Using process and motivation data to predict the quality with which preservice teachers debugged higher and lower-complexity programs

Brian R. Belland, ChanMin Kim, Anna Y. Zhang, Afaf Baabdullah, and Eunseo Lee

Abstract - **Contribution: This study indicates that supporting debugging processes is a strong method to improve debugging outcome quality among preservice, early childhood education (ECE) teachers.**
Background: Key to preparing ECE teachers to teach computer science is helping them learn to debug. Little is known about how ECE teachers' motivation and debugging process quality contributes to debugging outcome quality.
Research Questions: How do debugging process and motivation variables predict the quality with which participants debug lower- and higher-complexity programs?
Method: A Bayesian multiple linear regression model with debugging process and motivation variables as predictors was used to predict debugging outcome quality. An inverse gamma prior distribution for sigma^2 and uniform prior distribution for Betas was used.
Findings: The strongest positive predictor of debugging outcome quality for both the lower-complexity and higher-complexity debugging task was debugging process quality.

Index terms **–computing education; teacher learning; regression analysis; software debugging**

I. Introduction

Within early childhood education (ECE), computer science (CS) is often taught by inviting children to control robots using block-based programming [1], [2]. Within block-based programming, bugs can occur, and resolving such can be challenging [3]. In this study, preservice, ECE teachers used debugging scaffolding. Motivational and process variables were used to predict the quality with which they completed lower- and higher complexity debugging tasks.

In the next sections, background literature is described, research questions are articulated, method is described, and results are presented and discussed, including a statement of limitations and conclusions.

II. Background

A. Debugging

In this study, we focused on debugging of preservice ECE teachers because of its importance in programming and also its relation to motivation to learn. Debugging is the process of identifying faulty code or logic usage, determining how to fix it, and fixing it [4], [5]. Debugging skill is central to programming success not only in text-based programming but also in block-based programming [3]. Still, challenges with debugging cause frustration at best, and dropping out of CS pathways at worst [4], [6]. Novice programmers often examine isolated sections of code for bugs, attempt to fix what they find, and fail to consider how that section of code interacts with remaining code [7], [8]. Unskillful debugging often creates new bugs, which deepens difficulties with debugging because new bugs are often unnoticed [4], [9]. Debugging skills also have a close relation with the programmer's self-efficacy [6]. As such, the need for programming instruction with an emphasis on debugging has received increasing attention in CS education [3], [10].

B. Motivation

To get a more complete picture of preservice ECE teachers' debugging process and outcomes, their motivational variables, including goal orientations [11] [12], interests [16], and stereotypical conceptions [14], [15], need to be understood. Achievement goal orientation refers to what an individual wishes to accomplish within a learning task. Individuals with mastery goals aim to achieve mastery of content, individuals with performance-approach goals aim to demonstrate competence, and individuals with performance-avoid goals avoid challenging tasks to avoid appearing incompetent [11], [12]. While performance-avoid goal orientations are almost always maladaptive, performance-approach goal orientations can be adaptive when faced with high challenge tasks [13]. Lack of domain identification is often associated with stereotype threat, in which knowledge of a stereotype (e.g., CS jobs are not seen as feminine), causes participants to perform worse in a domain-relevant task than their capability would predict [14], [15]. Stereotype threat impacts one's motivation to pursue CS careers [16].

C. Scaffolding

As part of our effort in educating preservice ECE teachers,

This Manuscript received July 22, 2020; revised December 11, 2020 and January 14, 2021; accepted January 29, 2021. This work was supported by the National Science Foundation (USA) under grants 1906059 and 1927595. (Corresponding author: Brian R. Belland).

B. R. Belland (email: bbelland@psu.edu), A. Y. Zhang, and E. Lee are with the Department of Educational Psychology, Counseling, and Special Education, Pennsylvania State University (Penn State), University Park, PA 16802 USA

C. Kim is with the Departments of Learning Performance Systems and Educational Psychology, Counseling, and Special Education, Penn State and A. Baabdullah is with the Department of Learning Performance Systems, Penn State, and the Department of Curriculum and Instruction, King Saud University, Riyadh, Saudi Arabia

we used debugging scaffolding in this study. Scaffolding supports learners as they engage in problem solving by problematizing task elements essential to gaining problem solving skill, while simplifying unessential elements [17], [18]. Scaffolding forms include question prompts to which learners respond, and experts modeling how they would think about a similar problem [19], [20]. Ill-structured problem-solving ability is enhanced more when learners receive scaffolding than when they receive lecture, with the largest effects at the university, graduate, and adult levels [21], [22].

Analysis of scaffolding responses can be used to capture not only debugging knowledge and process, but also attitudes toward debugging. For example, word count can be used as a proxy for response quality, because respondents with good programming knowledge and mindful thinking are likely to generate more words to answer prompts than respondents without. Several studies have indicated word count to be a strong predictor of domain knowledge [23], [24]. Sentiment analysis has been used to monitor learners' attitudes toward learning and engagement/performance in learning [25], [26].

The current study is innovative because it uses only data that can be collected before and during debugging to predict debugging outcome quality, thereby providing the opportunity to remediate during real time. No prior, published study predicted debugging outcome quality using motivation variables, sentiment, or word count. Existing work uses variables such as prior performance, hint usage, activity progress, interface interactions, and binary classification of passing or failing [27]. This study employs a numeric scale to address how participants engage in different debugging stages and substages (see Table I in the method section).

III. Research Questions

1. How do process and motivation variables predict the quality with which participants debug low complexity programs?
2. How do process and motivation variables predict the quality with which participants debug high complexity programs?

