Journal of Systemics, Cybernetics and Informatics (2024) 22(4), 41-46
https://doi.org/10.54808/JSCI.22.04.41

D-CIDE: An Interactive Code Learning Program

Lukas GRANT
Matthew F. TENNYSON
Jason OWEN

Computer Science and Information Systems, Murray State University
Murray, KY 42071, U.S.A.

ABSTRACT

This paper introduces D-CIDE (Distributed Classroom Integrated Development Environment), a tool that is designed to improve student-teacher interactions in programming classes. D-CIDE's main objective is to provide more meaningful interactions between teachers and students. Its goal is to create a more seamless and interactive learning environment for everyone who uses it. D-CIDE is a distributed integrated development environment (IDE), where the teacher (host) can manage and interact with the IDEs of all students (clients). It makes use of server-client interactions to allow live sharing and editing of code, making it a useful tool for demonstrating coding techniques and quickly addressing student questions. The front-end was developed using HTML, CSS, and JavaScript, and provides a way for the students and teachers to interact with each other. The back-end is made with JavaScript and NodeJS and handles data processing and transmission. The effectiveness of D-CIDE was analyzed through a classroom case study involving a small group of students. The study measured students' engagement, enjoyment, and learning outcomes using D-CIDE compared to traditional teaching methods. Results indicated an increase in student engagement and satisfaction when D-CIDE was used, as well as an improvement in students' learning experiences.

Keywords: Computer Science Education, Programming Curriculum, Distributed IDE, Teaching Tools.

1. INTRODUCTION

A problem many classrooms have, especially in the context of programming, is that the student-teacher interaction is often difficult to make seamless throughout the whole class session. Classes can have many students, which can be overbearing to a teacher if they are all asking for help at the same time. If every student needs help, it is going to take a substantial amount of time for the teacher to aid all of them. This becomes especially difficult when it comes to programming, due to the difficulty of the subject and the attention to detail required to solve some problems. When presented with a programming problem, there are typically many ways that it can be correctly solved, and even more ways that it can be incorrectly solved. This can lead to a student being confused about either why their program is not working, or why the teacher's solution is different although their own solution worked. The main point here is that teaching programming in the computer science field requires a large amount of interaction between the teacher and the student. The amount of time students have with a teacher each week is often very limited, so there is rarely enough time to give each individual the amount of time necessary for optimal learning in a traditional teaching environment.

This paper presents D-CIDE (Distributed Classroom Integrated Development Environment), which is an open-source tool that was developed with a goal to improve learning how to program in the classroom. It is designed to make student-teacher interaction more seamless, which in turn is intended to make the classroom learning experience more beneficial and enjoyable. It links the students' and the teacher's devices together over a LAN (local area network) which is applicable to a classroom setting. The tool is a Java development environment that links both the students and the teacher to the same server, and provides them each with ways to ease the learning process. The students are able to see what the teacher is typing within their environment, so the students get to watch any code being written live on their own devices. This is much like a lecture based around the teacher displaying their code via projector, but by using D-CIDE, the students get to watch it up close on their own screen. The teacher can then create a coding exercise for the students, and proceed to share it with each of them. The students can then all attempt the exercise that was created in front of them. The teacher has access to all the students' solutions as they are writing them and after they submit them. Not only does this make interaction quicker and easier, but it also allows the professor to use student work as examples for the rest of the class to learn from. This is all possible to an extent already, but this tool makes the process faster and seamless, which is important in a limited class time setting.

In the upcoming sections, we will take a deeper look at the specifics of the design of D-CIDE. In the literature review, we take a look at already existing research and tools in the field of technology in education, mostly focusing on programming education. The next section discusses both its front-end interface, and its back-end architecture, along with the installation process of the tool. This section explains the framework of D-CIDE and highlights its features that are included to enhance the teaching and learning experience in programming classes. The methodology is described in the following section. We discuss how D-CIDE was implemented in a classroom setting and the approach we used to test its effectiveness. We detail the process of data collection, including both student surveys and performance analysis, to assess the tools impact on different aspects of a classroom environment. Our results and analysis are then presented. Finally, we conclude the paper and discuss future work that can be done to further this research.

