Edit, Run, Error, Repeat: Learning Analytics To Identify Most Improved Programming Student

Johan Snider
johan.snider@it.uu.se
Uppsala University
Uppsala, Sweden

ABSTRACT

In the face of learning to program, students are often divided into two camps: those who excel and those who struggle. Through this challenge, some students manage to persevere. These thesis projects aims to identify the students who have made the most significant improvements in their programming skills, by leveraging various learning analytics and metrics. The field of learning analytics in programming has often sought to identify struggling students, utilizing metrics such as the error quotient (EQ) and time-on-task. By framing error quotient through operant conditioning and time-on-task through expectancy-value theory, we aim to root these learning analytics in established learning theory to validate their relevance. These preliminary results illustrate how we can identify students who have improved the most based on these metrics.

CCS CONCEPTS

• **Social and professional topics → Computer science education**.

KEYWORDS

Computer Science Education, Learning Analytics, Error Quotient, Time-on-Task

ACM Reference Format:
Johan Snider. 2023. Edit, Run, Error, Repeat: Learning Analytics To Identify Most Improved Programming Student. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (IT '23).* ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 PROBLEM AND MOTIVATION

As the age old adage goes, learning to program is difficult [3]. Often in a classroom, two groups of students emerge: students that perform well and students who struggle. Over time, effects compound and the differences between these students exaggerate. In this divide, there exists students who somehow make the transition between struggle and understanding. Unfortunately, this is not always the case. There are always students who perform well throughout a course, and vice versa, students whose struggle is constant. The interesting case, dare said most interesting, is the student who makes this leap.

Research question:

(1) **Using various learning analytics, how can we identify the student who has improved the most?**

2 BACKGROUND AND RELATED WORK

The field of learning analytics in programming often tries to identify struggling students[1, 5, 8, 11]. As such, several metrics have been developed to ascertain meaning from students programming data [2, 6]. One prominent approach is to concentrate on specific repeated errors to score a students programming sessions, commonly referred to as error quotient (EQ) [5]. Along with this, time-on-task is another metric which can be computed to reason about a student's experience while programming [7]. These projects usually aim, at least in part, to identify struggling students. Unfortunately, according to a literature review from 2015 less than 20% of learning analytics projects followed students in a longitudinal manner[4]. Additionally, from the same literature review, only 11% of the papers formally referenced learning or educational theories in the development or analysis of their project[4].

3 LEARNING THEORY

Error quotient and time-on-task each need to be explained and defined in their context. For this purpose, operant conditioning is used to explain why EQ is a valuable metric[9]. When a student encounters an error, this error can be seen as a negative stimulus. This negative stimulus should prompt a student towards some behavior to solve or remove the error. Failure to do so shows that the student has at least syntactically misunderstood some programming concept. Tallying these repeated programming errors over time gives us an indication that a student is struggling, or for lack of a better expression failing to learn.

Time-on-task can be framed with several learning theories that relate to motivation. In this work, time-on-task is framed by expectancy-value theory [9]. Here a students expectancy is their belief that they are able to finish or complete a programming exercise. The value is the benefit of successfully solving the task. Expectancy-value theory suggests that motivation is the product of the expectancy and the value. If we postulate that time-on-task is a reflection of motivation, we can expect that students with higher time-on-task have a stronger belief in their abilities and a hope for success, while students with a lower time-on-task might have a weaker belief in their own abilities.

IT '23, Spring 2024, Uppsala University Sweden

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

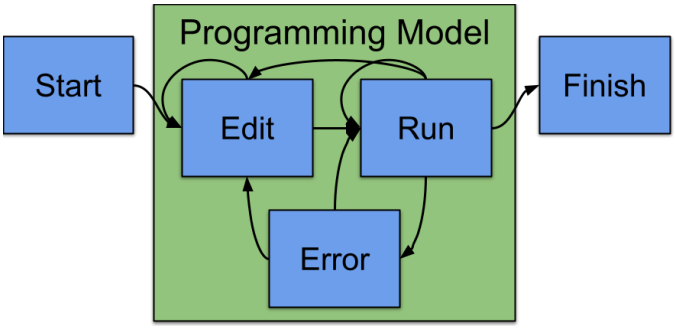

Figure 1: Programming State Diagram

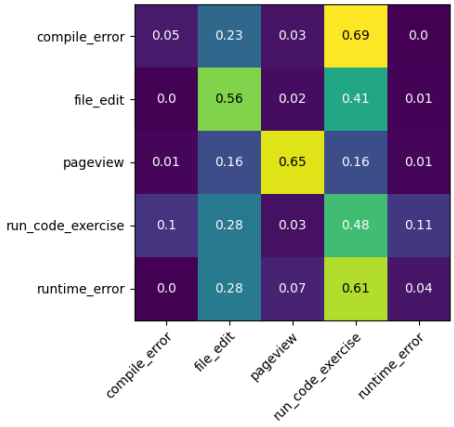

Figure 2: State Transition Matrix

4 RESULTS AND CONTRIBUTIONS

The following are preliminary results which have been, to some extent, cherry picked for illustration purposes.

Figure 1 shows a state transition diagram which can be used to reason about students programming processes [10]. While programming, students rapidly transition between these states, editing code, running their program and generating errors. By collecting this fine-grained programming data we can analyze a students programming experience in many ways.

Figure 2 shows a transition matrix for a single student during a course. Here we see that for this student, after a compile error or a runtime error, the student is more likely to run their code again, as opposed to edit their program. How these values change over time can offer insight into a student's programming behavior.

