Using WebIDE as a distance learning tool for high school programming

Mirza Mesihović* and Vedran Ljubović* and Ibrahim Muharemović*
*Faculty of Electrical Engineering, University of Sarajevo, BIH

Abstract—During COVID-19 pandemics, not unlike their peers in other parts of the world, the high-school students in Bosnia and Herzegovina needed to get accustomed to learning from their homes. This proved to be particularly difficult for those who were scheduled to learn basic C++ programming, not only the in-person support by their teachers is invaluable in these early stages of learning, but many of them did not have access to adequate hardware or software. A web-based integrated development environment (WebIDE) was deployed, and a number of sample assignments were created. Altogether, over 130 students from 5 schools participated actively, solving a total of 65 assignments (as of June 1st). Based on questionnaire results, overall the experiment was deemed successful, and the continued use of WebIDE after life returns to normal was justified.

Keywords - distance learning; webide; introductory programming; K-12 education; COVID-19

I. INTRODUCTION

Before the COVID-19 pandemics, Bosnia and Herzegovina introduced digital technologies in education fairly slowly, especially in the secondary education. In the recent years there has been a considerable push to introduce the ICT to children in elementary schools (which in B-H lasts 9 years, typically including children between 6 and 14 years of age), especially in the Sarajevo Canton. Starting in 2016, the subject titled “Informatics” was included in the elementary school curriculum throughout the nine years. This subject is mostly focused on the appropriate use of computers, mobile devices and the Internet, various applications, covering topics such as critical evaluation of online content, cyber-bullying, programming etc. [1]

Since 2014, a number of elementary schools were equipped with educational tablets through “The School of Future” program. Starting in 2017, the relevant legislation of the Sarajevo Canton require that all schools that have adequate equipment should perform one week per school-year as an “online week” during which students stay at home and learn using the Internet. A variety of online platforms were used in schools, including Office 365, Google Classroom, and Moodle. The success of this pioneer project in the first year varied from school to school [2].

As expected, higher education institutions (HEI) were the most advanced in adopting digital technologies, and several schools have approved Distance Learning (DL) curricula, so several of them offered assistance with DL technology to elementary and high schools during the COVID-19 pandemics. One of the greatest problems that emerged during the pandemics was related to a number of students not having access to computers with adequate performances to use the prescribed online tools. Various organizations, groups and individuals scrambled to organize donations of PCs to school children.

In Canton Sarajevo, most high schools also have a subject titled “Informatics”, although it has only one lesson per week allotted, which in many schools includes C++ programming as part of its syllabus. Gimnazija – the comprehensive education high-school offers a broad education in a number of subjects, preparing students for enrolling into an university, and its second-grade curriculum includes C++ programming. In addition, “Information technology” programme has been offered by several schools starting in 2017 with a curriculum that includes more lessons allotted for programming. Schools are plagued with shortage of equipment, inadequate and outdated textbooks, and a lack of qualified teaching staff necessary to fulfill this curriculum.

Web-based integrated development environment (WebIDE) is a technology that has become mature in the last 5-10 years. It is well known that WebIDE offers certain advantages to educators. Installation and maintenance of computer labs is simplified, the only software that is required is a modern web browser.

Traditional integrated development environments (IDE) are fairly resource intensive, which presents a problem for students who can’t afford powerful computers, while WebIDEs can even be used (although not comfortably) from mobile devices such as tablets or phones. Students can access the exact same environment from home, school, or a cafe. Most importantly, while IDEs are usually professional tools that are not easy to learn or friendly towards novice programmers, WebIDE can be customized with various extensions for novice programmers in a way that is easy to develop, evaluate and deploy to a large number of students.

All this considered, when the in-class education was suspended due to the COVID-19 pandemics, a WebIDE platform previously in use at a Sarajevo HEI was offered to high-school students. The immediate goal was to assist those schools where C++ programming curriculum was interrupted due to students inability to use computer labs. Schools were not instructed on how to use the platform, but most used it as a tool for distance learning, as opposed to remote learning. Due to administrative procedures, this platform became available only on May 1st and was used until June 1st. A total of 137 students from five schools logged into the platform and used it to complete a number of programming assignments prepared by their professors.

