Informatics in Education, 2021, Vol. 20, No. 2, 297–315

DOI: 10.15388/infedu.2021.14

Student and Lecturer Perceptions of Usability of the Virtual Programming Lab Module for Moodle

Vinicius F. C. RAMOS, Cristian CECHINEL,
Larissa MAGÉ, Robson LEMOS

Department of Information and Communication Technologies,
Federal University of Santa Catarina (UFSC), Araranguá-SC, Brazil
e-mail: email@viniciusramos.pro.br, contato@cristiancechinel.pro.br, larissa.mage@gmail.com, robson.lemos@ufsc.br

Received: September 2020

Abstract. Teaching introductory computer programming and choosing the proper tools and programming languages are challenging tasks. Most of the existing tools are not fully integrated into systems to support the teaching-learning processes. The present paper describes the usability evaluation of the Virtual Programming Lab module for Moodle (VPL-Moodle) based on a satisfaction questionnaire answered by 37 undergraduate students enrolled in CS1 and CS2 courses and 7 lecturers. Moreover, a heuristic evaluation performed by two specialists is also presented. Results of the descriptive and inferential analysis revealed mainly two things: a) the VPL-Moodle has a low level of usability in all five aspects analyzed by the students: satisfaction, efficiency, learnability, helpfulness, and environment control; and b) lecturers found some difficulties using the VPL-Moodle. A number of suggestions for the improvement of the VPL-Moodle interface are provided based on the findings of the paper.

Keywords: Virtual Programming Lab (VPL), moodle, usability evaluation, design and evaluation methods, Human-Computer Interaction (HCI).

1. Introduction

Teaching introductory programming language, also called Computer Science 1 (CS1), is a challenge all over the world. The introductory course is one of the most important subjects in Computer Science and related undergraduate courses. Usually, that subject is offered at the very beginning of the undergraduate curriculum course, most of them in the first semester of the program. Jenkins (2002) says that the subject is sufficiently difficult to be under the first semester of the curriculum's course. Students are passing through a transition from high school to the undergraduate program, in an unstable moment, with various changes and difficulties. They are in a less restrictive environment with different curriculum and subjects that they used to study.

The failure rate in that subject is roughly 30% to 80% around the world (Bennedsen & Caspersen 2007; Mooney *et al.*, 2010; Watson & Li, 2014; Vihavainen *et al.* 2014; Ramos *et al.*, 2015). We highlight two main factors that increase the high rate of failure: the high abstraction needed to develop algorithms (Jenkins, 2002; Bennedsen & Caspersen, 2006; Dunican, 2002; Gomes, 2010), and the difficulties in solving problems (Jenkins, 2002; Dunican, 2002; Gomes, 2010; Mathew, Malik & Tawafak, 2019). Many studies suggest different approaches to reduce those rates (Mow, 2006; Porter *et al.*, 2013; Gomes, 2010), and many others pursue factors that affect the teaching-learning process (Canedo, Santos & Leite, 2018; Mathew, Malik, & Tawafak, 2019).

Despite the problems, teaching programming languages is extremely important in the contemporary world. Every technology needs software, and every software needs an algorithm. Thus, everyone should learn how to develop an algorithm. Teaching programming is considered, nowadays, almost an obligation to convert people from consumers to producers.

Some of the teaching-learning processes in universities and colleges have the support of a Learning Management System (LMS), such as Moodle[1], Sakai[2], Blackboard,[3] and Canvas[4]. The former, Moodle, is a free, open-source with over 199,000,000 users in the World, and present in more than 200 countries. From January to November 2020, the number of registered Moodle sites increased to near 70%, from 110,000 to 184,000. There are plenty of tools to support the learning programming process and an extensive review on automated programming feedback can be found in Keuning *et al.* (2018). Notwithstanding the integration with a LMS is essential, Caiza and Del Alamo (2013) identified tools with the goal to integrate them to a LMS to improve the performance of the programming assignments assessment process. However, Llana *et al.* (2014) stated that integrating tools "in regular programming teaching presents an obstacle: the overhead work required for the design of each problem, for compilation of problem collections, and for mundane management tasks".

The Federal University of Santa Catarina (Brazil) uses Moodle as a platform to support all the programs in the university, including undergraduate distance learning and traditional courses. To support lecturers in the computer programming subject, the university enabled the Virtual Programming Lab module (Rodríguez del Pino *et al.*, 2012; Thiébault, 2015).

The Virtual Programming Lab (VPL-Moodle) is a free and open-source Moodle module that offers automatic evaluation based on input and output cases, features to assist lecturers to create assignments, manage the submissions, check for plagiarism and do assessments. The VPL-Moodle executes and evaluates more than 40 different programming languages (Rodríguez del Pino *et al.*, 2012). Thus, this tool can help instructors evaluate the programs and provide students with timely feedback. Mory (2013), for example, says that feedback plays an important role for students to achieve their goals.

[1] visit `http://www.moodle.org`
[2] visit `http://www.sakaiproject.org`
[3] visit `http://www.blackboard.com`
[4] visit `https://www.canvaslms.com`

Feedback confirms or proposes changes to students' knowledge represented by the code provided by them to solve a problem.

Nielson (1994) relates usability with the ease of use and learnability of an interface, suggesting that usability improves the quality of the users' interaction. Though, improving the usability of software would reduce, for example, the number of times users spend discovering how the features work and where they are, avoiding the workload, and in aspects such as effectiveness and satisfaction it is directly correlated to motivation and engagement. In this direction, Medeiros, Ramalho and Falcão (2018) found that, for students, problem solving, motivation and engagement are the most cited challenges for learning programming and the lack of methods and tools are challenges for teachers.

In short, teaching and learning programming languages can be a hard task and problems in the interface of the main tool used in this process can have consequences to teachers and students. In that context, this paper aims to identify usability interfaces' problems in the VPL-Moodle. To tackle this challenge, it describes usability tests of the module applied to students and lecturers, and heuristic and ergonomic analysis of the VPL-Moodle interface interaction. The research questions that guide this study are:

RQ1 – Considering the usability factors defined by (ISO, 1998), how positive is the VPL-Moodle's interface to support students and lecturers in the teaching-learning process?

