Real-time visual feedback: a study in coding analytics

Jeremie Seanosky, Isabelle Guillot, David Boulanger, Rébecca Guillot, Claudia Guillot, Vivekanandan Kumar, Shawn N. Fraser
School of Computing and Information Systems
Athabasca University
Athabasca, Canada
jeremie@rsdv.ca

Kinshuk
College of Information
University of North Texas
Denton, U.S.A.
kinshuk@unt.edu

Nahla Aljojo, Asmaa Munshi
Faculty of Computing and Information Technology
King Abdulaziz University
Jeddah, Saudi Arabia
naljojo@kau.edu.sa

Abstract—Higher dropout and failure rates among computer science students in introductory programming courses tend to be a norm for many institutions. Years of evidence indicate that dropouts and failures persist in spite of advancements in pedagogy, technology, and teacher training. Most advancements have relied on summative assessments and of late formative assessments. This research explores assessments computed from real-time measures, based on observational data collected during student engagement with study and remedial activities. An experiment was conducted to measure the impact of real-time code assessment and dashboard-based feedback in the domain of Programming. Results indicate better course grades for a small percentage of students, and the need for task-level and meta-level interactions to guarantee significant and persistent academic performance and programming mastery.

Keywords—coding analytics; formative; summative; interactive dashboard; performance; self-regulation

I. INTRODUCTION

Most introductory programming courses at the university level offer only summative (final) assessments [1], which may contribute to factors affecting the high rates of dropout and failure [2]. In contrast, research shows that feedback provided through formative assessments allows students to better engage in their course, regulate their learning, and overcome learning challenges [3]. Whereas formative assessment has proven to be of benefit to students, it represents a challenge for teachers in terms of workload. A coding analytics tool offering real-time assessment over students’ coding efforts with real-time feedback in a visual dashboard has the potential to address the challenge of teacher workload, while helping students improve their coding performance, and possibly reducing dropout and failure. The initial phase of such a tool was developed for the study presented in this paper, which is the first of a long-term project aiming at 1) exploring whether real-time assessment and feedback helps students succeed in their programming course and improve the mastery of their coding competence, and at 2) developing a high-quality and appealing tool that engages learners.

II. LITERATURE REVIEW

Traditional approaches to automated solutions or programming tutors providing timely formative feedback include: 1) creating test cases for every programming exercise against which students’ code is compared [4]; 2) identifying good or bad coding practices with source code patterns through machine learning techniques [5]; and 3) detecting programming error patterns where weaknesses in the student’s code are identified [6]. In addition to requiring significant involvement from teachers, these approaches have the constraint of working in pre-defined environments with only specific aspects of programming like syntax or semantics (logical error). Yet, technology exists [7] whereby students’ coding efforts are analyzed and feedback/remedial activities are offered in real-time. This paper discusses the initial phase of development of an analytics system designed to capture comprehensive aspects of programming, encompassing software design creation and comprehension, design-to-coding translation, and code testing and optimization in various programming contexts. The aim of this analytics system is to maximize these types of observational data for mapping onto a larger set of potential coding competences.

Real-time visual feedback based on continuous data collection prompting actionable facets having the potential to increase students’ academic success, the main goal of learning analytics [8], are pursued in this broader research: 1) a self-regulation module [9] based on a four-stage model [10] assisting students to a) define the task at hand, b) set and plan goals for themselves, c) enact study tactics and strategies, and d) adapt study habits; which data are tracked to extract inferences on self-regulation and explore intervention effects to improve self-regulatory capacities of students [11]; 2) well-designed visualizations prompting insightful constructive thoughts as described in experiment [12], and tracking the actions prompted by the visualizations; 3) an interactive learning analytics dashboard allowing interactions with resources [13], comprising graphic incentives promoting engagement and motivation [14]; and 4) an adaptive learning functionality optimizing and automating the learning process [15], which represents a challenge that has already been commercialized by the industry [16].

III. STUDY OVERVIEW

This study is an initial assessment of the coding analytics tool and its effect on student performance. Several iterations of that study, including one currently under preparation, will be conducted to improve the experiment design, the tool quality, its features and analysis, and the user experience with the goal of reaching the largest population of novice programmers and help them succeed in their introductory programming courses.
A. System description

The coding analytics tool used in the study comprises of three major components: 1) a code-tracking plugin in the NetBeans IDE that includes an offline buffering mechanism in case of server inaccessibility, 2) a code-processing engine for Java, and 3) feedback visualization in the form of graphs. Java code written by students in NetBeans was captured at a higher granular level, and sent to a server at real-time to be processed and analyzed against thirty-one Java competences. Most competences have been selected from the Java ontology [17]. Once analysed by a high-performance computing cluster, the results were refreshed in graphs shown in Fig. 1.

The term ‘proficiency’ in this study is defined as the students’ ability to use a programming competence based on the number of times s/he uses the concept. Thus, the proficiency is calculated only on simple ‘frequency’ data; however, other data types will be observed and calculated in future studies to measure coding proficiency. The left graph of Fig. 1 displays the cumulative level of proficiency for each competence from all coding assignments experienced by that student. Each competence is updated as new data are captured and sent to the server. Students can compare their level of proficiency with the average of the class, the top, or bottom students. The right graph of Fig. 1 offers a line graph showing the progression of any Java competence over time.

