Abstraction as a Predictor of Difficulty in Quizly Problems

Beryl Hoffman
Computer Info. Tech.
Elms College
Chicopee, MA
hoffmanb@elms.edu

Ilya Ilyankou
Computer Science
Trinity College
Hartford, CT
ilyankou@gmail.com

Ralph Morelli
Computer Science
Trinity College
Hartford, CT
ralph.morelli@trincoll.edu

***Abstract*—The Mobile Computer Science Principles curriculum collects data on embedded Quizly programming exercises, which are based on the App Inventor version of Blockly. We have recently started mining this data to determine whether student performance on the programming exercises matches our assumptions about the difficulty of the individual exercises. Various analytic techniques, such as linear regression, are used to identify those features that are most determinative of problem difficulty. Our analysis supports that the number of abstractions may be a useful predictor for the difficulty (defined for our data set as the average number of attempts) in solving Quizly exercises. However, there are other not so easily quantifiable factors that also affect a problem's difficulty.**

I. Introduction

Quizly is an embeddable live coding tool based on the App Inventor version of Blockly [1]. The Mobile Computer Science Principles curriculum [2], an AP CSP high school course, collects data on Quizly programming exercises embedded in its lessons. The course engine records the number of attempts and whether the code was completed correctly for each exercise. We have recently started mining this data to determine whether student performance on the programming exercises matches our assumptions about the difficulty of the individual exercises. The goal of this analysis is to find or develop a complexity measure that can predict a problem's difficulty. We will use the analysis to help design better Quizly exercises and an appropriate sequence of Quizly exercises for future editions of the course. Determining which programming concepts are difficult could also influence how and in which order we teach these concepts.

In Quizly exercises, students solve a programming problem in an interactive environment similar to App Inventor where they can drag and drop in code blocks. Fig. 1 below shows the solution for a sample Quizly exercise that asks the student to "define a procedure called increment(X) that adds X to the global variable Z".

Students interact with the Blockly workspace to complete and test their solution until it is correct with unlimited time. In most Quizly problems, the student's solution is interpreted in JavaScript and run on a set of test data. For incorrect attempts, the student is given feedback that their solution failed on a particular set of inputs.

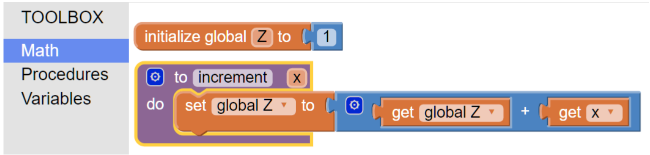

Fig. 1. Sample Quizly Exercise

In this analysis, Quizly exercises are rated with a number of features such as the number and types of blocks needed to solve the problem and the number of different programming abstractions (e.g., loops, parameters) used in the solution. Various analytic techniques, such as linear regression, are used to identify those features that are most determinative of problem difficulty.

II. The Dataset

There are two data sets from the Mobile CSP curriculum that are used in this analysis: Set 1 consists of data gathered on 15 unique Quizly exercises completed by 2,500–5,000 students between 2014-2017. Set 2 is a superset of Set 1 and consists of data gathered on 27 Quizly exercises completed by approximately 150 teachers in professional development workshops during June-July 2017. The following Fig. 2 shows attempted and completed Quizly problems, sorted by decreasing average number of attempts for Set 1 (including 3 duplicates for a total of 18 questions). As will be discussed below, the numbers on each bar represent the number of unique programming abstractions for that problem.

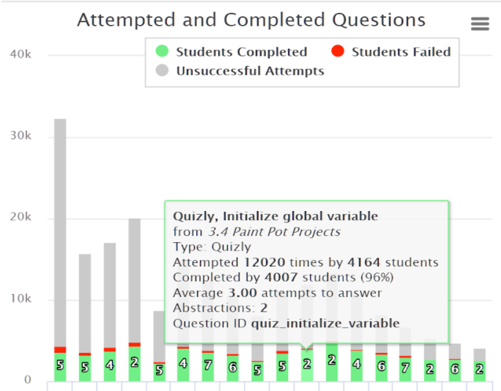

Fig. 2. Quizly Set 1 (average attempts X axis by total attempts Y axis)

This work is partially supported by the National Science Foundation under grants CNS-1240841 and CNS-1440947.