IV. Method

A. Setting and Participants

The study took place during 3 sessions of 2.5 h each of a class on play-based activities in ECE in a large university in the eastern USA. Nineteen students (all female) participated. Eighteen were ECE majors (4 seniors, 10 juniors, and 4 sophomores) who had completed field experience at the infant/toddler, preschool, kindergarten, or early elementary levels. The other was a non-major who had not completed field experience. Two participants (10%) were Black, while the rest were white. That participants were all female aligns with demographics in the ECE and elementary workforces, of which 97.4% [28] and 89% [29], respectively, are women. Also, most (78%) ECE teachers in the USA are white [30].

B. Materials

1) Ozobots

Ozobot Evos are small robots that can sense colors, lines, and obstacles and use such information for navigation and in logic. Ozobots were chosen because they are simple enough for preschool, while still involving coding (Ozoblockly) [31].

2) Scaffolding

Scaffolding design was informed by performing (a) a two-step cluster analysis of an expanded coding dataset from a meta-analysis [21], (b) a literature review on debugging education, and (c) a synthesis of Kim et al.'s scaffolding recommendations for debugging in block-based programming [3]. The cluster analysis generated a list of scaffolding features of clusters where scaffolding led to medium or high effect sizes in university or graduate-level CS or engineering classes. The scaffolding invited participants to: (a) identify blocks (e.g., movement) used within the buggy code, (b) identify the order in which blocks were used, (c) create hypotheses about the bug that caused the unexpected robot behavior, and articulate reasons for each hypothesis, (d) articulate changes made to test each hypothesis, why the changes were made, what happened after the changes were made, and the current sequence of blocks, and (e) reflect on the process.

3) Debugging tasks

Complexity of debugging tasks is defined by number of tasks/goals and sub tasks/goals, the degree of connectivity among bugs, and the clarity of the presented goal state [32].

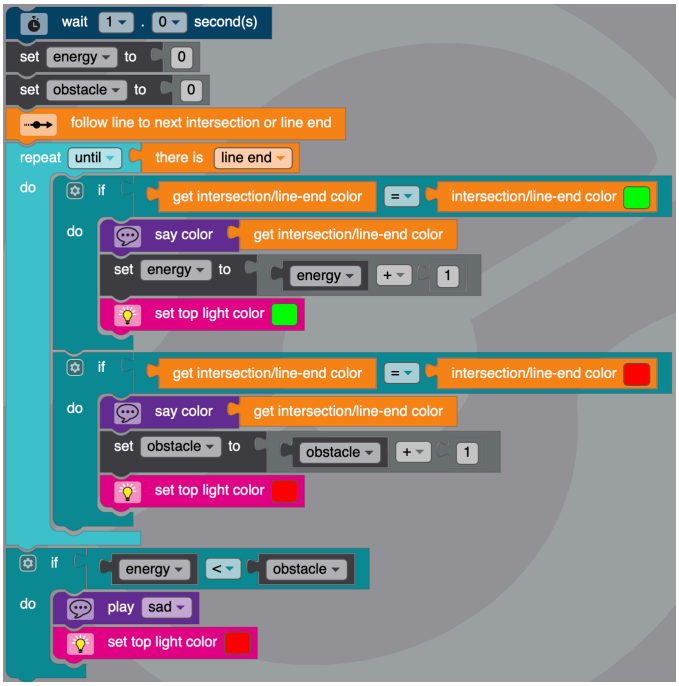

Fig. 1. Higher-complexity buggy code

The higher-complexity task (see Fig. 1) included the following CS concepts: variables, logic, loops and elementary operations. In it, participants needed to debug a program in which an Ozobot used line navigation to follow a grid with red and green intersections. The program used equations to (a) add one point to the Energy score and play a happy sound when the Ozobot passed through a green intersection, and (b) add one point to the Obstacle score and play a sad sound when it passed through a red intersection. The Ozobot was supposed to move along the grid as long as obstacle points were less than or equal to energy points. But one bug did not allow the Ozobot to make it out of a small area of the grid, and the other

bug caused the Ozobot to never stop moving even when Obstacle points are greater than Energy points. To fix the program, participants needed to examine Ozobot movement under four conditions: when the Ozobot passes a green intersection, when the Ozobot passes a red intersection, when energy points are greater than or equal to obstacle points, and when energy points are less than obstacle points. Another factor that made this task complex was the connectivity among bugs. One bug hindered observing the effect of fixing the other bug; participants needed to fix one bug before the other to judge the appropriateness of debugging.

The lower-complexity task (see Fig. 2) included the CS concepts of variables, logic, loops and geometry. Participants needed to debug a program that instructed the Ozobot to make an octagon pattern. Bugs caused the Ozobot to not turn the correct angle and not turn enough times. The goal state was clear because participants were given a printed octagon that they needed to make the Ozobot trace. Participants needed to fix values within two variables: angle (rotate block) and the number of repetitions (repeat while block). These bugs were not isolated from each other; both had to be fixed to perform the desired behavior. However, without fixing one, the effect of the other bug was easily observable. For example, without changing 5 to 8 in the repeat while block, the robot turns were noticeably misaligned with the octagon pattern.

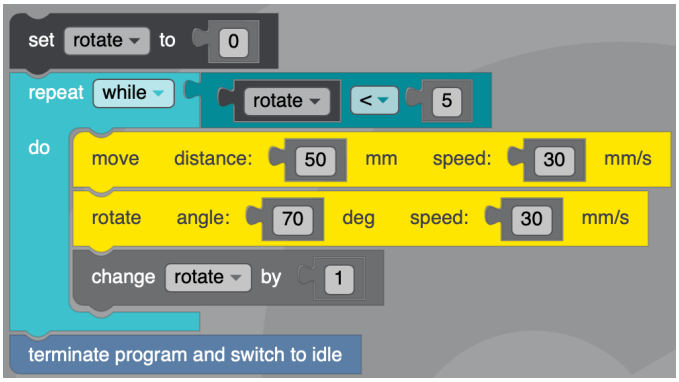

Fig. 2. Lower-complexity buggy code.

