Building an Authentic Novice Programming Lab Environment

Natalie Culligan and Kevin Casey
natalie.culligan, kevin.casey {@mu.ie}
Faculty of Computer Science
Maynooth University
Maynooth, Co Kildare, Ireland

Abstract

As computer science becomes increasingly popular and classes become larger, there is an ever-increasing demand on course coordinators' time. As well as teaching classes, running labs, preparing exams, and providing feedback to students on their work throughout the year, course coordinators are required to keep their courses updated in order to prepare their students in a rapidly changing and evolving industry. As computer scientists, and as programmers, automation stands out as a potential solution. Automating the correction of labs and exams would free the course coordinators' time, allowing them to focus on improving the course in other ways. VPL, or Virtual Programming Lab, is a plugin for a Learning Management System, such Moodle, that provides automation of this nature, by using shell scripts to assess student code and provide automated feedback. The VPL system includes a web-based editor embedded in Moodle that students use to write their code. Our concern is that VPL does not provide a sufficiently authentic programming experience. With this in mind, we have created MULE, a browser-based desktop environment in which students can view course assignments, write, compile and run their code, while maintaining the advantages provided by VPL such as instant feedback.

Keywords

Computer Science Education, Programming, Virtual Coding Environment, Automatic Assessment, Computer Science Pedagogy, Online Programming

1. Introduction and Motivation

It has been claimed that the most effective way to teach programming to students is through practical exercises [1]. However, the increasing number of students in software engineering classes makes it harder to correct and provide feedback to these students in a timely manner. This can result in fewer practical assignments and/or less useful feedback for each student. Automated grading tools that can provide useful feedback to help the student understand any issues with their code is essential to cope with these growing numbers of students [6]. It has been reported that students do not read feedback unless the feedback is provided quickly [7]. When students can receive instant feedback on their code as they write, they are given the opportunity to assess their progress, and use the feedback given to improve their work and their understanding [10]. Feedback is one of the most powerful influences on learning and achievement [8]. VPL, or Virtual Programming Lab is a Learning Management System (such as Moodle) based system that provides instant feedback to students as they perform programming assignments. The feedback can be tailored to the assignment and to the level of the class by the course coordinator.

VPL is a scaffolded coding environment. Scaffolding in education refers to support provided to students [9], in this case through an interface. Scaffolding is useful in educational settings, but it is important that when scaffolding is removed, the student can perform the learned tasks competently without the scaffolding [13]. VPL is different from an "authentic" coding environment in many ways and our concern is that students may encounter problems when "graduating" from VPL to a traditional coding environment. This research is focused on creating a programming environment that is as authentic as possible, while also providing tools for course coordinators to provide instant feedback to their students.

2. VPL

VPL, or Virtual Programming Lab, is an auto-grading plugin for Moodle, or other SCORM compliant LMS [12]. VPL provides a simple, online development environment for writing programming assignments within the LMS. VPL has a large range of languages it supports and can modified. It has automatic and semiautomatic grading, plagiarism detection, and offers configurable features for every assignment

and allows for diverse and sophisticated ways of testing student code [5,12,14]. For example, the course coordinator has the option to define the rubric used to grade the code, make the grade visible or invisible to the student, restrict access by IP address or disable copy/paste in the VPL code editor. One of the key advantages of VPL is the opportunity to provide students with instant feedback when they are programming.

3. MULE

MULE is an online desktop-like environment that students can log into from anywhere and view their previous work and continue work on assignments, or practice coding. MULE imports or recreates the advantages of VPL and adds new features. It allows students to open windowed applications, emulating a traditional desktop. The course assignments are delivered through an application in the MULE desktop called "Workbook". From the application, the user can browse assignments to be completed, assignments already completed and all their submitted code. From any assignment page in the workbook, the user can open the code editor to write their code.

From the editor, a student can write, compile, run and evaluate their code. When a student runs their code, it does not run on their local computer. Instead we use the VPL jail server (an external server that runs a student's code and returns the output).

Every time a student makes an "attempt" on an assignment – when they save their work, or when they submit it for evaluation – the attempt is recorded by the system. We hope that this will allow cautious students to experiment with their code, without fear of doing irreparable damage to their final grade, as our system always saves the student's highest grade and, at any point in time, a student can always return to a previous attempt.

The system uses two rating systems - "Grade" and "Personal Grade". "Grade" is the grade that contributes to the students' continuous assessment and final grade. "Personal Grade" is to allow the students to go back and retry assignments they have completed, to complete them in a different way, or to try to achieve a higher score. Again, the idea behind this is that we want to encourage students to continue to use the system, rewrite code and revisit previous work after the labs have ended.