978-1-5386-2480-7/17/$31.00 ©2017 IEEE

III. Complexity Metrics

Complexity metrics have not been well studied in block languages. Code.org [3] has created a Basic Coding Proficiency matrix with 1-5 difficulty values assigned to puzzles involving programming concepts such as sequencing, conditionals, loops, event handlers, variables, and functions. In Code.org, students reach "coding proficiency" for a programming concept if they successfully create solutions using the target optimal number of blocks and no hints. In Grover et al [4][5], Scratch programming projects were labeled with an algorithmic idea such as sequence, loop, variables, conditionals, boolean logic, event handlers, etc., and researchers found that loops (especially with variables) were the most difficult programming concept to grasp.

The Quizly module in the Mobile CSP curriculum collects the number of attempts and whether the final answer is correct for each student and each exercise. We labeled each Quizly exercise with the features in the following Table I.

TABLE I. Labelled Features

Total # of blocks (total)	Number of unique blocks in the solution.
Given # of blocks (given)	Number of unique blocks given to the student from the beginning.
Abstractions (abstr)	Number of different programming abstractions, such as if, else, loops, procedures, parameters, return values, lists, global variables, local variables, setters, getters, operators.
Block Types	Number of each individual type of block counted in the number of abstractions such as loops, ifs, ifelse, procedure, procedure with parameters, global variables.

The ***number of abstractions*** is calculated as the total of the different programming concepts that were used to solve the problem, such as different types of data abstraction (global and local variables, lists), procedural abstraction (procedures or functions, parameters, return values) and the different types of control structures, operators, and setters and getters that were used. Each type of abstraction was given a weight of 1. For example, the sample Quizly question above for the procedure increment was given an abstraction value of 6 for the concepts of 1) a variable, 2) procedure, 3) parameter, 4) setter assignment, 5) + operator, 6) getter. We also investigated using a more narrow interpretation of just data and procedural abstraction, with similar results, however it can be argued that all programming constructs are abstractions in a general sense of the term.

Our measure of programming abstraction is very similar to Xie and Abelson's [6] six computational concepts: {procedure, variable, logic, loop, conditional, list}, but more extensive with 12 counted concepts distinguishing simple conditionals from more complex ones with else statements, simple procedures from procedures with parameters and return values, and different types of variables, setters, and getters.

IV. Methodology

A. Difficulty

For our analysis, we used the average number of attempts per correct solution as a ***measure of difficulty***. We initially assumed that there would be linear dependencies between some of the Quizly features listed above (the independent variables) and the average number of attempts (a dependent variable). Our hypothesis was that the number of abstractions would be a good predictor of the difficulty of the problem -- the more programming abstractions required to solve the problem, the more difficult the problem. This squares with our intuitions on problem difficulty and is consistent with the way programming concepts are introduced in the course itself.

Fig. 3 below shows the average number of attempts on Quizly problems common to both data sets, where Set 1 is labeled as *Students* with approximately 5000 students and Set 2 labeled as *Teachers* with close to 150 teachers.

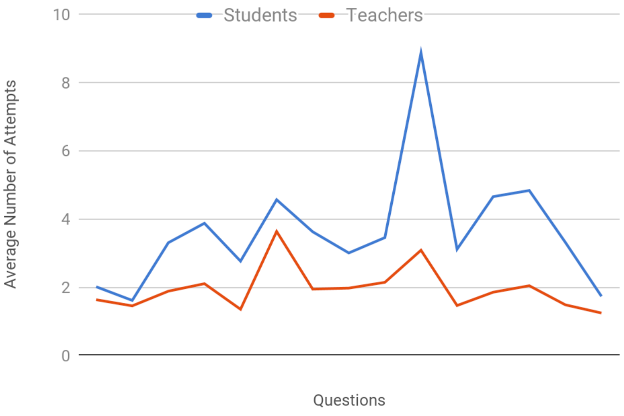

Fig. 3. Average Number of Attempts

We can see that the difficulty of each problem — as measured by the average number of attempts — is consistent across questions for both data sets. Teachers performed somewhat better than students — i.e., required fewer attempts — on the same set of exercises, which is good. But the same problems were found to be difficult for teachers and students, as seen by the corresponding spikes. The fact that the relative performance on specific problems was similar among students and teachers supports our assumption that the *attempts per correct answer* is a reasonable measure for difficulty.