B. Participants and procedure

Participants involved in this study were 66 undergraduate female students registered in a 2nd year introductory Java programming course. The code-tracking plugin as well as the NetBeans IDE were installed in the lab computers where students completed their programming assignments. Their programming experiences were tracked throughout the semester whenever they attempted to solve assignment or exercise problems in the lab computers. In their 1st year introductory Java programming course, the students had been introduced to 24 out of the 31 Java competences assessed by the coding analytics tool. The study underwent a research ethics approval process at the participating institution.

C. Data sources and analysis

During the experiment, dashboard access logs containing the time at which a student accessed the dashboard, the web page requested, and the graph viewed were automatically collected and linked to student assignment and course grades along with the number of visits for each graph. Data were analyzed to test the extent to which frequent, real-time feedback was related to student success in their courses. Given the novel test of this system and approach, results were focused on 1) effect sizes and 2) direction of effects, and used one-tailed tests with a cut-off p-value of .10.

D. Results and discussion

1) Descriptive analysis: Results indicated that of the 66 students, 27 visited neither graphs (41%), 29 visited only the ‘Proficiency per Competence’ graph (graph 1) at least once (44%), and 10 visited both graphs (15%). All students who visited the ‘Progression of Competence over Time’ graph (graph 2) also visited the ‘Proficiency per Competence’ graph (graph 1). No student visited the ‘Progression of Competence over Time’ graph (graph 2) only. This represents an approximate 60/40 ratio of students viewing the graphs versus those who did not visit the dashboard at all.

Table I shows descriptive statistics of the grades and number of views for both graphs of the whole group of students except two outliers who got final grades of 2% and 23%. Results in Table I indicate that the course grade mean and median is around 69%. The range, standard deviation, and IQR values, all show a high level of variation among the grades. The positive skewness for Views Graph 1 and Views Graph 2 demonstrates that these distributions are positively skewed with several small values.

Grade mean (GM) per group was compared between those who: a) did not view any graph (N=27; GM=66%), b) viewed only the first graph (N=29; GM=68%), and c) visited both graphs (N=10; GM=78%). The GM difference between no views and viewing both graphs, and between viewing the first graph versus viewing both graphs was significant (both ps<.05, η² = .095 and η² = .08, respectively).

2) Correlations: The Pearson correlation between the final grade and the number of visits for the first graph is 0.26 (p < .05). This implies that the more students visit the graphs, the more they tend towards better grades. There were medium to high Pearson (0.37, p < .01) and Spearman (0.57, p < .01) correlations on the number of views between the first and second graph suggesting that the more students visit the first graph, the more they tend to visit the second graph.

3) Discussion: In this study, we found that only a small group of students who viewed real-time feedback from both graphs achieved better course grades, and that most students who did not view or only partially viewed the graphs obtained lower course grades. The study surmises that ‘seeing’ competence feedback in a dashboard is a good first step, but not enough to engage and energize students, thus ensuring their success. A more interactive approach is needed to motivate, support, and maintain students’ effort throughout their programming course.

Three major observations can be derived from the results of this study: 1) The use of the graphs of lower than anticipated. Specifically, approximately 40% of the students did not view their graphs at all, which may be linked to their perceived usefulness of the tool in terms of quality of the tool.
[18]. Participants have filled a user experience survey and the results are being analyzed in a separate paper. 2) The higher grade mean (average of 10-12% higher) for students having visited both graphs when compared to those who either visited only the first graph or did not visit the dashboard at all, as well as the correlation between the use of graphs 1 and 2, suggest that students who have monitored their progress more closely may perform better. Monitoring their overall proficiency per competence (graph 1) and their progress per competence over time (graph 2) may indicate that these students likely possessed key intrinsic factors such as grit, self-motivation, and self-regulation leading to better performance, and the dashboards further nurtured them towards improved coding competence and course performance. This is well aligned with findings from studies on self-regulation where good self-regulation habits and academic performance are positively correlated [9]. Three important aspects are worth mentioning in relation to this second observation: a) these students represent only 15% of the whole group; the goal will be reached when most students, including average at-risk students, will use and benefit from the tool with positive impact on their performance; b) the contribution of the tool in supplementing the self-motivation of this small group of students requires further study; c) these students may have had more experience or competence, and thus be more likely to consult both graphs in comparison to those less skilled, which also requires further study. 3) The fact that 44% of the students visited the first graph, but never visited the second one suggests that the system missed the opportunity to engage those students after initial visits. A better designed tool with functionalities promoting interactions should be considered to attract this type of students who may be less self-motivated, thus needing more assistance in their task-level learning process and in their meta-level regulation process.

Therefore, though it may be concluded from this study’s results that frequent, real-time feedback seems to help students succeed in their programming courses, more research is needed to substantiate that conclusion.

IV. CONCLUSION

This paper presented the setting and results of a study assessing whether frequent, real-time feedback from a coding analytics tool could help students performed better in their programming courses. Results suggest promising potential for this tool to guide and motivate students towards success. However, it also revealed limitations. A significant one is that visual feedback, without interactive functionalities to encourage students to engage in their learning process is more likely to only assist a small percentage of highly motivated students, thus missing out on a large proportion of students. Potential ways to enhance the learning analytics tool with engaging features are being explored. Future experiments should provide a better designed experiment and tool in terms of functionalities, analysis, and user experience to ensure a maximum level of usage of the tool. The more students use the tool, the better it will be possible to assess its capability in aiding students achieve higher programming performances.

ACKNOWLEDGMENT

Authors acknowledge the Deanship of Scientific Research, King Abdulaziz University, Jeddah, under grant No. (D1436 – 31- 611), and NSERC for funding this research.

REFERENCES


