programming activities", in 40th ACM Technical Symposium on Computer Science Education, New York, NY, USA, 2009.

[28] A. Vihavainen, T. Vikberg, M. Luukkainen and M. Pärtel, "Scaffolding students’ learning using TestMyCode", in 18th ACM Conference on Innovation and Technology in Computer Science Education, (New York, NY, USA, 2013.

[29] F. Abu Deeb and T. Hickey, "Spinoza: The Code Tutor," in International Conference on Computer and Information Science and Technology, Ottawa, Ontario, Canada, 2015.

[30] A. Head, E. Glassman, G. Soares, R. Suzuki, L. Figueredo, L. D'Antoni, and B. Hartmann, “Writing Reusable Code Feedback at Scale with Mixed-Initiative Program Synthesis”, In Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale (L@S '17). Association for Computing Machinery, New York, NY, USA, 89–98, 2017.

[31] J. Pino, E. Royo, and Z. Figueroa, “A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features”, International Conference on e-Learning, e-Business, Enterprise Information Systems, & e-Government, 2012.

[32] CodeGrade [Online]. Available: https://www.codegrade.com/ [Accessed: 19-Jun-2021].

[33] Voccareum [Online]. Available: https://www.vocareum.com/. [Accessed: 19-Jun-2021].

[34] Codio [Online]. Available: https://www.codio.com/. [Accessed: 19-Jun-2021].

[35] Gradescope [Online]. Available: https://www.gradescope.com/. [Accessed: 19-Jun-2021].

[36] UNESCO, “Education: From disruption to recovery” [Online]. Available: https://en.unesco.org/covid19/educationresponse. [Accessed: 19-Jun-2021].

[37] B. Evan and S. Jaime, "SnapViz: Visualizing Programming Assignment Snapshots", in Proceedings of the 18th ACM Conference on Innovation in Computer Science Education, New York, NY, USA,, 2013.

[38] K. Heinonen, H. Kasper, L. Matti and V. Arto, "Using CodeBrowser to Seek Differences Between Novice Programmers", in Proceedings of the 45th ACM Technical Symposium on Computer Science, New York, NY, USA, 2014.

[39] A. Ahadi, R. Lister and L. Mathieson, "ArAl: An Online Tool for Source Code Snapshot Metadata Analysis", in Twenty-First Australasian Computing Education Conference, January 2019.

[40] M. Garcia, J. Quiroga and F. Ortin, "An Infrastructure to deliver Synchronous Remote Programming Labs", in IEEE Transactions on Learning Technologies, doi: 10.1109/TLT.2021.3063298.

[41] F. Hattingh, A. Buitendag and M. Lall, “Systematic Literature Review to Identify and Rank the Most Common Reasons for Plagiarism”, InSITE 2020, pp. 159-182, 2020.

[42] J. Dawson, “Plagiarism: What’s really going on. Seeking Educational Excellence”, Proceedings of the 13th Annual Teaching Learning Forum. 13th Annual Teaching Learning Forum, Perth: Murdoch University, 2004.

[43] S. Eaton, M. Guglielmin, B. Otoo, “Plagiarism: Moving from punitive to proactive approaches”, IDEAS Conference 2017: Leading Educational Change. 2017.

[44] M. Mokdad and S. Aljunaidi, "Whither plagiarism in distance learning academic assessment during COVID-19?", 2020 Sixth International Conference on e-Learning (econf), pp. 1-5, 2020.

[45] N. G. Fonseca, L. Macedo and A. J. Mendes, “Using early plagiarism detection in programming classes to address student’s difficulties”, in Simpósio Internacional de Informática Educativa, Jerez de la Frontera, Spain, 2018.