C. Data Collection

1) Presurvey

The STEM interest survey section contained 30 items using a 7-point Likert scale to measure interest in STEM content and careers [33], [34]. A prior study using this scale reported good reliability with Cronbach's $\alpha = 0.89 - 0.91$ [34]. A modified version of the a questionnaire on academic emotions in mathematics [35], contained 7 items using a 5-point Likert scale (e.g., When doing my STEM-related homework, I am in a good mood.). A recent study used the scale, reporting good reliability ($\alpha = .90$) [36]. The Learning Self-Regulation Questionnaire (LSRQ) section contained 6 items with 5-point Likert scale that assessed participants' reasons for engaging in STEM learning behaviors (e.g., I would feel bad about myself if I didn't do STEM-related class activities) [37]. LSRQ showed good reliability ($\alpha = .83$) in a prior study [38]. The Domain Identification Measure contained 16 items on a 5-point Likert scale that assesses identification with Mathematics and English (e.g., I do badly in tests of math) [39], and 8 items on a 5-point Likert scale that assesses perceptions of CS and engineering (e.g., Computer science is stressful). Prior studies indicate that it exhibited good reliability ($\alpha = 0.9$-0.93) [39], [40]. The Patterns of Adaptive Learning Section contained 14 items on a 5-point Likert scale that measure personal goal orientations (e.g., One of my goals is to show others that I'm good at my class work); good reliability ($\alpha = 0.7$-0.89) for the scale was found in prior studies [41]. The Views of Coding section assessed attitudes toward coding using 19 questions on a 5-point Likert scale (e.g., I think that coding is interesting.) [42].

Mean value imputation [43] was used to fill in missing data (about 1% of presurvey data). Mean value imputation is appropriate, and accuracy of estimates using imputed data is likely good, because less than 5% of data was missing [44].

2) Debugging process quality rubric

Quality of debugging processes was assessed using a rubric to evaluate participants' processes of identifying, locating, and fixing bugs (see Table I). Identifying bugs included the subprocesses of *reviewing program*, *running program to evaluate output*, *describing discrepancy between program goal and program output*, *and generating hypothesis for a bug causing the discrepancy*. Locating bugs included the subprocesses of *reviewing program structure to investigate probable location of bug, using cues while examining program,* and *locating bug*. Fixing bugs included the subprocesses of *examining program goal to determine how to fix bug*, *revising program, reevaluating program after revision*, and *concluding*. Each subprocess was evaluated on a scale from 1 (poor quality) to 4 (high quality).

Two raters applied the rubric independently to each group's scaffold responses. Before coming to consensus, interrater reliability was 0.954, based on the intraclass correlation coefficient metric. The raters met to discuss and reach consensus on scores. A unique debugging process quality score was assigned to each team's debugging of the lower-complexity task and the higher-complexity task.

3) Debugging outcome quality rubric

The debugging outcome quality rubric (see Table II) assessed whether participants found the bug and fixed it. Two raters applied the rubric independently to assess the final code for each debugging task, and then came to consensus. Interrater reliability before coming to consensus was 0.88, as measured by the intraclass correlation coefficient.

4) Word count

A unique word count was assigned to each team's scaffold responses and debugging reflections of the lower-complexity and higher-complexity debugging tasks. The word counts were then averaged to arrive at 2 unique word counts for each team: (a) word count of scaffold responses during the lower-complexity task, and reflections on such, and (b) word count of scaffold responses during the higher-complexity task, and reflections on such. Missing data was not included in the calculations of averages in the absence of evidence that the student saw the writing prompt and failed to write anything.

5) Sentiment analysis

The SentimentAnalysis package for R [45] was used to determine the valence of scaffolding responses and debugging process reflections. Analysis was at the level of each entire response to a prompt or reflection question [46]. Possible

scores ranged from -1 to 1. Scores were averaged to arrive at 2 unique sentiment analysis scores for each team: (a) sentiment analysis of scaffold responses while debugging the lower-complexity task, and reflections on such, and (b) sentiment analysis of scaffold responses while debugging the higher-complexity task, and reflections on such.

D. Analysis

Bayesian multiple linear regression was used to predict the quality with which participants debugged the higher- and lower-complexity debugging tasks because, with small samples, frequentist methods often lack precision due to limitations of the central limit theorem [47]. In the Bayesian paradigm, a probability model is set up using a prior distribution and then updated using observed data to obtain the posterior distribution; the end result is a credible interval that represents a distribution of the parameter [48], [49]. The prior distribution can be updated through Markov Chain Monte Carlo (MCMC) sampling, which samples from the population distribution [50]. This is because in the Bayesian approach, one assumes that data are fixed and parameters are random [47]. It is easier to work with and understand a regression model with a limited number of predictors [51]. In addition, too many variables in the model may increase the sampling variance of model coefficients and decrease the model's descriptive and predictive abilities [51]. The stepwise model selection algorithm in Bolstad R package [52] was used to extract the potentially useful subset of explanatory variables.