RQ2 – Which are the aspects of the VPL-Moodle's interface that students consider presenting problems?

RQ3 – Which are the aspects of the VPL-Moodle's interface that lecturers consider presenting problems?

The remainder of this paper continues with the review of the literature in Section 2 and the description of the problem context in Section 3. Section 4 depicts the methodology followed in this study, and Section 5 presents and discusses the results. Finally, Section 6 ends the paper presenting the final remarks and proposals for future work.

2. Related Work

There are already a number of studies focusing on the advantages of using e-assessment tools, but little attention has been given to tools focused on programming courses (Chirumamilla & Sindre, 2019). VPL-Moodle is an important tool for teachers to prepare programming activities as part of Moodle and for students who want to test and execute their codes inside a more controlled environment able to give them some sort of feedback about their codes. VPL-Moodle also is extremely secure as it uses an approach to separate code execution and data handling by a Moodle server (Kakadiya, 2020). Moreover, VPL-Moodle offers an anti-plagiarism check that helps teachers to verify the authenticity of students' code.

The impact of using VPL-Moodle in programming classes has been reported in the literature. Alatawi (2019) has observed significant differences in learning achievement

between students using VPL-Moodle in comparison to students not using it. According to the author, students who use VPL-Moodle achieve higher grades and develop better programming skills. Moreover, Skalka *et al.* (2019) investigated the impact of different types of e-learning activities with students from an introductory programming course and used VPL-Moodle to offer programming tasks. The authors stated that automated assessment does not harm students' performance and that students who solved more automated assessment exercises achieved better ratings in tests. Besides, Cardoso *et al.* (2020) conducted experiments using VPL-Moodle to teach Java classes and gathered the opinions of students and teachers involved. Both groups considered that VPL-Moodle added value to the teaching-learning process. The authors also highlighted the positive acceptance and participation of students and teachers during the experience.

On the other hand, Ribeiro *et al.* (2014) compared the use of VPL-Moodle with the use of a drag-and-drop code tool named iVProg. The authors reported that students who used VPL-Moodle presented a higher number of attempts and submissions than those who used the visual tool (iVProg). The use of VPL-Moodle in comparison to the visual tool led to more mental demand and effort for users to accomplish tasks, and even more frustration from the students while accomplishing more complex exercises. Those results strongly suggest that there is room for improvements to the VPL-Moodle interface. Moreover, Kaunang *et al.* (2016) conducted a survey for students to identify the weaknesses and strengths of an electrical power system course. Results show that students suggest that VPL-Moodle has a weakness in its online editor and has strength in its free-of-charge characteristics. Although the authors evaluated a course using online and offline surveys, they were not focused on evaluating VPL-Moodle. The authors presented 7 questions related to VPL-Moodle in a general sense. None of them were related to usability. Besides, Vanvinkenroye *et al.* (2013) evaluated a Web-based programming lab tool called ViPLab (Richter *et al.*, 2012). The authors implemented and analyzed a survey to get users' feedback, experience, and relate to learning success. The main results are: ViPLab is as efficient as classical tools and the use of ViPLab does not have any significant impact on learning success.

Although some previous works evaluated virtual programming labs, they essentially diverged from our research because they were only interested in validating the tool to use it in its technological course. In this paper, we are interested in evaluating the usability of the virtual programming lab by the students' and lecturers' perceptions, more than to validate the VPL-Moodle as an alternative for the classical tools.

While there is no work about evaluating the usability of a VPL-Moodle, there are already other works focusing on the evaluation of automated assessment tools (Daradoumis *et al.*, 2019), or the usability of educational software (Sarmento *et al.*, 2011; Chagas *et al.*, 2011; Junior *et al.*, 2016). An interesting work is the one conducted by Junior *et al.* (2016) who analyzed 14 different approaches for the evaluation of educational software. In that work, the authors were interested in the patterns and comprehensiveness of those approaches for the software quality literature. The results have shown the need for standardization of approaches for assessing educational software.

3. Problem Context

This research reports students' and lecturers' perceptions about the VPL-Moodle that is used in teaching programming languages within Moodle.

3.1. *Institution*

All results reported in this paper were collected and analyzed from a 100% free of charge to students, large, and public university called Federal University of Santa Catarina (Brazil). The academic year is divided into two periods of 18 weeks of classes, including tests, exams (final assessments), and recovery assessment: from March to June, and from July to December, each one is called a semester.

At this institution, students may drop classes up only in the first week of the course. On the other hand, freshmen can be enrolled in the first phase (the very first semester they enrolled at the university) subjects until the sixth week. Students receive a final score for each subject from 0 to 10. They have to reach, at least, 6 points to succeed and must have, at least, 75% attendance in the classroom.

3.2. *Course, Subjects, and Students*

*S*tudents are enrolled in a bachelor's degree program in Information and Communication Technology (ICT). The bachelor course accepts 50 students per semester. The classes work from 6:30 pm to 10:00 pm. It is considered a nightly course. Our research was applied in two subjects: Computer Science 1 (CS1) and Computer Science 2 (CS2).

In the first semester of the program, 50 students (freshmen) were enrolled in CS1, but only 16 were in classes to participate in this study, being 11 men and 5 women. There is a huge problem at the first night hour: some of the students live far from the university, in other cities, and the buses arrive from 15 to 45 minutes after the starting classes hour. That is why only 16 were in the class to participate in this experiment. The CS1 curriculum focuses on the development of software using the Python programming language, and it is, essentially, an introductory course for this programming language. In the second semester, students learn, in CS2, how to program in the C programming language. The subject had 32 enrolled students, but only 21 were available to participate in this study, being 15 men and 6 women. The missing 11 students did not come to this class and did not participate in the experiment. The average age was 21 years old. The youngest and the oldest one was 16 and 46 years old, respectively.