In more detail, we can observe the time in between program runs and the correctness of the program [4, 8]. Figure 3 shows how often a student runs their code, and the result if the code goes from working (Run) or not working (Error). Here we see that for this student, transitions between code with an error and code with an error happens quickly usually under ten seconds (the biggest red bar). Whereas when code goes from working to not working, usually more than 120 seconds pass (the biggest orange bar). How these behaviors change over time can also offer insight into how a student is improving in a course.

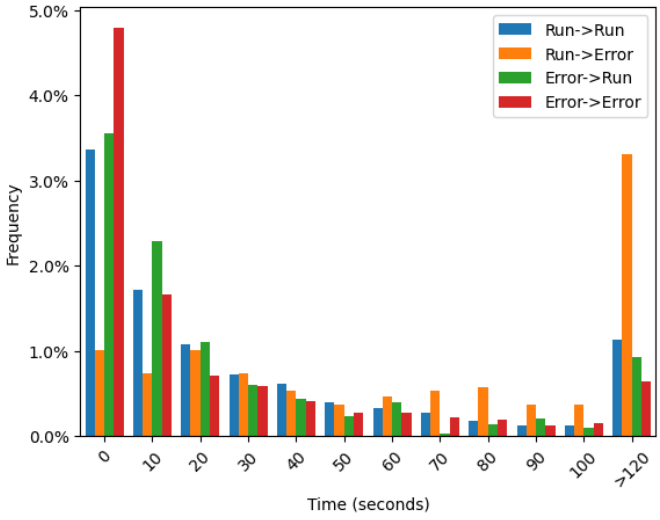

Figure 3: Time between program runs

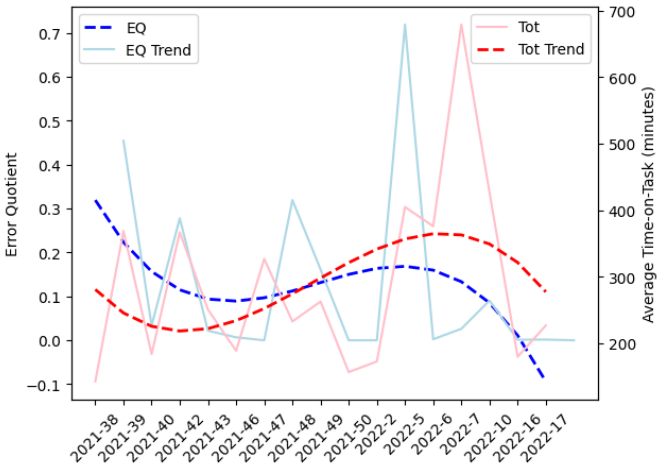

Figure 4: EQ and Tot Change Over Time

Using the fine grain programming data we can also compute an average time-on-task and an average error quotient (EQ) score for a student every week. Figure 4 shows data for one student over several months with trend lines fitted to the data for clarity. Here a break threshold of three minutes was used to compute time-on-task [6]. EQ was computed using a two parameter penalty for identical errors and errors of the same types fitted for a python context [8].

Here we see that over time the EQ score trends downward and the time-on-task trends upward, with some variation. Arguably, this is an ideal case for identifying student improvement: over time spending more time programming and having a lower EQ score. Of course, the complexity of the programming topics also factor into this analysis. As measured by these metrics we can identify the student, or students, that improve the most during a programming course.

REFERENCES

[1] Brett A Becker. 2016. A new metric to quantify repeated compiler errors for novice programmers. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. 296–301.

[2] Adam S Carter, Christopher D Hundhausen, and Olusola Adesope. 2015. The normalized programming state model: Predicting student performance in computing courses based on programming behavior. In *Proceedings of the eleventh annual international conference on international computing education research*. 141–150.

[3] Ge Gao, Samiha Marwan, and Thomas W Price. 2021. Early performance prediction using interpretable patterns in programming process data. In *Proceedings of the 52nd ACM technical symposium on computer science education*. 342–348.

[4] Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstler, Stephen H Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, et al. 2015. Educational data mining and learning analytics in programming: Literature review and case studies. *Proceedings of the 2015 ITiCSE on Working Group Reports* (2015), 41–63.

[5] Matthew C Jadud. 2006. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the second international workshop on Computing education research*. 73–84.

[6] Juho Leinonen, Francisco Enrique Vicente Castro, and Arto Hellas. 2021. Fine-grained versus coarse-grained data for estimating time-on-task in learning programming. In *Proceedings of The 14th International Conference on Educational Data Mining (EDM 2021)*. The International Educational Data Mining Society.

[7] Juho Leinonen, Francisco Enrique Vicente Castro, and Arto Hellas. 2022. Time-on-task metrics for predicting performance. *ACM Inroads* 13, 2 (2022), 42–49.

[8] Andrew Petersen, Jaime Spacco, and Arto Vihavainen. 2015. An exploration of error quotient in multiple contexts. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*. 77–86.

[9] Dale H Schunk. 2012. *Learning theories an educational perspective*. Pearson Education, Inc.

[10] Johan Mattias Snider. 2023. Edit, Run, Error, Repeat: Learning Analytics to Find Struggling Students in Upper Secondary Programming Classes. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 2*. 629–630.

[11] Christopher Watson, Frederick WB Li, and Jamie L Godwin. 2013. Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *2013 IEEE 13th international conference on advanced learning technologies*. IEEE, 319–323.