TABLE I
DEBUGGING PROCESS QUALITY RUBRIC

	Points assigned			
	1	2	3	4
Identifying bug				
Reviewing code	Did not review code	NA	Reviewed code; Block list incomplete	Reviewed code; Block list complete
Running program to evaluate output	Did not run program	NA	NA	Ran program
Describing discrepancy between program goal and output	Did not describe discrepancy	Described discrepancy incorrectly	Described discrepancy correctly yet vaguely	Described discrepancy correctly and clearly
Generating hypothesis for a bug causing the discrepancy	Did not generate a hypothesis	Generated hypothesis that was unrelated to discrepancy	Generated hypothesis that was partially related to discrepancy	Generated hypothesis that was fully related to discrepancy
Locating bug				
Reviewing program structure to investigate probable location of bug	No attempt to locate bug	Line-by-line review of program; no review of program structure	Chunk-by-chunk review of program; no review of program structure	Thoroughly reviewed program and program structure
Using cues while examining program	No cue was mentioned	Cues mentioned; cues unrelated to bug	Cues mentioned; cues partially related to bug	Cues mentioned; cues related to bug
Locating bug	No attempt to locate bug	Attempted but failed to locate bug	Bug was partially located	Bug was correctly located
Fixing bug				
Examining program goal to determine how to fix bug	Did not mention program goal	NA	NA	Mentioned program goal
Revising program	No attempt to revise program	Revised program incorrectly	NA	Revised program correctly
Reevaluating program after revision	No attempt to reevaluate	Failed numerous trial-and-error attempts	Success after numerous trial-and-error attempts	Success without trial-and-error attempt
Concluding	Bug was not fixed	Bug fixed but not in structurally correct manner; Desired output produced	Bug fixed in structurally correct manner; Desired output not produced	Bug fixed in structurally correct manner; Desired output produced

TABLE II
DEBUGGING OUTCOME QUALITY RUBRIC

0	3	5	7	10
Failed to find buggy block	Found buggy block but not exact location requiring change(s) in value and/or sequence; Failed to fix	Found buggy block and exact location requiring change(s) in value and/or sequence but (b) failed to fix	Found buggy block and exact location requiring change(s) in value and/or sequence; Fixed bug partially	Fixed bug

Note: Two points were deducted (-2) for each unnecessary code change

Prior distributions of inverse gamma for sigma^2 and uniform prior for Beta values were used. MCMC sampling (11,000 iterations with 1,000 burn in) was run in MCMCpack R package [53] to determine the posterior distribution.

V. RESULTS

A. Descriptives of Significant Variables within Each Model

See Tables III and IV for descriptive statistics of the significant predictors for the models predicting debugging outcome quality of the lower-complexity task and debugging outcome quality of the higher-complexity task. Debugging process quality was substantially higher on average (Cohen's $d = 4.23$) and more tightly clustered around the mean in the lower-complexity task ($SD = 2.41$) than in the higher-complexity task ($SD = 7.55$). Word count was slightly higher in the higher-complexity debugging task than in the lower-complexity debugging task (Cohen's $d = 0.3$), but its standard deviation was two times higher in the higher-complexity debugging task versus the lower-complexity debugging task.

TABLE III
DESCRIPTIVE STATISTICS: SIGNIFICANT PREDICTORS OF QUALITY WITH WHICH PARTICIPANTS DEBUGGED THE LOWER-COMPLEXITY TASK

Variable	Mean	SD	Min	Max	SE
Debugging process quality	82.56	2.41	80	88	$8.48e^{-3}$
Performance-avoid goal orientation	10.74	3	5	16	$4.78e^{-3}$
Word count	282.2	100.34	157	481	$4.81e^{-6}$
Sentiment analysis	0.09	0.05	0.03	0.19	$2.001e^{1}$

TABLE IV
DESCRIPTIVE STATISTICS: SIGNIFICANT PREDICTORS OF QUALITY WITH WHICH PARTICIPANTS DEBUGGED THE HIGHER-COMPLEXITY TASK

Variable	Mean	SD	Min	Max	SE
Debugging process quality	58.85	7.55	45	71	$1.1e^{-2}$
Performance approach goal orientation	11.32	3.59	5	18	$4.44e^{-2}$
Word count	331	207.73	146	807	$1.58e^{-5}$

Debugging outcome quality scores were substantially higher (Cohen's $d = 2.53$) and scores were more tightly clustered around the mean in the lower-complexity task ($SD = 1.3$) than in the higher-complexity task ($SD = 4.42$; see Table V). This is not surprising in that defining the problem and goal state required more effort for the higher-complexity task.

TABLE V
DESCRIPTIVE STATISTICS: DEBUGGING OUTCOME QUALITY

Variable	Mean	SD	Min	Max	SE
Lower-complexity task debugging outcome quality	18.56	1.3	16	20	$3.9e^{-2}$
Higher-complexity task debugging outcome quality	10.31	4.4	-1	17	$4.92e^{-1}$

B. Convergence Diagnosis

Trace plots showed sufficient state change, histograms and density plots showed no problems in normality in the simulated samples, and autocorrelation plots showed no issue.