In CS1 and CS2, lecturers teach in a computing laboratory. The lab has a small number of computers, but most of the students bring their notebook, and, most of the time, only three computers, on average, are shared for 6 students. Both CS1 and CS2, have 3:20 hours per week of classes, divided into two days of 1:40 hours. There is no difference between theory and practice classes. It is important to highlight that the classes are in the lab, therefore lecturers are free to choose how to split the theory and practice over the semester.

3.3. *Lecturers*

The lecturers are associate professors at different campuses at the university, six of them with a Ph.D. degree: four graduated in computer science, one graduated in mechanical engineering, and one in applied math, and one master's degree in computer science. Two lectures have more than 15 years of experience in teaching programming and 4 of them have at least 5 years of experience. Most of the lecturers use the VPL-Moodle for over 3 years. The student's evaluation and its working sessions were applied with one of the lecturers with 7 and 3 years of experience teaching CS1 and CS2, respectively, and with 3 years of experience using VPL-Moodle.

3.4. *Heuristic Evaluation Background*

The heuristic evaluation was performed by two specialists. One specialist is an associate professor at the university and teaches human-computer interaction for over 3 years. The other specialist has been working with heuristic evaluation for over one year.

4. Methodology

An overview of the methodology followed in the research is presented in Fig. 1. The courses were taught with Moodle's support, and the lecturers included videos, tutorials, and programming problems as activities. Our first step is related to the working session, which is detailed in Section 4.1. The second step consisted of data capture. In this case, we divided it into two sessions: the student's questionnaire (see Section 4.3), and the lecturer's questionnaire (see Section 4.4). In the third step, researchers performed the data analysis (detail in Section 5). It is important to note that data analysis consists of three different analyses: analysis of the students' questionnaire, analysis of the lecturers' questionnaire, and heuristic evaluation performed by two specialists.

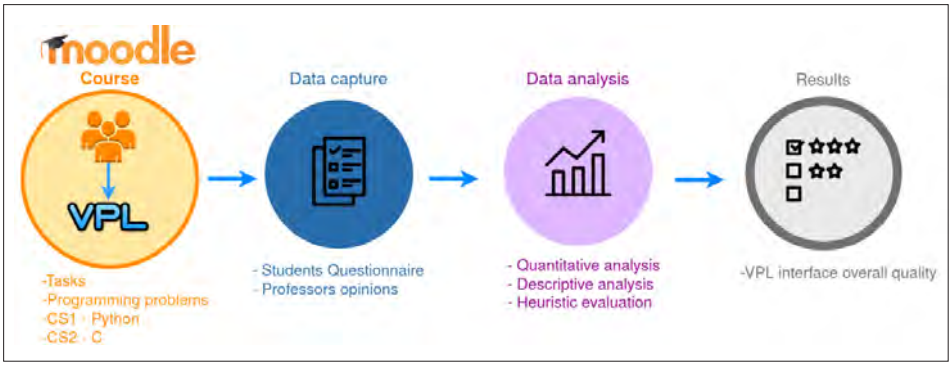

Fig. 1. Methodology overview.

Results are addressed in the last step and detailed in Sections 5 and 6. The results were obtained after using quantitative and qualitative analyses, such as descriptive and inferential statistical analysis (normality of the distribution, T-test and Wilcoxon test), and heuristic evaluation.

4.1. *Work Session*

The experiment is divided into three main steps for both CS1 and CS2 classes, concerning the time and the proposed tasks. The experiment was done in one class of 1:40 hours, as shown in Fig. 2.

At the beginning of the class, the researcher explained to the students about the experiment detailing that the experiment is not related to the subject, and they will not be evaluated about their activities during the proposed session and tasks. The researcher also highlights that the main goal of the experiment is to evaluate the VPL's interface. The only explanation about VPL-Moodle was that the module is like an IDE that can compile and run codes, and evaluate their correctness based on case tests. No other information was given to the students.

4.2. *Tasks*

The students were asked to solve and evaluate two programming problems using VPL-Moodle. The problems are different for each CS1 and CS2 class. The tasks contain simple problems because the main goal of the tasks is to let students explore the VPL's interface. Thus, the first task asked the student to calculate the square of a number and the second one was to calculate a percentage of a number, informed by the user.

4.3. *Students' Questionnaire*

The questionnaire was created based on the Software Usability Measurement Inventory (SUMI) (Kirakowski & Corbett, 1993), which is mentioned by (Bevan, 1998). SUMI

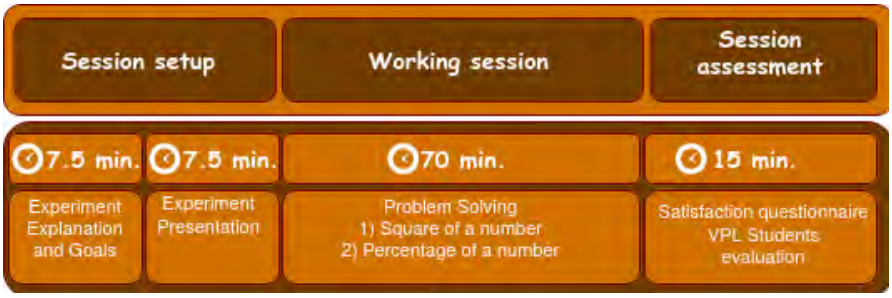

Fig. 2. Work session.

proposes five usability factors to be evaluated by the questionnaire, that is satisfaction, efficiency, learnability, helpfulness, and control. The questions of the questionnaire and the usability factor associated with each question are presented in Appendix A. The answers use a Likert scale of 5 points, that is 1- Totally disagree; 2- Disagree; 3- Not sure/ No opinion; 4- Agree; 5- Totally agree.