2. LITERATURE REVIEW

Some tools already exist that are intended to aid the teaching of programming. One of the more popular online tools used is Codecademy [1]. This is a tool that has a built-in curriculum and a large variety of lessons that can be taken by students. It is a very interactive programming teaching tool that lets students actively

solve programming problems and provides feedback and explanations on their solutions. While this is a useful tool, it does not necessarily add any value to a classroom setting, it more so replaces the classroom. The teacher can decide what lessons the students can take, but it is still preset lessons that the teacher did not create. This means there could be topics the teacher wants to be taught that are not provided by Codecademy. This tool takes the teaching away from the teacher and puts it into the hands of this tool. While this can be very beneficial for supplementary content for the students to delve into, it does not add value to the active teaching environment.

The effectiveness of using technology in teaching, specifically in the field of computer science, is well documented. Some tools already exist that make use of computer technology to improve the effectiveness of teaching. Studies show that innovative teaching methods significantly enhance students' learning outcomes in computer-related subjects [11]. Similarly, incorporating technology to help bridge learning and teaching gaps has been proven to be effective, especially in simulation-based programming education [12]. The broader role of computer technologies in enriching educational experiences is also well established [13]. Furthermore, the positive impact of technology-enhanced interactive strategies on teaching programming cannot be overstated [14]. These findings help validate the importance of tools like D-CIDE in the classroom, which use technology to enhance the teaching and learning of computer programming.

A tool that also fits into the realm of computer-based tools designed to enhance learning is LanSchool [2]. This is a tool that allows teachers to share their screens with all students on the LAN network while also granting them some other useful control features. The teacher can also view each student's screen, block certain web pages and programs like YouTube and computer games, instant message with students, and more. This tool does grant the connection that is desired, but it is a broader-scoped software that is designed more with a focus on monitoring students rather than teaching. While the teacher is able to display a visual representation of their screen onto students' screens, they are not able to instantly duplicate all the work they have done from their screen over to the students' screens. If the students want to be able to have the same work that the teacher has, they must manually type what the teacher has written onto their own device. Something this tool does grant is that the teacher can actively look at students' screens and take control to solve any issues that the student may be having.

The overall downside of using LanSchool for teaching programming is that it is not designed specifically for programming. There are many features of this program that can be added and improved upon to specifically focus on improving the teacher-student interactions for programming-related courses. D-CIDE is designed with programming as the main focus, allowing teachers to have meaningful feedback provided to them about what each student has submitted in a much cleaner and more efficient manner. It also keeps students from having the issue of making sure they perfectly copy what the teacher is writing. D-CIDE fixes some issues which makes the process more seamless.

Tracking students' engagement during lessons is important for effective learning, especially in a programming context. A practical approach involves monitoring their latest computer interactions [3], highlighting the need for real-time engagement assessment. D-CIDE builds on this concept by enabling teachers to view each student's work as it happens, ensuring active participation and correct problem-solving approaches. This feature not only keeps students on task but also guides their progress in the right direction. Similarly, engaging students with instructor solutions or partially solved problems enhances learning, as shown in studies on interactive programming exercises [4]. D-CIDE incorporates these strategies, offering a dynamic environment for multiple teaching methods and appealing to different student preferences.

The idea of keeping students engaged in the lesson is extremely important, with numerous studies showing the positive impact of retained engagement on learning quality [6-10]. Traditional lectures often have the challenge of maintaining attention while teaching meaningful content. Here, computer technology, and by extension D-CIDE, comes into play. This program is designed specifically for programming education. It provides opportunities for teachers to incorporate active participation into their lectures. By doing so, not only is the issue of engagement taken care of, but students also get to practice their programming skills, which is an important part of computer science.

The teacher being able to know how each student is performing on the problem at hand is important. It allows the teacher to provide help to the students who need it the most. This need is supported by insights on real-time analytics in interactive programming assignments [5], showing how this type of information can streamline the teaching process and make it more effective. Sometimes students are afraid to ask for help because they do not want to feel unintelligent. This can detract from their learning experience. With the teacher having real-time reports about what the student is working on, it takes that pressure off the student. With D-CIDE the teacher can easily look through their student's solutions and how their solution performed. This increases the amount of assistance the teacher can provide, while also letting the students receive help that need it but are too afraid to ask for it.