C. How do process and motivation variables predict the quality with which participants debug low complexity programs?

The maximum number of predictors a Bayesian regression model can take with sample size $n = 19$ is 12. Thus, we needed to remove a few predictors before fitting the full model. We tried to fit multiple initial full models and went through the selection procedure multiple times to decide which predictors to remove. From the pool of potential predictors ($n = 18$), we removed seven: achievement emotions in STEM, self-regulated learning ability, views of coding, technology interest, perceptions of STEM careers, engineering interest, and mathematics domain identification, because these predictors either had large (>= 4) variance inflation factors, or were not significant. The remaining predictors were: (1) science interest, (2) mathematics interest, (3) computer science interest, (4) English domain identification, (5) computer science and engineering domain identification. (6) mastery goal orientation, (7) performance-approach goal orientation, (8) performance-avoid goal orientation, (9) debugging process quality (lower-complexity debugging task), (10) sentiment analysis (lower-complexity debugging task), and (11) word count (lower-complexity debugging task). We then used the step() function (using both backward and forward direction and AIC criterion) to decide on final predictors. AIC penalizes models with more predictors and if two models explain the same amount of variation, the model with fewer predictors is preferred. AIC removed the following seven predictors from the model: science interest, mathematics interest, computer science interest, English domain identification, computer science and engineering domain identification, performance approach goal orientation, and mastery goal orientation. The final model (See Table VI) was: LCDOQ = 0.3 * LCDPQ - 0.19 * PAvGO - 0.003 *LCWC - 9.3 * LCSA - 2.97.

The Beta for lower-complexity task debugging process score was 0.3. For each one-point increase in lower-complexity task debugging process quality, one can expect an increase in lower-complexity task debugging outcome quality score of 0.3 points. The Beta for performance-avoid goal orientation was -0.19. For each one-point increase in performance-avoid goal orientation, one can expect a decrease in lower-complexity task debugging outcome quality of 0.19 points. The Beta for lower-complexity debugging task word count was -0.003. For each additional word typed in the lower-complexity task, one can expect a decrease in lower-complexity debugging outcome quality of 0.003 points. The Beta for lower-complexity task sentiment analysis score was -9.3. For each one-point increase in lower-complexity task sentiment analysis, one can expect a decrease in lower-complexity task debugging outcome quality of 9.3 points.

TABLE VI
BAYESIAN MULTIPLE REGRESSION MODEL PREDICTING LOWER-COMPLEXITY TASK DEBUGGING OUTCOME QUALITY (LCDOQ)

Coefficients	Estimate (95% CrI)	Naïve SE
(Intercept)	-2.97 (-19.66, 14.17)	$8.47e^{-2}$
Lower-complexity debugging process quality (LCDPQ)	0.3 (0.1, 0.5)	$9.91e^{-4}$
Performance-avoid goal orientation (PAvGO)	-0.19 (-0.34, 0.49)	$7.53e^{-4}$
Lower-complexity Word count (LCWC)	-0.003 (-0.007, 0.002)	$2.38e^{-5}$
Lower-complexity sentiment analysis (LCSA)	-9.3 (-18.85, 0.34)	$4.82e^{-2}$

Note. MCMC iterations = 10,000; 95% CrI = 95% credible interval

D. How do process and motivation variables predict the quality with which participants debug high complexity programs?

From the pool of potential predictors (n = 18), we removed seven: engineering interest, computer science interest, science interest, achievement emotions in STEM, self-regulated learning, mathematics domain identification, and views of coding because these predictors either had large (>= 4) variance inflation factors, or were not significant. The remaining predictors were: (1) performance-approach goal orientation, (2) performance-avoid goal orientation, (3) mastery goal orientation, (4) mathematics interest, (5) technology interest, (6) STEM career interest, (7) English domain identification, (8) computer science and engineering domain identification, (9) debugging process quality (higher-complexity debugging task), (10) word count (higher-complexity debugging task), and (11) sentiment analysis (higher-complexity debugging task). Then we fitted the full model and used the AIC criterion to remove the following eight variables: higher complexity sentiment analysis, mathematics interest, technology interest, STEM career interest, English domain identification, computer science and engineering domain identification, performance-avoid goal orientation, and mastery goal orientation. The final model (See Table VII) was: HCDOQ = 0.45 * HCDPQ + 0.41 * PApGO - 0.005 * HCWC - 19.17.

TABLE VII
BAYESIAN MULTIPLE REGRESSION MODEL PREDICTING DEBUGGING OUTCOME QUALITY FOR THE HIGHER-COMPLEXITY DEBUGGING TASK (HCDOQ)

Coefficients	Estimate (95% CrI)	Naïve SE
(Intercept)	-19.17 (-32.75, -5.18)	$6.96e^{-2}$
Higher-complexity debugging process quality (HCDPQ)	0.45 (0.22, 0.67)	$1.14e^{-3}$
Performance-approach goal orientation (PApGO)	0.41 (-0.05, 0.86)	$2.28e^{-3}$
Word count (HCWC)	-0.005 (-0.014, 0.003)	$4.28e^{-5}$

Note. MCMC iterations = 10,000; 95% CrI = 95% credible interval

The Beta for higher-complexity task debugging process quality was 0.45. For each one-point increase in higher-complexity task debugging process quality, one can expect an increase in higher-complexity task debugging outcome quality of 0.45 points. The Beta for performance-approach goal orientation was 0.41. For each one-point increase in performance-approach goal orientation, one can expect a decrease of 0.41 points in higher-complexity task debugging outcome quality. The Beta for lower-complexity task word count was -0.005. For each additional word typed in the higher-complexity task, one can expect a decrease in higher-complexity task debugging outcome quality of 0.005 points.

VI. Discussion

A. Similarities in Predictors of Debugging Quality

In this study, the strongest positive predictor of debugging quality for both the lower-complexity and higher-complexity debugging tasks was debugging process quality. This indicates that well-designed scaffolding has great potential to improve debugging quality among ECE teachers. Quality of problem-solving processes has consistently been shown to be a positive predictor of problem solving quality [54], [55]. And yet, studies that predict success in introductory computing courses often identify different top predictors: predicted grade in the class [56], comfort level [57], and total debugging time [58].