4.4. *Lecturers' Questionnaire*

Lecturers were invited to answer a questionnaire with two essay type questions asking for the positive points of the interface and suggestions to improve the interface, and 3 objective questions, each one associated with a usability factor, that is satisfaction, efficiency, and helpfulness. The questions of the questionnaire are presented in Appendix B. The answers use a Likert scale of 5 points, that is 1- Totally disagree; 2- Disagree; 3- Not sure/No opinion; 4- Agree; 5- Totally agree.

4.5. *Data Analysis*

We used descriptive analysis based on the responses obtained with the questionnaires to describe and consolidate the data obtained. In the specific case of the students, as a larger number of participants, we used Cronbach's alpha (1951) to determine the degree of reliability of the responses. Usually, it is preferred to have alpha values between 0.80 and 0.90, but not below 0.70 (Streiner, 2003).

5. Results and Discussion

In this section, we present the results of the analysis and answer the research questions proposed in Section 1.

5.1. *Descriptive Analysis of Student's Answers*

The study included a total of 37 students from two courses CS1 (16 students) and CS2 (21 students). Based on that, only 4 (11%) of the students reported having little experience with the VPL-Moodle, being all four from the CS2 course, all others said they did not have any previous experience. It is important to note that in CS1 and CS2 Moodle classrooms, there are VPL activities and the student is free to solve them, but there was no explanation about how to solve them or how to use the VPL-Moodle. Students that answered that have little experience with the tool mean that they tried to solve some of the proposed activities by themselves previously in the semester. None of them have had any other experience with VPL-Moodle before that.

As the students carried out their activities, a tutor helped them to solve the questions regarding the programming language, but without intervention regarding the exploration and use of the VPL-Moodle. However, two main questions have been answered for students to give continuity in the tasks, as described below:

Student_1 – "*Where should I send the file*?"
Student_2 – "*I finished, but I cannot find the button to send the activity. Where is it?*"

We clarify that, for Student 1 above, although it was explained that s/he (actually, it was explained for all the students) should perform the activity within the module text editor, some students wanted to send a file with the code previously written to start the programming activity. It is understandable because many of them (students CS1 class) had no more than eight weeks of classes (some students were enrolled in the subject at the end of the first month, and, consequently, had less than 4 weeks of classes). As the module allows students to send, and, later, to edit files, in this particular case, we gave two directions to the student. The first one was to send the file chosen by him/her and later editing. We did not detail, nor did we show the steps to be taken to carry out this activity. The second option, which was adopted by the majority, was to copy the previously written code and paste it into the VPL-Moodle text editor.

Student 2's question is directly linked to the experience of some in using Moodle activities. Most of these activities should be "sent" (a button with the "send" label) to the teacher's correction. In this case, since it was a resource exploitation activity, this button does not exist. The orientation, therefore, was to write the code, save it, and evaluate it (as mentioned before, there is an option for evaluation and automatic grade assignment). Thus, the activity would be stored in Moodle's database for further evaluation from the teacher.

With the analysis of the questionnaire, we answer research question 1 (RQ1).

Research question 1 (RQ1) is:

RQ1. Considering the usability factors defined by (ISO, 1998), how positive is the VPL-Moodle's interface to support students and lecturers in the teaching-learning process?

Table 1 presents the average of the students' responses separated by the CS1 (Python) and CS2 (C) subjects.

Table 1

Mean, Median, Standard Deviation, and *p-value* of the usability factors obtained in a questionnaire for CS1 and CS2 students

	Mean (M)		Median (Mdn)		Std. Deviation (SD)		*p-value*
	CS1	CS2	CS1	CS2	CS1	CS2	
Satisfaction	3.5152	2.8108	3.6667	3.0000	0.9243	0.8223	.0053
Efficiency	3.5227	2.9054	3.6250	2.7500	0.8342	0.7623	.0068
Learnability	3.8788	3.2523	3.8333	3.3333	0.7313	0.7634	.0030
Helpfulness	3.3864	3.0135	3.5000	3.0000	0.9377	0.7772	.1205
Control	3.7727	3.3063	3.8333	3.3333	0.7081	0.7386	.0202

Table 1 shows the mean, median, and standard deviation of the students' usability perception about the VPL-Moodle interface according to the five dimensions found in the SUMI questionnaire. This data was collected by the questionnaire (see Appendix A) replied to by the students during the working sessions.

It is important to clarify why these usability factors are important to the teaching-learning process. Learning computer programming is not an easy process and it can be an unbelievable hard task (Watson and Li, 2014). Further, student motivation and engagement are considered factors associated with the success and retention of the students (Bruinsma, 2004; Kori *et al.*, 2016). The five factors presented in Table 1 are important to the student's motivation and engagement in different levels: efficiency refers to the user feeling that the software is enabling the tasks to be performed in a quick, effective and economical manner or, at the opposite extreme, that the software is getting in the way of performance; satisfaction refers to the user feeling mentally stimulated and pleasant or the opposite as a result of interacting with the VPL-Moodle; helpfulness refers to the user's perceptions that the tool communicates in a helpful way and assists in the resolution of operational problems; control is the degree to which the user feels that he/she, and not the product, is setting the pace; and learnability is the ease with which a user can get started and learn new features of the tool (Kirakowski and Corbett, 1993).

Cronbach's alpha legitimizes the reliability of the questionnaire's replies with $\alpha = 0.8555$ for CS1 and $\alpha = 0.7959$ for CS2 (both above the minimum of 0.7). As it can be seen in Table 1, all factors presented relatively low averages with means ranging from 2.8108 to 3.8788. To consider the usability of the interface as satisfactory, mean values should be greater than 3.6 since the value seems to be a better estimate of "neutral" or "average" subjective satisfaction (Nielsen and Levy, 1994). The highest average of the factors is the learnability factor for CS1 ($M = 3.8788$) and, the control factor for CS2 ($M = 3.3063$). The learnability in CS2 ($M = 3.2523$) is close to the control factor. It is interesting to note that the VPL-Moodle interface is relatively simple for the students, having no more than one suspended menu and a text editor, as shown in Fig. 3. In VPL-Moodle the lecturer can select which "execution" options are offered to the students (execute, debug and evaluate), in the present work the lecturer chooses only the execute (it opens a terminal window and executes the code. The student can interact, if the code allows it.) and the evaluate options (VPL-Moodle automatically evaluates the code with input and output cases).