A distributed integrated development environment (IDE) is a platform that enhances the standard IDE concept by including server-client interactions. Unlike conventional IDEs that focus solely on assisting programmers in writing code efficiently, a distributed IDE introduces the element of collaboration and data sharing across a network. D-CIDE makes use of this concept by establishing a host (the teacher) who not only manages their own IDE but also has access to all connected student IDEs. This setup enables the host to view and edit the contents of each student's IDE and share resources from their own IDE with the students. Therefore, D-CIDE not only takes advantage of the usual benefits of distributed IDEs, such as enhanced collaboration and resource sharing, but also extends its functionality to facilitate a more interactive and productive educational environment. This approach contrasts with typical distributed IDEs by focusing specifically on the educational context, where studies have shown that real-time interaction and feedback are crucial for effective learning [15].

3. PROGRAM DESIGN

D-CIDE is designed with two parts. There is a front-end GUI that the teacher and the students see and use to interact with the program and a back-end that handles all the data processing and transfer. The front end is created with HTML, CSS, and

JavaScript, and the back end is created with JavaScript and NodeJS runtime.

Front-End

The front end is a website that has 3 different views. There is the landing page, the host view, and the student view. The landing page is very simplistic and minimalistic. An input field allows the users to enter the names that they wish to be identified by. A "Join" button is for the students to press to indicate to the server that they are a student. A "Join as Host" button is for the teacher to press to indicate to the server that they are going to be the host of the session and be in control.

The page the teachers will land on after hitting the "Join as host" button is shown in Figure 1. This page has a line of text at the top of the screen, a large code editor in the center, a console output right below the editor, a side panel on the right side of the screen that shows a preview of each student's screen that is connected to the session along with their name and a progress indicator circle, a submit button, a button to toggle sharing, a "reset statuses" button, and a privacy mode toggle button. All the buttons are below the console output section. The line of text is used to indicate whose code is being displayed in the code editor. The code editor is exactly as the name suggests, it is the place where the code is written. The console output is simple as well; it displays the output of the attempt to run the code within the code editor.

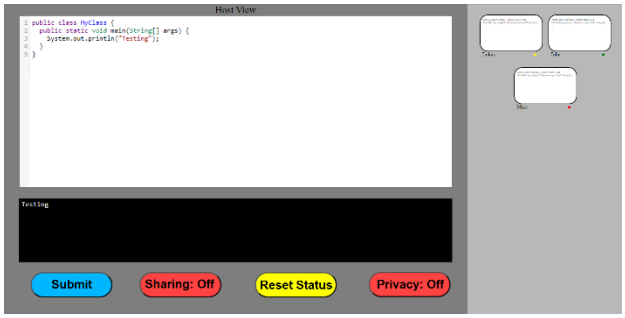

Figure 1. Host view.

The side panel is updated with a representation of each student active in the session once they join. They are given a small screen that shows the teacher the code the student currently has in their code editor, along with the name the student chose displayed underneath that screen, as well as their own progress indicator circle. Each of these screens is clickable to allow the teacher to edit the student's code. This process will be discussed later. The submit button causes the client to make a request to the server to run the code within the code editor. The toggle-sharing button sends a request to the server to enable or disable code sharing from the teacher to the students.

The reset statuses button sends a request to the server to change all the students' progress variables to "in progress," which is represented by the progress indicator being yellow. Finally, the privacy button simply modifies the HTML and CSS to hide the side panel with all the students' screens and names, along with the line of text above the code editor. The server-side functionality of all of these features will be explained when the back-end part of the program is discussed.

The page the students will land on after hitting the "Join" button is shown in Figure 2. This page is very similar to the page the teachers will see. The key difference is that the student view does not have the side panel, the share toggle button, the reset status button, or the privacy mode toggle button. The functionality from a client-side perspective is basically identical to the teacher for the line of text, the code editor, and the console output that is displayed on the students' screens.

Figure 2. Student view.

Back-End

The back end of the program has two main sections. One section is the JavaScript code that handles data processing, page updates, and server requests. The other section is the Node server which handles the storage and transfer of the data between the clients and the host.