B. Linear Regression Analysis

We used simple linear regression to determine which Quizly features best predict the average number of attempts. Each of 27 Quizly exercises in Set 2 (which included more difficult problems than Set 1) was represented as a 14-dimensional vector of integers, where the ith coordinate represented the ith feature. As a result, we obtained a 27x14 matrix of dependent variables. We used ordinary least squares approach as an error estimator and the scikit-learn Python library [7] to build a linear model, y = ax + b, and then to calculate the coefficient of determination (CoD) for each of the

14 features. The CoD can be represented as 1 - E where E is the regression error. Therefore, a score of 1 would be an ideal approximation.

We did not explore more sophisticated models, including multiple linear regression (with two or more dependent variables), or polynomials of higher degrees due to a limited number of data points. Building more complex models with only 27 vectors would most likely cause overfitting.

The following Table II shows 10 of the CoD values for the regression runs on Set 2. The feature names are explained in Table I above.

TABLE II. CoD Values for Features

total blocks	given blocks	**abstr**	if	ifelse
0.12	0.00	**0.47**	0.00	0.04

loop	**list**	proc	proc param	vars
0.43	**0.52**	0.02	0.02	0.05

As the table clearly shows, the presence of lists, loops, and a high number of abstractions were identified as the best predictors of difficulty as measured by average number of attempts. These three features of a problem far outperformed the other possible candidates (number of blocks, given blocks, etc.). This squares with our own intuitions about the relative difficulty of programming concepts; lists and loops are more difficult for students than assignment and if statements and a problem with lots of different interacting abstractions is even harder. It is surprising that procedural abstraction was not a big factor in this analysis, and this may be due to the limited number of Quizly problems in this area.

V. Discussion

From our analysis, it appears that a model based on the number of abstractions may be a useful predictor for the difficulty (the average number of attempts) in solving Quizly exercises. Fig. 4 shows a scatter plot of the Quizly exercises from both data sets, 15 blue dots for Set 1 and 24 red dots for Set 2. Set 2 includes all of the Quizly exercises in Set 1, but adds additional questions on loops, lists, functions, and procedures with parameters. The exercises that are shared across both sets are mostly clustered together on the left side of the graph, with low abstraction values (and no loops or functions) and correspondingly low difficulty.

For the problems common to both data sets — i.e., the easier problems — the model performed almost identically. This lends some support to the model's assumption that problems with low levels of abstraction are predicted to be less difficult.

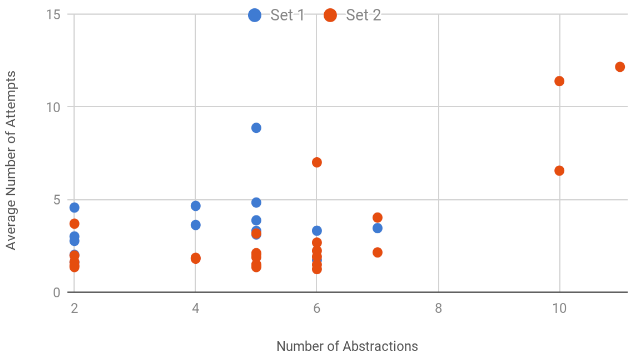

Fig. 4. Abstraction as a Predictor of Difficulty

On the right side of the graph, we see three red dots for problems from Set 2 that each have 10 or more unique abstractions. These problems all involved loops, and two of them involved traversing lists with loops. These problems with high abstraction are correlated with high difficulty, as the model should predict. This supports our hypothesis that the number of abstractions required to solve a problem may be a useful predictor of problem difficulty — with the caveat that for these more difficulty problems the model has only been tested on the relatively small data Set 2.