Fig. 3. VPL-Moodle menu and execution options.

Indeed, during the working sessions, when students were frustrated with themselves and asked any question to the researcher or the lecturer, they were rapidly able to see what was wrong and how to act over the interface.

The satisfaction factor is an important measure to motivate students to use the module, and the average is the lowest for CS2 students (M = 2.8108) and the second-lowest for CS1 students (M = 3.5152). Consequently, it is a huge problem for the VPL_moodle tool, since the evaluated satisfaction factor plays an important role in the process of learning programming and this can have consequences for the students.

Based on the results shown in Table 1, CS1 presented higher mean values for all factors in comparison to CS2. We performed statistical analysis to evaluate to what extent those differences are statistically significant. The normality distribution of the student's answers to each of the five usability factors (satisfaction, efficiency, learnability, helpfulness, and control) was checked. Only two factors in CS2 were not normally distributed and for them, we applied a Wilcoxon test to verify the difference between their medians. For the others, we applied the T-test.

The *p-value* for all the factors, except for the helpfulness factor, indicates strong evidence against the null hypothesis, which means that the differences between student's perception in CS1 and CS2 are statistically significant. As the means and the medians in all factors are higher for CS1, we could say that students from that group tend to have better impressions about the VPL-Moodle usability than students from CS2.

The difference between both courses' context is that in CS1 the programming language used is Python, and in CS2 it is C language. The C language is considered a more difficult language to learn than Python (Fangohr, 2004), and the difficulties here may be related to that rather than the VPL-Moodle interface. One could says that this is a matter of the process of learning a programming language, but the reader can perceive, further in Section 5.2, that some of these problems are shared with the lecturers, which means that there are aspects to be improved in VPL-Moodle to support students and lecturers in the teaching-learning process.

Research question 2 (RQ2) is:

RQ2. Which are the aspects of the VPL-Moodle's interface that students consider presenting problems?

Regarding the students' perception and the researcher that participated during the work sessions, it is fair enough to say that students were frustrated while performing the tasks. Such frustration can be seen in the evaluation of the satisfaction factor, which is too low for CS2 (average = 2.8108) and for CS1 (average = 3.5152). It slightly corroborates the findings of (Ribeiro *et al.*, 2014) that say that students who use VPL-Moodle are more frustrated than students who use the iVProg, a visual programming tool, "while accomplishing more complex exercises."

The second aspect we can analyze is the lack of helpfulness, efficiency, and control of the interface. In both scenarios, CS1 and CS2 classes, few students did not know how to start programming using VPL-Moodle. Some of them could not execute their codes since it gets a "randomly" error by the tool, like: "*I'm sorry, but I haven't a default action to run these type of submitted files*" or "*python: can't open file 'acen-*

tuação.py': [Errno 2] No such file or directory." It means that students did not know how to name a file. There is no help for them. The Student_1 question above in this section also reflects this absence of interface's help. Kakadiya (2020) found a similar error while evaluating the *Submitty* autograding tool[5], and he says, "*The specific violation falls under Error prevention, flexibility, and efficiency, error recovery with the severity of high*."

The efficiency aspect also received low scores. Again, the Student_2 question above in this section reflects these scores. Student_2 finished its task but s/he was unable to execute their code or to send it to be evaluated. It reflects in other aspects evaluated, such as learnability and control. In this case, Student_2 did not have control over the interface, and s/he was unable to learn how to do that by her/himself. It also corroborates the findings of (Kakadiya, 2020). Another example of how the efficiency of the interface is reduced is in the description of the task and its implementation. When a student goes to implement the code itself, the description of the task disappears. Students must open two browser tabs, one to read the description and another to implement the code.

5.2. *Descriptive Analysis of Lecturer's Answers*

We had the participation of seven lecturers. Although this sample is small, we understand that the participating professors are experts and their respective analyses reflect, in detail, and very precisely, the years of working with the VPL-Module, even though they did not follow guidelines or performed a controlled experiment.

Professors answered five questions, three objectives and two subjective. The objective questions are related to the satisfaction, efficiency, and helpfulness usability factors, receiving an average of 3.00, 3.00, and 2.86, respectively (Likert scale from 1 to 5). These results show that lecturers must invest time and effort to work with the VPL-Moodle, even though they use the interface for years.

Research question 3 (RQ3) is:

RQ3. Which are the aspects of the VPL-Moodle's interface that lecturers consider presenting problems?

In order to strongly capture the perception of the lecturers about the VPL-Moodle interface, they were asked to show their point of view through two subjective questions. The lecturers were asked to: "*Please, indicate the positive aspects of VPL's interface in terms of its usability and ease of use.*" The positive points nominated by lecturers are that the VPL-Moodle eliminates the need for programming environments and is highly configurable whether (s)he knows where its features are located.

It is clear that the satisfaction, efficiency, and helpfulness usability factors are considered problems to the VPL-Moodle interface for lecturers, having an average lower

[5] Submitty is an open source course management, assignment submission, exam and grading system. `https://submitty.org` visited in November, 3rd 2020.

than 3.0 on a 1 to 5 scale. We try to figure out where these problems are related to the interface asking, "*In your opinion, what are the aspects of VPL's interface that could be improved?*"

Creating exercises is a hard task for lecturers. They must respect a few steps, but the sequence of steps is not intuitive and it is hard to remember, moreover there are no predefined templates or help options for that type of task in VPL-Moodle. Another point mentioned was that the default settings of the execution buttons do not allow students to evaluate their code, so the teacher has a greater workload. To completely provide the task functionality to the student, the user must follow at least four additional steps. They have also emphasized that even if they were aware of the tool's configuration options, those options should be more intuitive. Currently, the location-specific settings are difficult to memorize or to find in the interface. Another point that should be improved would be the inclusion of a wizard feature to aid in the system configuration. Finally, one of the teachers suggested that those steps should be part of the menu corresponding to the creation of the activity. These findings, related to the creation of the activity, are also connected to (Kakadiya, 2020) when evaluating the Submitty tool, the author says, "*there is a lack of navigation while going through the process of creating/editing an assignment.*" The tool forces the user to navigate through buttons over a menu on top, and says, "*it can disturb the flow of the task.*" The author associated that problem with the satisfaction usability factor. Those are important limitations that need to be addressed as the preparation of the assignment requires a considerable amount of work and time that must be incorporated into lectures' time (Davuluri, 2016).