Word count for the higher- and lower-complexity tasks had similar Betas (-0.003 and -0.005, respectively). Thus, writing more or less in response to scaffolding prompts did not change debugging quality much, which goes against much scaffolding literature [23], [24]. One may expect that as more is articulated in response to scaffolding, learning would increase. If more changes were made and documented, word count may have been higher and debugging quality lower compared to students who documented less changes (in the scaffolding, they needed to list changes they made, explain why they made each change, and report the consequence of each change). But this could have been counter-balanced by the typical finding that more written in response to scaffolding indicates higher achievement, resulting in the Betas close to zero.

B. Differences in Predictors of Debugging Quality

While participants wrote more words on average ($ES = 0.3$) in response to scaffolding when engaging with the higher-complexity debugging task, the quality with which they debugged the higher-complexity debugging task was substantially lower ($ES = 4.23$) than the quality with which they debugged the lower-complexity task. Participants may have engaged more deeply with the scaffolding when they were engaging with a demonstrably more difficult debugging task. But this did not translate to stronger debugging performance. It is thus critical to carefully unpack predictors of debugging outcome quality to understand what led to stronger and weaker debugging outcome quality.

That performance-approach goal orientation positively predicted higher-complexity debugging outcome quality but did not predict lower-complexity debugging quality makes sense in that performance approach goals are associated with strong performance in high challenge contexts, but not in low-challenge contexts [13]. But performance approach goal orientation has also been associated with help avoidance and anxiety [59]. Avoiding help can lead to ignoring scaffolding [60] and procrastination when self-efficacy is low [61].

Learners with performance-avoid goal orientations often refrain from engaging in activity so as to avoid appearing less competent than peers [62]. So it makes sense that having a performance-avoid goal orientation is a negative predictor of lower-complexity debugging quality. But yet it is not so for the higher-complexity debugging task. One possibility is that participants who were higher in performance-avoid goal orientation simply asked the instructor for hints or the answer, rather than engage deeply with the higher-complexity debugging task. Future research may add an observation checklist to quantify the number of times participants ask the instructor for hints or the answer.

Intriguingly, sentiment analysis was associated with a large negative Beta for the lower-complexity debugging task. This was the strongest predictor from among the two Bayesian regression models. Thus, as what people write in response to scaffold prompts when debugging a lower-complexity program demonstrates a more negative valence, debugging quality improves. Note that sentiment analysis is performed by a computer with no human intervention. When the computer deems writing to be more negatively valenced, that does not necessarily mean that the participant wrote negatively worded posts such as "I hate this scaffolding." Rather, a response to scaffold prompt of "I did X, but I should have done Y," would likely be interpreted as negatively valenced. However, that is the type of response that would lead to better debugging, as it evidences critical reflection on action, and on how to improve.

C. Notable Absences from the Final Models

The absence of mastery goals from the final models is striking because such are assumed to be the best predictor of persistence in the face of difficulty [63]. Participants with mastery goal orientations are said to engage in high challenge tasks because doing so has the potential to promote learning [11], [12]. Still, the regression models in this paper predicted debugging outcome quality, not willingness to engage in debugging. Furthermore, much research does not find a link between mastery goals and cognitive learning outcomes [64].

It is surprising that CS and engineering domain identification was not in the final models. Future studies should use additional data sources (e.g., interviews) to gauge participants' CS and engineering domain identification.

That STEM career interest was not a significant predictor makes sense in that all but one participant were ECE majors, presumably because of an interest in being ECE teachers.

D. Limitations and Threats to Validity

The study is observational and thus no causal relation between a participant's debugging process and motivation

variables and debugging outcome quality can be found. Due to the nature of these process and motivation variables, it is not possible to make the study a controlled experiment.

Sample size was small, which can lead to erroneous conclusions. Using a Bayesian approach can partially mitigate such concerns. By generating a distribution of true effects instead of a single point estimate, Bayesian regression can avoid overfitting, a problem often associated with Maximum Likelihood linear regression. However, non-informative priors can become informative with small sample size [47].

Using self-report data related to motivation relies on participants' appropriate interpretation of motivational constructs and principles. But this concern is strongest among K-12 participants [65]. This study included a behavior-related data source related to motivation: sentiment analysis of scaffolding responses and debugging reflections. Future research may consider engaging in further behavioral-based data collection methods related to motivation [65].

Because participants only engaged in two debugging tasks using the same Ozoblockly level and supported by scaffolding, only one debugging task for each debugging complexity level was used in this study. Future studies should employ more than one debugging task for each complexity level because different predictors and corresponding Betas could emerge for debugging outcome quality for different programming errors.

Only one block-based coding language (Ozoblockly) was used. It is beyond the scope of this paper to determine whether results can be generalized to other block-based coding languages. But Ozoblockly is similar to other block-based coding languages.

VII. Conclusion

Debugging process quality was a positive predictor, and performance-avoid goal orientation, and word count and sentiment analysis were negative predictors of the quality with which participants debugged a lower-complexity program. Debugging process quality and performance-approach goal orientation were positive predictors, and word count was a negative predictor of the quality with which participants debugged a higher-complexity program. This paper has important implications. First, scaffolding should challenge learners to engage in constructive criticism of their work. Greater challenge may cause writing sentiment to become more negatively valenced, but it can in turn lead to stronger debugging outcome quality. Next, performance-avoid goal orientations need to be addressed, because they are deleterious to the quality with which learners debug lower-complexity debugging tasks. Finally, scaffolding should enhance performance-approach goal orientations, as such are positive predictors of debugging quality of higher-complexity tasks.

VIII. References

[1] M. Bers, "The tangibleK robotics program: Applied computational thinking for young children," *Early Child. Res. Pract.*, vol. 12, no. 2, Sep. 2010.