There is some existing research on the use of WebIDE in higher education, however usage of WebIDE in K-12 education is insufficiently studied. This paper documents the experiences of using a WebIDE platform in high schools in
Sarajevo and attempts to fill this void in research.

II. RELATED WORK

Learning by doing is an integral part of modern CS education [3], [4]. This effort is thwarted by professional programming environments, which are invaluable tools in the hands of professionals, but to novice programmers they can often be overwhelming. A multitude of features and options can often present an excessive cognitive load for students, while the information about what is actually happening in the code is either missing or difficult to interpret. This was the rationale behind the decision to try and develop an educational IDE. Seminal work in this regard are BlueJ [4] and GILD [5], while more recent efforts include greenfoot [6].

In the recent years, improvements in web standards, web development technology and consumer hardware have made it possible to create an integrated development environment as a web application executing in a web browser. This technology was developed primarily to simplify application development in scenarios that would otherwise require complex cross-compiling and deployment strategies, such as embedded development (Raspberry Pi) and cloud development (Amazon AWS Lambda) [7].

However, the research in the area of computer-assisted learning technologies shows that WebIDE can be useful in programming education, and in particular to enforce Test-Driven Learning (TDL), where regular programming assignments are supplemented with tests to verify if student solution is correct. Research shows that groups of students who were taught using TDL show better results on tests and quizzes, and a more positive attitude towards the course [8], [9], [10].

Another topic that was studied in literature is use of the WebIDE to promote formal reasoning and specification in the process of software development [11]. The software development process is inherently iterative, and WebIDEs give an opportunity to provide instructive feedback in the form of verification conditions (VCs) at each iteration [11] which is in fact also closely related to TDL.

Using a web-based environment opens various opportunities for collaboration. Computer-Supported Collaborative Learning (CSCL) is an active field of research, however in the past it was mostly focused on education in social studies. Early attempts to use CSCL tools such as wiki, forum and chat in programming education have shown limited success [12], [13]. An interesting idea that was well-received is usage of an audio discussion feature [14], [13].

A more recent approach is a shared editing workspace where two or more students work on the same code or can see solutions by other students in real-time [15], [16]. While it is shown in literature that student satisfaction and attitudes towards such systems are generally positive, and the resulting code is of higher quality on average, it is difficult to show an improvement in programming performance of individual students [17]. Students who are used to working in a collaborative environment might show a decreased performance in individual tests.

On the other hand, there were other innovative ideas for collaboration. Codeopticon tool allows a tutor to see in real-time the coding workspace of many students and help them using a text-based chat [18]. These workspaces are read-only to tutor. Another useful feature is logged playback (also read-only) where the tutor can see how student progressed towards the finished task and which errors they made [19]. A common problem with text-based collaboration tools such as chat is difficulty in expressing the exact location in code where the problem exists. This, too, can be addressed in WebIDE [19].

A very common problem among novice programmers is their difficulty to understand various error conditions, either because error messages are confusing [20], [8], or due to inability to visualize the current program state. Some degree of visualisation can be provided by debugger tools, however they are usually very difficult to use. Our experience indicate that a low percent of students use debugger despite receiving a detailed instruction on its use. There is a wealth of research on code visualization techniques in programming education, and even collaborative visualization [21], however it would be beneficial if such tools were integrated with the development environment.

There is some anecdotal evidence that web-based environments are more popular among female students [10]. Škorić et al. [22] evaluate several available WebIDEs for use in an introductory programming course. CodeBoard was found to be the best. It should be noted that this research was published in 2016, several tools have emerged since that provide a much higher level of interaction.

III. DESCRIPTION OF IMPLEMENTED SYSTEM

Starting in 2015, a WebIDE platform was implemented at a course in introductory C and C++ programming delivered on a HEI in Bosnia and Herzegovina. To reduce the development costs, the institution opted to implement an existing open-source solution as opposed to developing its own. The initial research showed that none of the available WebIDEs provided all of the necessary features. However, Cloud9 IDE was selected as the most full-featured solution, in particular in regards for its C/C++ support.