Apart from that, most of the participating lecturers believe that the translation of some standard actions, the full establishment of the activity, and the creation of test cases, should be improved considerably for the interaction interface.

5.3. *Heuristic Evaluation*

Based on the ergonomic criteria (Bastien & Scapin, 1993) and the heuristic evaluation (Nielsen & Molich, 1990), it was possible to identify numerous problems at the interface. Regarding the guidance ergonomic criteria, the VPL-Moodle presents a dialog box to "create a new file," but does not present any type of information or instruction to describe this type of operation. The titles and descriptions are objectives, but the information is not clear, they do not have data entry, description, or clearly indicated help options. For immediate feedback, the VPL-Moodle has some dialog boxes for general feedback, but it does not have a message box to assist the user in taking action to overcome possible errors. In terms of conciseness and the workload ergonomic criteria, the VPL-Moodle has short, but often non-intuitive titles, labels, and denominations. Therefore, the perception, cognitive, and motor load associated with individual outputs and inputs are not minimal. From the user control point of view and the explicit control ergonomic criteria, as the VPL-Moodle is used for actions that consume a considerable user time such as code development, the module should present control situations such as stop and go that it is possible to obtain the information of the exercises to be

performed. The VPL-Moodle has options for “undo” or “redo”, but only within the programming environment, and it does not have those features in the interaction interface. Evaluators also agreed that the VPL-Moodle displays a few error messages when some not allowed action is taken or done erroneously, but the quality of those messages is, generally, not effective because they do not present the reason or nature of the error. Another important issue in this VPL-Moodle version is that the module is not completely supported on mobile or small device screens.

6. Conclusion and Future Work

This paper presents an evaluation of the VPL’s interface available as a module for Moodle. Considering student’s perceptions of the VPL’s interface, results show that students who were studying C programming language (CS2) considered the interface with a lower low level of usability than students that were studying Python language (CS1). Both groups of students reported a low level of usability for the tool according to the adopted usability factors. The higher average of the aspects evaluated is 3.8788 (scale up to 5.0) given for the learnability aspect by CS1 class, and 3.3063 for the control aspect by CS2 class. It is interesting to note that students in CS2 class already have experience in programming, however, their perceptions about the five aspects of the VPL’s interface interaction evaluated (satisfaction, efficiency, learnability, helpfulness, and control) are considerably lower than the CS1 students.

According to the heuristic evaluation, lecturers considered VPL-Moodle an interesting tool for teaching programming, but they also did not consider the VPL-Moodle interface easy to use. In another evaluation, lecturers have complained about the learnability while creating new VPL-Moodle activities. For them, the VPL has to improve the way it creates the activity since some steps are not so intuitive and the module does not have any support or help with that. There are two main aspects that the lecturers’ considered strengths of the VPL-Moodle: the integration with a learning management system, enabling students to interact in only one place while programming, and the high possibility of configuration when lecturers know how to configure that.

For future work, we propose to improve the VPL-Moodle interface based on the insights of the heuristic evaluation, and the students and lecturer’s perception, such as for instance: to modify the menu to give clearer information, to modify the main configuration to include important steps in the very first time lecturers include an activity, to show some message boxes to help the user while using the interface, to show the activity description in the same page of the text editor, to include colors to divide the menus of the text editor and the buttons to save, execute, debug or evaluate the code, to improve the text editor with an option to change the font size and to highlight the error messages and help options.

References

Alatawi, M.N. (2019). *Examining the Impact of Learning Management Systems in Computer Programming Courses*. PhD thesis. Nova Southeastern University. Retrieved from NSUWorks, College of Engineering and Computing. (1084) https://nsuworks.nova.edu/gscis_etd/1084

Bastien, J.C. & Scapin, D.L. (1993). Ergonomic criteria for the evaluation of human-computer interfaces. *Technical Report RT-0156*. INRIA. pp. 79. inria-00070012

Bennedsen, J., Caspersen, M.E. (2006). Abstraction ability as an indicator of success for learning object-oriented programming?. *ACM SIGCSE Bulletin*, 38(2), 39–43.

Bennedsen, J., Caspersen, M.E. (2007).'Failure rates in introductory programming. *SIGCSE Bull.*, 39(2), 3236. URL: http://doi.acm.org/10.1145/1272848.1272879

Bevan, N. (1998). Iso 9241: Ergonomic requirements for office work with visual display terminals (vdts)-part 11: Guidance on usability. *TC*, 159.

Marjon Bruinsma. (2004). Motivation, cognitive processing and achievement in higher education. *Learning and Instruction*, 14(6), 549 – 568. https://doi.org/10.1016/j.learninstruc.2004.09.001

Canedo, E.D., Santos, G.A., Leite, L.L. (2018). An assessment of the teaching-learning methodologies used in the introductory programming courses at a Brazilian University. *Informatics in Education*, 17(1), 45–59. DOI: 10.15388/infedu.2018.03

Cardoso, M, Marques, R, de Castro, AV, Rocha, Á. Using Virtual Programming Lab to improve learning programming: The case of Algorithms and Programming. *Expert Systems*. 2020; e12531. https://doi.org/10.1111/exsy.12531

Caiza, J.C., Del Alamo, J.M. (2013). Programming assignments automatic grading: review of tools and implementations. In: *INTED*, pp. 5691–5700.