[2] M. Bers and A. Ettinger, "Programming robots in kindergarten to express identity: An ethnographic analysis," in *Robots in K-12 education*, IGI Global, 2012, pp. 168–184.

[3] C. Kim, J. Yuan, L. Vasconcelos, M. Shin, and R. B. Hill, "Debugging during block-based programming," *Instr. Sci.*, vol. 46, no. 5, pp. 767–787, Oct. 2018, doi: 10.1007/s11251-018-9453-5.

[4] S. Fitzgerald *et al.*, "Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers," *Comput. Sci. Educ.*, vol. 18, no. 2, pp. 93–116, Jun. 2008, doi: 10.1080/08993400802114508.

[5] I. Vessey, "Expertise in debugging computer systems: A process analysis," *Int. J. Man-Mach. Stud.*, vol. 23, no. 5, pp. 459–494, Nov. 1985, doi: 10.1016/S0020-7373(85)80054-7.

[6] M. Ahmadzadeh, D. Elliman, and C. Higgins, "An analysis of patterns of debugging among novice computer science students," in *ITiCSE '05*, Caparica, Portugal, Jun. 2005, pp. 84–88, doi: 10.1145/1067445.1067472.

[7] K. M. Rich, C. Strickland, T. A. Binkowski, and D. Franklin, "A K-8 debugging learning trajectory derived from research literature," in *SIGCSE '19*, Minneapolis, MN, USA, 2019, pp. 745–751, doi: 10.1145/3287324.3287396.

[8] D. Spinellis, "Modern debugging: The art of finding a needle in a haystack," *Commun ACM*, vol. 61, no. 11, pp. 124–134, Oct. 2018, doi: 10.1145/3186278.

[9] R. McCauley *et al.*, "Debugging: A review of the literature from an educational perspective," *Comput. Sci. Educ.*, vol. 18, no. 2, pp. 67–92, Jun. 2008, doi: 10.1080/08993400802114581.

[10] T. Michaeli and R. Romeike, "Current status and perspectives of debugging in the k12 classroom: A qualitative study," in *IEEE EDUCON '19*, 2019, pp. 1030–1038.

[11] M. V. Covington, "Goal theory, motivation, and school achievement: An integrative review," *Annu. Rev. Psychol.*, vol. 51, no. 1, pp. 171–200, Feb. 2000, doi: 10.1146/annurev.psych.51.1.171.

[12] P. R. Pintrich, "Multiple goals, multiple pathways: The role of goal orientation in learning and achievement," *J. Educ. Psychol.*, vol. 92, pp. 544–555, 2000, doi: 10.1037/0022-0663.92.3.544.

[13] C. Senko, A. M. Durik, L. Patel, C. M. Lovejoy, and D. Valentiner, "Performance-approach goal effects on achievement under low versus high challenge conditions," *Learn. Instr.*, vol. 23, pp. 60–68, Feb. 2013, doi: 10.1016/j.learninstruc.2012.05.006.

[14] M. Appel and N. Kronberger, "Stereotypes and the achievement gap: Stereotype threat prior to test taking," *Educ. Psychol. Rev.*, vol. 24, pp. 609–635, 2012, doi: 10.1007/s10648-012-9200-4.

[15] D. B. Thoman, J. L. Smith, E. R. Brown, J. Chase, and J. Y. K. Lee, "Beyond performance: A motivational experiences model of stereotype threat," *Educ. Psychol. Rev.*, vol. 25, no. 2, pp. 211–243, Jun. 2013, doi: 10.1007/s10648-013-9219-1.

[16] S. Cheryan, A. Master, and A. N. Meltzoff, "Cultural stereotypes as gatekeepers: Increasing girls' interest in computer science and engineering by diversifying stereotypes," *Front. Psychol.*, vol. 6, 2015, doi: 10.3389/fpsyg.2015.00049.

[17] B. Reiser, "Scaffolding complex learning: The mechanisms of structuring and problematizing student work," *J. Learn. Sci.*, vol. 13, pp. 273–304, 2004, doi: 10.1207/s15327809jls1303_2.

[18] D. Wood, J. Bruner, and G. Ross, "The role of tutoring in problem solving," *J. Child Psychol. Psychiatry*, vol. 17, pp. 89–100, 1976, doi: 10.1111/j.1469-7610.1976.tb00381.x.

[19] S. N. Demetriadis, P. M. Papadopoulos, I. G. Stamelos, and F. Fischer, "The effect of scaffolding students' context-generating cognitive activity in technology-enhanced case-based learning," *Comput. Educ.*, vol. 51, pp. 939–954, 2008, doi: 10.1016/j.compedu.2007.09.012.

[20] M. Liu and S. Bera, "An analysis of cognitive tool use patterns in a hypermedia learning environment," *Educ. Technol. Res. Dev.*, vol. 53, no. 1, pp. 5–21, 2005, doi: 10.1007/BF02504854.

[21] B. R. Belland, A. E. Walker, N. J. Kim, and M. Lefler, "Synthesizing results from empirical research on computer-based scaffolding in STEM education: A meta-analysis," *Rev. Educ. Res.*, vol. 87, no. 2, pp. 309–344, 2017, doi: 10.3102/0034654316670999.

[22] B. R. Belland, A. E. Walker, and N. J. Kim, "A Bayesian network meta-analysis to synthesize the influence of contexts of scaffolding use on cognitive outcomes in STEM education," *Rev. Educ. Res.*, vol. 87, no. 6, pp. 1042–1081, 2017, doi: 10.3102/0034654317723009.