The open-source edition of Cloud9 was a single-user software, so development of a new platform was started providing the missing support for multiple users, as well as an interface for teachers. This infrastructural software was developed in PHP, and its features include:

- User management (integrated with institutional LDAP server)
- Sandboxing – limiting users so they can’t access others environment without permission
- User monitoring and remote assistance
- Statistical information on usage
- Course management

Unfortunately, in the intervening time the Cloud9 project was acquired by Amazon, close-sourced, and further development on the project all but seized. For these reasons, current
work is mostly focused on porting the existing work to Theia project (http://theia-ide.org).

A. System architecture

The system currently has over 2000 registered users. At any moment, up to 300 users can be active simultaneously (note that users are automatically logged out after a certain period of inactivity). For each user, an instance of (single-user) Cloud9 environment is executed in a sandbox.

To facilitate this, a distributed architecture was developed specifically for WebIDE (Fig. 1). WebIDE cluster consists of one control node, one storage node and a number of compute nodes. Control node is the only publicly accessible node. It contains an nginx server which serves the dashboard and teacher interface (using php-fpm module), and also serves as a proxy server towards compute nodes using HTTP.

Compute nodes execute user Cloud9 instances which are served by nodejs server. By increasing the number of compute nodes, one can scale the system for a larger number of users.

Finally, all user files and configuration reside on storage node. These files are shared with compute nodes and the control node using NFS protocol. Storage node also holds monitoring service which tracks every file change and commits it to Subversion server. Cloud9 is configured to autosave all changes as they are typed, meaning that the system creates detailed usage traces, sometimes down to individual keypresses. Users also have Git repositories at their disposal where files are committed once each day.

B. Student user interface

Upon logging in, users are greeted by a dashboard showing system operational status and news. This also provides a distraction while user Cloud9 instance starts in the background. This dashboard includes a one-on-one chat using a free “website support” tool. For the duration of COVID-19 experiment, volunteer university students provided assistance to high-schoolers using the platform. In addition, various schools used tools such as Microsoft Teams and Google Hangouts to create chatrooms where students could discuss assignments with their teachers.

A number of Cloud9 plugins were developed using its native JavaScript API:

- Assignment plugin
- Send homework plugin
- Processing compiler output to provide a more user-friendly display
- Considerable work was done on improving debugger support and making it more user-friendly
- Integration with Valgrind profiler, offering dynamic (runtime) code analysis and detection of errors such as uninitialized variables or memory leaks
- A plugin for unit testing using our custom software testing platform
- Help plugin, which integrates with C and C++ documentation

Of greatest interest is the assignment plugin (Fig. 2), which provides a task-oriented view for the students as an abstraction of the actual filesystem. The assignments are easier to navigate since they are hierarchically organized by subjects, and then chronologically. Upon opening the assignment for the first time, a default code is generated from template, allowing half-finished assignments where students are required only to provide a portion of code or code snippet.

C. Teacher user interface

Teachers are presented with a list of courses, followed by a list of student groups within the course. For each group a window with detailed statistics is presented (Fig. 3). These statistics include: estimated time that each student spent solving an assignment, number of compilations and number of tests.

By clicking on the cell corresponding to a student and an assignment, teachers can see the exact code that a student is working on currently, with real-time updates [18]. Teachers can also replay the log and see how students’ codes evolved through time [19], and there is a textual log which is color-coded by types of actions performed by students (Fig. 4).

Teachers can create assignments by providing an assignment text, template code and tests. All of these fields are optional.
Fig. 3. Teacher user interface showing statistics per student and assignment.

IV. EVALUATION OF THE SYSTEM AND ANALYSIS

Between May 1st and June 1st, 2020, the system was used by 137 students from five schools, most of whom are in the second grade (15-16 years old) and units taught introductory C++ programming. It should be noted that the whole idea was not implemented earlier due to a number of administrative obstacles. The organization of online education based on the traditional IDEs for those who had the technical capacity to use them was already underway in most schools.