The JavaScript code has a large section of code that runs at a one-second interval. This code handles almost all the functionality of the program. The short interval makes the updates appear to be in real-time, however, the interval is balanced to ensure that both the browser and server can efficiently process the data without being overwhelmed. Within each loop of the interval, all the code that the users have written is sent to and stored within the server, which is then handled in different ways depending on the user and the state of the program. The three possible states that the program can be in are "Sharing," "Not Sharing," and "Editing."

For the "Sharing" state to be active, the host must have the "Share mode" toggled to the on status. In this state, all code that is within the host's code editor is automatically pulled from the server into each student's code editor. The client is unable to edit the code while this mode is active, it just copies the host's code to their screen. This is useful for providing the students with examples of code, or incomplete code for them to look at while the teacher is walking them through it.

For the "Not Sharing" state to be active, the host must have the "Share mode" toggled to the off status. In this state, the host's code is no longer shared to each client's screen. The clients are now able to edit the code within their editor and submit the code to be run. This is useful for letting the students practice with the content they have just been taught. The teacher can provide incomplete code while sharing, then stop sharing and let the students finish the code.

The final state is the "Editing" state. For this state to be active,

the host must click on a student's miniature screen that they have in the side panel. How the side panel works will be explained in a later section. This then sets the program state to "Editing" for both the host and the student they clicked on. In this state, the host gets control over the student's code. The student's code is displayed in the host's code editor, and all changes the host makes on their device will be reflected on the student's device as well. This is useful for helping a student who is struggling with parts of the code they are working on.

The side panel is an important part of the host's part of the program. It is automatically updated with a new miniature screen and indicator dot for each student who joins. Each mini-screen holds a small preview of the code the respective student has written. Below this screen is their name and a small indicator dot. This dot can be one of three colors: green, yellow, or red. When the student submits their code, it is sent to the server and run by the host's device. If the student has not submitted anything and is still working on their code, then the indicator will be yellow. If they have submitted their code and it compiles and does not have any runtime errors, the indicator will turn green. If they submit their code and there is either a compilation error or a runtime error, the indicator will turn red. The indicator will remain this color until the host hits the "Reset Statuses" button, which will return all students to the yellow status.

The NodeJS server is how the host and clients communicate with each other. Within the JavaScript code that handles this server are multiple route handlers that handle the transmission of data between the client, host, and server. Examples of some of these routes are a route that grabs the code that each client has in their editor, a route that tells the server which client the host is editing, a route that sends the host's code to each client, a route that executes java code, among many others. This server is also the way the program is connected to each user. When it is running, it hosts a webpage from the host machine, using the IPv4 address of that machine. It will be explained in a later section how to get the program running on a machine. Once it is running, the host and clients can connect to the page by typing the host's IPv4 address, followed by the port the host has chosen, followed by 'main.html.' An example of what this would look like is '192.168.4.172:8080/main.html' This URL puts the user on the landing page.

Installation

For this program to work, the host must have NodeJS installed on their device. This can be acquired from the following URL: https://nodejs.org/en/download. Once this is installed D-CIDE can be installed from a GitHub repository. The URL for this repository is https://github.com/lukasgrant3102/D-CIDE. Once it is downloaded, it can be placed anywhere on the machine. To get it to run, the index.js file must be modified to have the appropriate IP and port. Towards the very bottom of the file, there are two lines of code that determine these values. There is one that starts with "let ip = …" and another that starts with "let PORT = …" These variables must be adjusted to get the server running properly. The PORT must simply be a port that is open on your network. The IP must be set to the host device's IPv4 address. This can be found by going to a command prompt and typing the command "ipconfig" on Windows and going to the "Network" settings in system settings and clicking into the "Properties" section on Mac. Once this is done, there will be a line that provides the IPv4 address. This IP must be typed into the IP variable in the index.js file. After all this is done, simply run the "D-CIDE.bat" file within the folder. This will print a URL into the console. All users should go to this URL to use the tool.

4. METHODOLOGY

To determine D-CIDE's effectiveness, a small case study was conducted in a classroom setting with a sample of 13 students in two class sessions. The study involved half the class using D-CIDE on alternate days, while the other half used traditional methods, and assessing student engagement and perception through quizzes and surveys.