Chagas, D.A., Lisboa, R.P., Furtado, E.S. (2011). Framework MAAVA – Metodologia de Avaliação de Ambientes Virtuais de Aprendizagem. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, 1 (1).

Chirumamilla A., Sindre G. (2019) E-Assessment in Programming Courses: Towards a Digital Ecosystem Supporting Diverse Needs?. In: Pappas I., Mikalef P., Dwivedi Y., Jaccheri L., Krogstie J., Mäntymäki M. (Eds), *Digital Transformation for a Sustainable Society in the 21st Century. I3E 2019*. Lecture Notes in Computer Science, 11701. (pp. 585–596) Springer, Cham. https://doi.org/10.1007/978-3-030-29374-1_47

Daradoumis, T., Puig, J.M.M., Arguedas, M., Liñan, L.C. (2019). Analyzing students' perceptions to improve the design of an automated assessment tool in online distributed programming. *Computers & Education*, 128, 159–170.

Davuluri, Prathibha, Madadi, Pranavi Reddy. (2016). Moodle Java Autograder. *All Capstone Projects*, 199. http://opus.govst.edu/capstones/199

Dunican, E. (2002). Making the analogy: Alternative delivery techniques for first year programming courses. In: *Proceedings from the 14th Workshop of the Psychology of Programming Interest Group*. Brunel University, 89–99.

Fangohr, H. (2004). A comparison of C, MATLAB, and Python as teaching languages in engineering. In: *International Conference on Computational Science*, pp. 1210–1217. Springer, Berlin, Heidelberg.

Gomes, A. (2010). *Dificuldades de Aprendizagem de Programação de Computadores: Contributos Para a sua Compreensão e Resolução*. PhD thesis. Faculdade de Ciências e Tecnologia da Universidade.

Derya Gurer, M., Cetin, I., Top, E. (2019). Factors Affecting Students' Attitudes toward Computer Programming. *Informatics in Education*, 18(2), 281–296. DOI: 10.15388/infedu.2019.13

ISO (1998). *9241-11. Ergonomic Requirements for Office Work with Visual Display Terminals (VDTS)*. The international organization for standardization, 45, 9.

Jenkins, T. (2002). On the difficulty of learning to program. In: *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, 4, 53–58.

Junior, O. d. O.B., Yuska, A.P.C., Tavares, T.A. (2016). Abordagens para avaliação de softwares educativos e sua coerência com os modelos de qualidade de software. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, 27(1), 270–279.

Kakadiya, A. (2020). *Usability Enhancements on Current Autograding Tools for Introductory Computer Science Courses*. Mather's thesis, California State University, CA.

Kaunang, S.T.G., Paturusi, S.D., Usagawa, T., Mangindaan, G., Sambul, A., Sugiarso, B. (2016). Student perceptions of virtual programming lab on e-learning class at university of sam ratulangi. In: *Information & Communication Technology and Systems (ICTS), 2016 International Conference on*. IEEE, 244–248.

Keuning, H., Jeuring, J., Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education*, 19(1). https://doi.org/10.1145/3231711

Kirakowski, J., Corbett, M. (1993). Sumi: The software usability measurement inventory. *British Journal of educational technology*, 24(3), 210–212.

Kori, K., Pedaste, M., Leijen, Ä., Tõnisson, E. (2016). The role of programming experience in ICT students' learning motivation and academic achievement. *International Journal of Information and Education Technology*, 6(5), 331.

Llana, L., Martin-Martin, E., Pareja-Flores, C., Velázquez-Iturbide, J.Á. (2014). FLOP: A user-friendly system for automated program assessment. *J. UCS*, 20(9), 1304–1326.

Mathew, R., Malik, S. I., Tawafak, R. M. (2019). Teaching problem solving skills using an educational game in a computer programming course. *Informatics in Education*, 18(2), 359–373. DOI: 10.15388/infedu.2019.17

Medeiros, R.P., Ramalho, G.L., Falcao, T.P. (2019). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, 62(2), 77–90. https://doi.org/10.1109/TE.2018.2864133

Mooney, O., Patterson, V., O'Connor, M., Chantler, A. (2010). *A Study of Progression in Irish Higher Education*. Dublin: Higher Education Authority.

Mow, I. T. V. C. (2006). *The Effectiveness of a Cognitive Apprenticeship-Based Learning Environment (CABLE) in Teaching Computer Programming*, Ph.D. thesis. University of South Australia.

Nielsen, J. (1994). *Usability Engineering*, Elsevier.

Nielsen, J., Levy, J. (1994). Measuring usability: preference vs. performance. *Communications of the ACM*, 37(4), 66–75.

Nielsen, J., Molich, R. (1990). Heuristic evaluation of user interfaces. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 249256.

Porter, L., Guzdial, M., McDowell, C., Simon, B. (2013). Success in Introductory Programming: What Works? *Communications of the ACM*, 56(8), 3436. http://doi.acm.org/10.1145/2492007.2492020

Ramos, V., Wazlawick, R., Galimberti, M., Freitas, M., Mariani, A.C. (2015). A comparação da realidade mundial do ensino de programação para iniciantes com a realidade nacional: Revisão sistemática da literatura em eventos brasileiros. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE)*, 26(1), 318–327. https://www.br-ie.org/pub/index.php/sbie/article/view/5178/3566

R. da Silva Ribeiro, L. de Oliveira Brandão, T.V.M. Faria, A.A.F. Brandão, (2014) Programming web-course analysis: How to introduce computer programming? In: *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, Madrid, pp. 1–8, DOI: 10.1109/FIE.2014.7044140.

Richter, T., Rudlof, S., Adjibadji, B., Bernlöhr, H., Grüninger, C., Munz, C.-D., Stock, A., Rohde, C., Helmig, R. (2012). Viplab: a virtual programming laboratory for mathematics and engineering. *Interactive Technology and Smart Education*, 9(4), 246–262.