There are several anomalous examples that are visible in the scatter plot. For example, the red outlier at 6 required relatively many attempts for an abstraction value of 6. This may be explained by the fact that it was the very first loop problem that students encountered. Similarly, the blue outlier at 5 abstractions required relatively many attempts, but it was the first procedure definition problem encountered by students. Interestingly, the students found this problem difficult with an average of 8 attempts, while the teachers did not, with an average of 3 attempts. We also observed teachers misinterpreting problems and getting stuck on a small error due to poor feedback or a poor statement of the problem. These anomalies suggest that there are other, harder to quantify factors that affect a problem's difficulty. At the same time, they suggest the usefulness of the model in identifying problems that may need to be reworked. Table III below presents some other features that probably play a significant role in determining difficulty, such as the clarity of the problem statement, hints, feedback, and the order and similarity of the problem to previous problems.

This simple regression analysis has helped to pinpoint where students struggle and the types of mistakes they make. The higher abstraction problems involving lists and loops are difficult, and these exercises need better problem clarity, hints, and feedback. Our analysis has also led us to look at the outliers in our model to analyze why they were unnecessarily difficult. This will help us improve and develop Quizly problems with better clarity and feedback.

TABLE III. ADDITIONAL FACTORS

Statement	The clarity of the problem statement in terms of whether the student correctly interprets it.
Hints	The number and quality of the hints provided.
Feedback	The quality of the feedback provided for mistakes.
Order	The placement of the problem within a sequence of problems.
Similarity	The similarity of the problem to previous problems.

VI. Conclusions

Although this simple analysis suggests that the number of abstractions may be a useful predictor of problem difficulty, there are a number of limitations in this study. First, the model needs to be tested on a much larger set of problems and among a larger number of students, which we plan to do this coming year. Second, there are other data mining techniques that could be explored with a larger data set, such as multiple linear regression, support vector machines or neural networks. Third, properly conducting this kind of study might require a more deliberately designed set of problems; for example, placing the same or similar problem at different stages of the student's progress. Fourth, while the abstraction-based model seems to be promising, it is clearly not a sufficiently rich model for predicting difficulty. For example, assignment statements are currently given the same weight as loops when calculating the abstraction measure, but clearly loops are a more difficult concept to learn. To address this, we will experiment with models that weight the abstractions, e.g., giving loops a higher weight than assignment statements, in order to capture a more accurate measure of complexity. Finally, an analysis of this sort could be expanded to try to capture some of the harder-to-quantify elements, such as hint-giving, problem statement, feedback and clarity. This sort of extension would likely require building additional instrumentation into the course engine.

It seems reasonable that a problem's difficulty depends in large part on the number of different programming abstractions that it involves -- if all other things are equal. This simple study suggests that such a model is not only useful as a predictor of problem difficulty but also may help, through outliers, in identifying poorly structured problems. In future work, we will use the results of this study to identify and fix issues with some of the existing problems and develop new Quizly exercises with varying degrees of abstraction. In addition, knowing which Quizly exercises were difficult for students will lead us to provide more instruction and scaffolding on those concepts.

References

[1] F. Mairorana, D. Giordano, R. Morelli. “Quizly: a live coding assessment platform for App Inventor”. IEEE Blocks and Beyond Workshop. 2015.

[2] Mobile Computer Science Principles Curriculum. http://mobilecsp.org.

[3] Code.org. “Basic coding proficiency matrix”. https://code.org/about/evaluation/proficiency. 2016.

[4] S. Grover and S. Basu. “Measuring student learning in introductory block-based programming: examining misconceptions of loops, variables, and boolean logic”. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. ACM, 2017.

[5] S. Grover, R. Pea and S. Cooper. “Factors influencing computer science learning in middle school”. Proceedings of the 2016 ACM SIGCSE Technical Symposium on Computer Science Education. ACM, 2016.

[6] Xie, B., and H. Abelson. "Skill progression in MIT App Inventor." Visual Languages and Human-Centric Computing (VL/HCC), 2016 IEEE Symposium on. IEEE, 2016.

[7] Scikit-learn Machine Learning in Python Library. http://scikit-learn.org.