Each student was assigned a random ID number to determine their usage of D-CIDE. Students with odd ID numbers used the program on the first day, while those with even numbers used it on the second. When not using D-CIDE, students utilized traditional tools or IDEs. The professor integrated D-CIDE into teaching, using it to demonstrate coding examples relevant to the day's topic. This approach's goal was to enhance the instructional process and keep students engaged in the lesson.

At the end of each session, students completed a quiz comprising two problems specifically designed around that day's topic. The quiz provided insight into the students' understanding of the material. Additionally, students filled out a survey at the end of class to measure their attentiveness and overall satisfaction with the lesson. The survey included additional questions for those who used D-CIDE, focusing on their experience with the tool. This data collection was important in being able to analyze the impact of D-CIDE on the students' learning experience.

5. RESULTS AND ANALYSIS

The questions that were administered to every student each day consisted of:

1) How much did you enjoy today's lesson?
2) How well do you think you learned the material from today's lesson?
3) How well do you think you paid attention during today's lesson?

The students had to respond to these questions with a value of 1 to 5, with 1 meaning "Not at all" and 5 meaning "Very well." Table 1 shows the average results for both days combined and Figure 3 shows a clustered bar chart visualization of the data.

Table 1. Summary of Survey Reponses.

	Not using D-CIDE	Using D-CIDE	Difference
How much did you enjoy today's lesson?	3.375	4.2	24.44%
How well do you think you learned the material from today's lesson?	3.75	4.5	20.00%
How well do you think you paid attention during today's lesson?	3.25	4	23.08%

As seen from the results, there tends to be an increase in the lesson's ratings from the students who used the program. There were no instances of decreased satisfaction. When looking at the data for both days, the minimum increase was 20% which is for the percent change of how well the students thought they learned the material for that day's lesson, and the maximum increase was 24.44% which is the percent change of how much the students enjoyed that day's lesson. The other question had a 23.08% increase, which was for how well the students thought they paid attention to the lesson. This shows that during this experiment, the students who were using the program felt as if they learned better, paid attention better, and enjoyed the lesson more compared to the students who were not using the program.

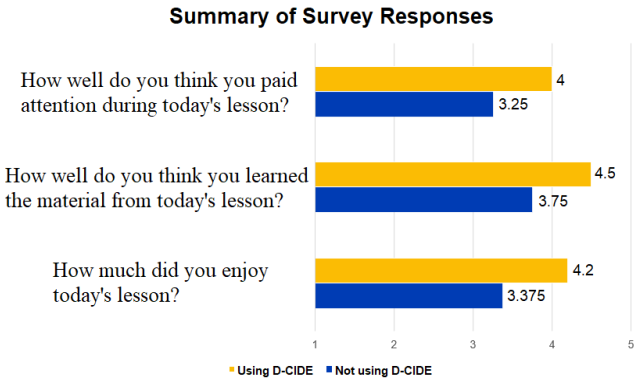

Figure 3. Clustered bar chart of survey responses.

The students who used the program were also asked about their experience while using the program. These questions included:

1) How easy was the program to use and understand?
2) What changes, if any, would you make to the program?
3) Please share any additional thoughts you have about the tool.

When asked how easy the program was to understand, the students were given a similar rating system to before with 1 meaning the program was very difficult to understand and 5 meaning the program was very easy to understand. The average for all responses over the two days for this question was 4.7. The ratings consisted of only 4s and 5s. This means that the students overall found the program to be very easy to understand and make use of during their class sessions.

Some of the responses for changes the students suggested consisted of simple changes such as "Submit button becomes highlighted when mouse hovers over it (to show it is clickable)," along with changes that are a bit more complex such as, "If possible, multiple file support." The change to the submit button is a small change but does hold value in making the program easier to understand. The change to make the program support multiple files is a very large change, but one that holds a lot of value in making this program more versatile for the topics it could be used to teach with.