Rodríguez-del Pino, J.C., Rubio-Royo, E., Hernández-Figueroa, Z.J. (2012). A virtual programming lab for moodle with automatic assessment and anti-plagiarism features. In: *Proceedings of the International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE)*, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), p. 1.

Sarmento, W.W., Harriman, C.L., Rabelo, K.F., Torres, A.B. (2011). Avaliação de usabilidade no processo de desenvolvimento contínuo em ambientes virtuais de aprendizagem: um estudo de caso com o ambiente solar. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, 1(1), 781–791

Skalka, J., Drlík, M., Obonya, J. (2019, April). Automated Assessment in Learning and Teaching Programming Languages using Virtual Learning Environment. In: *2019 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. 689–697

Streiner, D. L. (2003). Starting at the beginning: an introduction to coefficient alpha and internal consistency, *Journal of personality assessment*, 80(1), 99–103.

Thiébaut, D. (2015). Automatic evaluation of computer programs using moodle's virtual programming lab (VPL) plug-in. *Journal of Computing Sciences in Colleges*, 30(6), 145–151.

Vanvinkenroye, J., Gruninger, C., Heine, C.-J., Richter, T. (2013). A quantitative analysis of a virtual programming lab. In: *Multimedia (ISM), 2013 IEEE International Symposium on*. IEEE, pp. 457–461.

Vihavainen, A., Airaksinen, J., Watson, C. (2014). A Systematic Review of Approaches for Teaching Introductory Programming and Their Influence on Success. In: *Proceedings of the Tenth Annual Conference on International Computing Education Research, ICER '14*. ACM, New York, NY, USA, pp. 1926. http://doi.acm.org/10.1145/2632320.2632349

Watson, C., Li, F.W. (2014). Failure rates in introductory programming revisited. In: *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education, ITiCSE '14*. ACM, New York, NY, USA, pp. 3944. http://doi.acm.org/10.1145/2591708.2591749

V.F.C. Ramos holds a Ph.D. in Systems and Computer Engineering at the Federal University of Rio de Janeiro (Brazil) jointly with the Eindhoven University of Technology (Holland). He is an Associate Professor in the Sciences, Technologies, and Health Center of the Federal University of Santa Catarina (Brazil). His research interests include the use of intelligent systems in education, learning analytics, social network analysis, and computer science education.

C. Cechinel received a Ph.D. degree in information and knowledge engineering from the University of Alcalá, Spain. He is currently an Associate Professor with the Sciences, Technologies and Health Center, Universidade Federal de Santa Catarina, Brazil. His research interests include the development and analysis of digital learning technologies, learning analytics, and distance learning.

L. Magé holds a bachelor degree in Information and Communication Technology at the Federal University of Santa Catarina. Her research interests include human-computer interaction and interactive technologies for education.

R.R. Lemos, a faculty member of the undergraduate and graduate programs in Information and Communication Technology at the Federal University of Santa Catarina (Brazil). Received a Ph.D. in Computer Science at the University of Calgary in 2004, Canada. His research interests include data science and visualization, interactive virtual anatomy, muscle modeling, serious games, computer graphics, human-computer interaction, and interactive technologies for education.

APPENDIX A – Users Satisfaction Questionnaire – Student

Age: _____ Sex: ( ) Female ( ) Male

Level of Expertise in VPL Module: ( ) Beginner ( ) Intermediate ( ) Advanced

Choose the answer that best fits your perception about the VPL module, where:
1. Totally disagree; 2. Disagree; 3. Not sure/Don't know; 4. Agree; 5. Totally agree.

	SENTENCES	OPTIONS				
		1	2	3	4	5
1	In general, the use of VPL-Moodle was a satisfactory experience.					
2	The time required to learn how to use the VPL-Moodle module did not prevent me from finishing the activities of the course.					
3	As a VPL-Moodle user, I feel confident to use basic commands of the module such as execute, compile and export files.					
4	The VPL-Moodle is a useful tool to complete my practical tasks in the computer science programming subject.					
5	The VPL-Moodle is consistent and does not present unexpected behavior while using the interface.					
6	It is easy to understand the interface of the VPL-Moodle.					
7	There was no need to interrupt the proposed activity in order to overcome difficulties in the use of the VPL-Moodle interface.					
8	I did not find errors related to the VPL-Moodle during the proposed tasks from the beginning to the end.					
9	During the use of the VPL-Moodle, I obtained the desired functionality by a minimum number of operations.					
10	As a VPL-Moodle user, I feel confident enough to use advanced commands and features such as debugging.					
11	The interface of the VPL-Moodle uses a terminology that is consistent with programming language courses.					
12	The information provided by the VPL-Moodle is clear enough to understand what I am being asked to do in a programming task.					
13	Regardless of how often the VPL-Moodle is used, the interface is easy to remember.					
14	The information in the help of the VPL-Moodle is sufficient.					
15	While using the VPL-Moodle, error and warning messages during compilation, implementation and evaluation are sufficient for understanding and completing the activities.					

APPENDIX B – Users Satisfaction Questionnaire – Lecturer

From questions 1 to 3, choose the answer that best fits your perception about the VPL-Moodle module:

1) I did not have difficulties using the VPL-Moodle module.

1. Totally disagree
2. Disagree
3. Not sure/Don't know
4. Agree
5. Totally agree

2) It is easy to understand the interface of the VPL-Moodle module.

1. Totally disagree
2. Disagree
3. Not sure/Don't know
4. Agree
5. Totally agree

3) The information provided by the VPL-Moodle module is clear enough to understand what to do over the interface.

1. Totally disagree
2. Disagree
3. Not sure/Don't know
4. Agree
5. Totally agree

4) Please, indicate the positive aspects of VPL-Moodle's interface in terms of its usability and ease of use.

5) In your opinion, what are the aspects of VPL-Moodle's interface that could be improved?