That was why it was impossible to assess the knowledge of students and their progress in programming performance. Instead, students and teachers were asked to complete a questionnaire that included a selection of questions providing an opportunity for self-assessment, and also to gauge their attitudes towards the system. A total of 89 students and five teachers completed this questionnaire.

The questionnaire includes seven questions for students and seven for teachers, each with answers on the Likert scale, as well as a textual field for comments. The questions facilitate both summative and formative evaluation [23], but also emphasize that the system could be improved in the future.

Before the analysis of the results, it should be said that the platform was offered as-is, with no particular instructions on how it should be used. The teachers opted to use it mostly in a distance learning scenario (as opposed to remote learning), that is, for the duration of an online class all of the students were expected to log in and work on assignments, and the students could continue working at a later convenient time.

Teachers were given some instruction on how to create assignments, but the time was too short to instruct them on how to create tests. There were a total of 65 teacher-created assignments. In addition, the platform was preloaded with a selection of material from university teaching experience (only those assignments that were adequate for high-school level), giving another 40 assignments with tests. Since this second group of assignments was not mandatory, less than 10% of students attempted to solve them.

The attitudes of students towards the system were mildly positive. Most students agree (but not strongly agree) that the assignments were easier to solve through the platform than they would be without it (Fig. 5), although over 40% strongly agree that the provided instruction materials and videos were sufficient for them to learn how to use the system (Fig. 6). Likewise, there is a very small bias of students agreeing to questions “The platform helped me learn more about programming” and “The platform motivated me to independently research and study”.

It should be understood that a large percent of students saw the new platform introduced this late in the semester as an additional burden and not as a tool to help them learn. Only seven students entered text into the free-form text field. Their biggest complaint was the confusing error messages,
which corresponds well to literature. Apparently, the C++ compiler on the server is more strict in some aspects than the one they used in school, so some language constructs that they were taught did not work.

Since only five teachers completed the questionnaire, it is difficult to analyze it statistically. Nevertheless, 4 out of 5 teachers strongly agree that the use of platform such as this could facilitate improvements in curriculum (the remaining teacher agrees). Teachers were divided on the issue of whether the platform helped the students learn more about programming.

Most students and teachers strongly agree that automated testing is useful learning and teaching in programming (Fig. 7). In addition, students agree very strongly that online help via chat tool was useful (Fig. 8) which somewhat contradicts the literature. A student and a teacher wrote in their comments that it would be better to have classical chat rooms than one-on-one chat.

Both students and teachers agree that it would be good to have such a tool in informatics education. Two of the teachers provided a detailed list of suggestions for teacher interface that would be better to have classical chat rooms than one-on-one chat.

Both students and teachers agree that it would be good to have such a tool in informatics education. Two of the teachers provided a detailed list of suggestions for teacher interface that will certainly help to improve its usability.

V. CONCLUSION AND FUTURE WORK

During one month of lockdown caused by COVID-19 pandemics, a WebIDE tool was used in high-school education, for introductory C++ programming. Results were analyzed from the perspective of system use patterns as well as a questionnaire for self-assessment.

The results indicate that WebIDEs are highly useful in elementary programming education. Test-driven learning (TDL) approach was welcomed by students and teachers alike, although the time was insufficient to use it to full extent. Online textual chat as a form of help is likewise useful. The few negative attitudes can be attributed to the overall depression of the epidemics as well as the novelty of the tool being introduced so late during school-year.

Teachers and students alike believe that a tool such as this should be used throughout the school-year, allowing them to get used to it.

Future work should be focused on improvements in two areas: clearer and more helpful error messages [8], and better visualization of program state (possibly by improving the debugger [20]). A community-based repository of content should, over time, provide a wealth of assignments with tests [10]. The work should continue on the chat tool, with focus on the ability to share code and point to specific locations in code [19]. Usage of eye-tracking is another promising research direction [24].

REFERENCES


Fig. 7. Answers to the question “Automated testing is useful for learning programming” (student questionnaire).

Fig. 8. Answers to the question “It was helpful to have someone available for questions” (student questionnaire).