The responses for the additional thoughts were mostly positive and had a few suggestions. They included responses such as:

1) "I like it, I think it will be very useful, especially for students who can't type as fast as now they can actually pay attention to the lecture instead of focusing on typing everything and not making mistakes."
2) "The tool works very well. Kept me engaged in class and I was really trying on the practice problems. It adds another level of accountability and almost makes you learn. I hope other teachers implement a tool like this."
3) "I think the program is great as a tool to help learning, but I believe it would work best in partnership with an external IDE. External IDEs have tools that are unnecessary for this program, but are still very helpful to use."

These students seemed to overall enjoy the use of the program and are feeling the benefits that the program is intended to have. The suggestion of having a partnership with an external IDE is a good one. The power of an IDE combined with the concept of connectivity this program provides would be very beneficial to this project.

A key point that must be discussed is the existence of bias. The answers being more favorable towards the tool could be due to this bias. The fact that the study was performed by professors and the tool was created by a fellow university student could make the participants more likely to approve of the tool. Furthermore, the study's investigators being present during the experiment could influence the responses of the students to be more favorable of the tool as well.

The averages of the students' scores with and without the tool are shown in Table 2. This data shows that the use of the tool seems to increase the scores by a very small percentage (3.03%). The problem with this data is that there is such a small sample size of students who were present both days and got to take a quiz both using the tool and not using the tool (8 students). This is a start to testing the tool's effectiveness, but more can be done to further analyze its impact on teaching.

Table 2. Summary of Quiz Scores.

	Score Using Tool	Score not using tool	Number of participants
Day 1	16.125	15	13
Day 2	18	16.8	8
Combined	16.3	15.82	21

When performing a paired t-test on the data of the students who took the quiz on both day one and day two, meaning they got a chance to use the tool and not use it, we determined the difference in scores was statistically significant. The paired t-test resulted in a t-stat of approximately -3.07 and a p-value of approximately 0.037. Since the p-value was less than the alpha of 0.05, we will reject the null hypothesis, meaning that there is a statistically significant difference in the scores between the day one quiz and the day two quiz.

6. FUTURE WORK

Looking ahead, several enhancements to D-CIDE are possible. Firstly, expanding its capabilities to include external web hosting would not only simplify setup but also extend its usefulness to virtual classrooms. A future study with a larger sample size and a more defined control group would provide more meaningful data, allowing for a better evaluation of D-CIDE's impact.

Additionally, programmatic improvements such as enabling

multiple file creation could significantly widen the scope of concepts that can be taught. Implementing user experience improvements – like removing the student's miniature screen from the host view upon their exit, allowing the host to view a student's screen without entering the 'Editing' state, making buttons responsive to mouse hover, and updating students' progress indicators more dynamically – could further refine the tool's applications and user-friendliness.

7. CONCLUSION

In this paper, we have demonstrated how the D-CIDE program (Distributed Classroom Integrated Development Environment) can help to streamline the teaching and learning process in programming education. Utilizing a LAN setup, D-CIDE has shown its applicability and effectiveness in a classroom setting, making lessons more engaging and accessible for both students and teachers. The experiment conducted, despite having a limited sample size, provided good insights, consistently indicating the benefits of D-CIDE in improving the educational experience. The tool, in its current form, stands as a meaningful aid in making programming education more seamless and interactive.

8. ACKNOWLEDGEMENTS

The authors would like to thank and acknowledge the reviewers of this paper, Dr. Victor Raj and Mr. Riza Marjadi. Dr. Raj is Professor of Information Systems and Department Chair of the CSIS Department at Murray State University. Mr. Marjadi is Systems Analyst and Adjunct Instructor of Computer Science at Murray State University.

9. REFERENCES

[1] J. H. Sharp, "Using Codecademy Interactive Lessons as an Instructional Supplement in a Python Programming Course," **Information Systems Education Journal**, vol. 17, no. 3, pp. 20–28, Jun. 2019. Accessed: Nov. 13, 2023. [Online]. Available: https://isedj.org/2019-17/n3/ISEDJv17n3p20.pdf

[2] B. Landry and A. Yrle, "The Use of LanSchool to Control and Enhance Management Lectures in Computer Classrooms," in **Proceedings of the Association of Business Information Systems 2006 Refereed Proceedings**, J. Hatcher, Ed., Renaissance Oklahoma City Hotel, Oklahoma City, Oklahoma, Mar. 2006, pp. 101.