	Strongly agree	Agree	Neutral	Disagree	Strongly Disagree
Q1 VPL is intuitive to use	0.00%	72.73%	9.09%	18.18%	0.00%
Q2 MULE is intuitive to use	16.67%	58.33%	8.33%	16.67%	0.00%
	None	**A little**	**Intermediate**	**A lot**	**Too much**
Q3 Rate the level of intervention needed for MULE	0.00%	33.33%	58.33%	8.33%	0.00%
Q4 Rate the level of intervention needed for VPL	0.00%	33.33%	44.44%	22.22%	0.00%
	Definitely MULE	**MULE mostly**	**About the same**	**VPL mostly**	**Definitely VPL**
Q5 Which is quicker for novice students to get started with?	70.00%	20.00%	0.00%	10.00%	0.00%
Q6 Which do you feel students would more likely use outside lab time	30.00%	60.00%	0.00%	10.00%	0.00%
Q7 Which makes it easier to review previous work from labs	80.00%	10.00%	0.00%	10.00%	0.00%
	Definitely Browser/ MULE	**Browser /MULE mostly**	**About the same**	**Moodle/VPL mostly**	**Definitely Moodle/VPL**
Q8 Which UI do you feel is most natural for students?	36.36%	45.45%	9.09%	9.09%	0.00%

Table 2.1 – Results of Demonstrator survey

6. Conclusions

MULE is an online programming education tool that retains of the advantages of VPL and has some significant improvements. Foremost among these is that it is a realistic browser-based representation of an authentic programming environment that students will encounter later in their courses, and ultimately in industry. The browser-based nature of the tool has a compelling advantage in that there is a low barrier of entry for students – they do not need to install any special software as they would normally be required to do. In "A Quantitive Analysis of a Virtual Programming Lab" [15], the authors state that "*Students typically spend too much time to install such tools and get acquainted with them, just to be able to perform their homework assignments*". This suggests that a system that can simply be logged into from any browser can be beneficial – it removes a significant amount of the initial learning curve that may be intimidating to students and one of the contributing factors to the high dropout rate of students in introductory Computer Science courses.

7. Future Work

There is a myriad of feature requests from the first-year students and their demonstrators. This is evidence of the high degree of engagement we have had over the first semester where MULE has been used. One of the more promising features to be provided for in the near future is enhanced error message reporting. There is evidence to suggest that clearer error messages for novice programmers may improve student success. In the paper "An Exploration Of The Effects Of Enhanced Compiler Error Messages For Computer Programming Novices" [2] the use of enhanced compiler error messages was tested, and the results were examined. The results showed that the use of the Decaf editor resulted in fewer signs of struggling students in comparison to a control group, who saw standard error messages. Integrating the Enhanced Compiler Error Messages into MULE would provide an opportunity to study how differently students behave when given clearer error messages

References

[1] Bai, Xue, Ade Ola, and Somasheker Akkaladevi. (2016). "Enhancing the Learning Process in Programming Courses Through an Automated Feedback and Assignment Management System" *Issues in Information Systems* 17.3

[2] Becker, Brett A. (2015) "An exploration of the effects of enhanced compiler error messages for computer programming novices." (Master's Thesis), Dublin Institute of Technology, Dublin, Ireland

[3] Berland, M., Martin, T., Benton, T., Petrick Smith, C., & Davis, D. (2013). Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences*, *22*(4), 564-599.

[4] Blikstein, Paulo (2011). "Using learning analytics to assess students' behavior in open-ended programming tasks." Proceedings of the 1st international conference on learning analytics and knowledge. ACM.

[5] Caiza, Julio C., José María del Álamo Ramiro (2013): "Programming assignments automatic grading: review of tools and implementations." 7th International Technology, Education and Development Conference 5691-5700 Valencia, Spain.

[6] Cheang, Brenda, et al. (2003). "On automated grading of programming assignments in an academic institution." Computers & Education 41.2 121-131.

[7] Duncan, Neil (2007) "'Feed-forward': improving students' use of tutors' comments." Assessment & Evaluation in Higher Education 32.3 271-283.

[8] Getzlaf, Beverley, et al. (2009) "Effective instructor feedback: Perceptions of online Graduate students." Journal of Educators Online 6.2 n2.

[9] Jackson, S. L., Stratford, S. J., Krajcik, J. S., & Soloway, E. (1995). Model-It: A case study of learner-centered design software for supporting model building. In Proc. from the Working Conference on Applications of Technology in the Science Classroom.

[10] Kitaya, Hiroki, and Ushio Inoue (2016). "An online automated scoring system for Java programming assignments." International Journal of Information and Education Technology 6.4 275.

[11] Richter, Thomas, et al. (2012) "ViPLab: a virtual programming laboratory for mathematics and engineering." Interactive Technology and Smart Education 9.4 246-262

[12] Rodríguez-del-Pino, Juan C., Enrique Rubio-Royo, and Zenón J. Hernández-Figueroa (2012). "A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features." 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).

[13] Sharma, P., & Hannafin, M. J. (2007). Scaffolding in technology-enhanced learning environments. Interactive learning environments, 15(1), 27-46.

[14] Thiébaut, Dominique (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.

[15] Vanvinkenroye, Jan, et al. (2013) "A quantitative analysis of a virtual programming lab." Multimedia (ISM), 2013 IEEE International Symposium on. IEEE.