[3] J. Edwards, K. Hart, and C. Warren, "A Practical Model of Student Engagement While Programming," in **SIGCSE 2022: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education** - Volume 1, Feb. 2022, pp. 558–564. doi: 10.1145/3478431.3499325.

[4] T. W. Price, J. J. Williams, J. Solyst, and S. Marwan, "Engaging Students with Instructor Solutions in Online Programming Homework," in **CHI '20: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems**, Apr. 2020, pp. 1–7. doi: 10.1145/3313831.3376857.

[5] N. Diana, M. Eagle, J. Stamper, S. Grover, M. Bienkowski, and S. Basu, "An Instructor Dashboard for Real-Time Analytics in Interactive Programming Assignments," in **LAK '17: Proceedings of the Seventh International Learning Analytics & Knowledge Conference**, Mar. 2017, pp. 272–279. doi: 10.1145/3027385.3027441.

[6] O. H. T. Lu, J. C. H. Huang, A. Y. Q. Huang, and S. J. H. Yang, "Applying Learning Analytics for Improving Students Engagement and Learning Outcomes in an MOOCs Enabled Collaborative Programming Course," **Interactive Learning Environments**, vol. 25, no. 2, pp. 220-234, 2017. doi: 10.1080/10494820.2016.1278391.

[7] O. Farrell and J. Brunton, "A Balancing Act: A Window into Online Student Engagement Experiences," **International Journal of Educational Technology in Higher Education**, vol. 17, 25, Apr. 2020. doi: 10.1186/s41239-020-00199-x.

[8] A. Rojas-López, E. G. Rincón-Flores, J. Mena, et al., "Engagement in the Course of Programming in Higher Education Through the Use of Gamification," **Universal Access in the Information Society**, vol. 18, pp. 583–597, Aug. 2019. doi: 10.1007/s10209-019-00680-z.

[9] Q. Hu, Y. Huang and L. Deng, "A framework to evaluate student engagement of programming course in blend learning environment," **2021 16th International Conference on Computer Science & Education (ICCSE)**, Lancaster, United Kingdom, 2021, pp. 915-919, doi: 10.1109/ICCSE51940.2021.9569553.

[10] L. A. Schindler, G. J. Burkholder, O. A. Morad, et al., "Computer-Based Technology and Student Engagement: A Critical Review of the Literature," **International Journal of Educational Technology in Higher Education**, vol. 14, 25, Oct. 2017. doi: 10.1186/s41239-017-0063-0.

[11] A. Zhang, C. J. Olelewe, C. T. Orji, N. E. Ibezim, N. H. Sunday, P. U. Obichukwu, and O. O. Okanazu, "Effects of Innovative and Traditional Teaching Methods on Technical College Students' Achievement in Computer Craft Practices," **SAGE Open**, vol. 10, no. 4, 2020. doi: 10.1177/2158244020982986.

[12] M. G. Jamil and S. O. Isiaq, "Teaching Technology with Technology: Approaches to Bridging Learning and Teaching Gaps in Simulation-Based Programming Education," **International Journal of Educational Technology in Higher Education**, vol. 16, 25, Aug. 2019. doi: 10.1186/s41239-019-0159-9.

[13] A. Haleem, M. Javaid, M. A. Qadri, and R. Suman, "Understanding the Role of Digital Technologies in Education: A Review," **Sustainable Operations and Computers**, vol. 3, pp. 275-285, 2022, ISSN 2666-4127. doi: 10.1016/j.susoc.2022.05.004.

[14] Y. Jiang, Z. Zhao, L. Wang and S. Hu, "Research on the Influence of Technology-Enhanced Interactive Strategies on Programming Learning," **2020 15th International Conference on Computer Science & Education (ICCSE)**, Delft, Netherlands, 2020, pp. 693-697, doi: 10.1109/ICCSE49874.2020.9201627.

[15] F. Coulon, A. Auvolat, B. Combemale, Y.-D. Bromberg, and F. Taïani, "Modular and Distributed IDE," in **Proceedings of the SLE 2020 - 13th ACM SIGPLAN International Conference on Software Language Engineering**, Nov. 2020, Virtual, United States, pp. 270-282. doi: 10.1145/3426425.3426947.