[23] J. E. Blumenstock, "Size matters: Word count as a measure of quality on wikipedia," in *WWW '08*, Beijing, China, Apr. 2008, pp. 1095–1096, doi: 10.1145/1367497.1367673.

[24] B. Rehder, M. Schreiner, M. Wolfe, D. Laham, T. Landauer, and W. Kintsch, "Using latent semantic analysis to assess knowledge: Some technical considerations," *Discourse Process.*, vol. 25, no. 2–3, pp. 337–354, Jan. 1998, doi: 10.1080/01638539809545031.

[25] M. Wen, D. Yang, and C. Rosé, "Sentiment analysis in MOOC discussion forums: What does it tell us?," 2014.

[26] L. C. Yu *et al.*, "Improving early prediction of academic failure using sentiment analysis on self-evaluated comments," *J. Comput. Assist. Learn.*, vol. 34, no. 4, pp. 358–365, 2018, doi: 10.1111/jcal.12247.
[27] A. Emerson *et al.*, "Predicting early and often: Predictive student modeling for block-based programming environments," in *EDM 2019*, 2019, vol. 39, p. 48.
[28] DataUSA, "Preschool & kindergarten teachers | Data USA," 2020. Accessed: May 28, 2020. [Online]. Available: https://datausa.io/profile/soc/preschool-kindergarten-teachers.
[29] National Center for Educational Statistics, "The Condition of Education - Preprimary, Elementary, and Secondary Education - Teachers and Staff - Characteristics of Public School Teachers - Indicator May (2020)," 2020. Accessed: May 28, 2020. [Online]. Available: https://nces.ed.gov/programs/coe/indicator_clr.asp.
[30] G. Saluja, D. M. Early, and R. M. Clifford, "Demographic characteristics of early childhood teachers and structural elements of early care and education in the united states," *Early Child. Res. Pract.*, vol. 4, no. 1, 2002.
[31] Ozobot and Evollve, "OzoBlockly," 2019. https://ozobot.com/ozoblockly (accessed Aug. 02, 2019).
[32] D. H. Jonassen, "Toward a design theory of problem solving," *Educ. Technol. Res. Dev.*, vol. 48, no. 4, pp. 63–85, 2000, doi: 10.1007/BF02300500.
[33] G. Knezek and R. R. Christensen, "STEM Semantics Survey." 2008, [Online]. Available: http://stelar.edc.org/sites/stelar.edc.org/files/STEMSemantics10x.pdf.
[34] R. Christensen, G. Knezek, and T. Tyler-Wood, "Student perceptions of Science, Technology, Engineering and Mathematics (STEM) content and careers," *Comput. Hum. Behav.*, vol. 34, pp. 173–186, May 2014, doi: 10.1016/j.chb.2014.01.046.
[35] R. Pekrun, T. Goetz, A. C. Frenzel, P. Barchfeld, and R. P. Perry, "Measuring emotions in students' learning and performance: The Achievement Emotions Questionnaire (AEQ)," *Contemp. Educ. Psychol.*, vol. 36, no. 1, pp. 36–48, Jan. 2011, doi: 10.1016/j.cedpsych.2010.10.002.
[36] J. S. Rosas, "The Achievement Emotions Questionnaire-Argentine (AEQ-AR): internal and external validity, reliability, gender differences and norm-referenced interpretation of test scores," *Rev. Evaluar*, vol. 15, no. 1, 2015.
[37] A. E. Black and E. L. Deci, "The effects of instructors' autonomy support and students' autonomous motivation on learning organic chemistry: A self-determination theory perspective," *Sci. Educ.*, vol. 84, no. 6, pp. 740–756, 2000, doi: 10.1002/1098-237X(200011)84:6<740::AID-SCE4>3.0.CO;2-3.
[38] C. P. Cerasoli and M. T. Ford, "Intrinsic motivation, performance, and the mediating role of mastery goal orientation: A test of self-determination theory," *J. Psychol.*, vol. 148, no. 3, pp. 267–286, 2014.
[39] J. L. Smith, C. L. Morgan, and P. H. White, "Investigating a measure of computer technology domain identification: A tool for understanding gender differences and stereotypes," *Educ. Psychol. Meas.*, vol. 65, no. 2, pp. 336–355, 2005.
[40] J. L. Smith and P. H. White, "Development of the domain identification measure: A tool for investigating stereotype threat effects," *Educ. Psychol. Meas.*, vol. 61, no. 6, pp. 1040–1057, 2001.
[41] C. Midgley *et al.*, *Manual for the Patterns of Adaptive Learning Scales (PALS)*. Ann Arbor, MI: University of Michigan, 2000.
[42] A. Yadav, N. Zhou, C. Mayfield, S. Hambrusch, and J. T. Korb, "Introducing computational thinking in education courses," in *SIGCSE '11*, 2011, pp. 465–470.
[43] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*, 5th ed. Boca Raton, FL, USA: CRC Press, 2011.
[44] J. L. Schafer, "Multiple imputation: A primer," *Stat. Methods Med. Res.*, vol. 8, no. 1, pp. 3–15, Feb. 1999, doi: 10.1177/096228029900800102.
[45] S. Feuerriegel and N. Proellochs, *SentimentAnalysis R package*. 2019.
[46] B. Liu, "Sentiment analysis and opinion mining," *Synth. Lect. Hum. Lang. Technol.*, vol. 5, no. 1, pp. 1–167, May 2012, doi: 10.2200/S00416ED1V01Y201204HLT016.
[47] D. M. McNeish, "Challenging conventional wisdom for multivariate statistical models with small samples," *Rev. Educ. Res.*, vol. 87, no. 6, pp. 1117–1151, Aug. 2017, doi: 10.3102/0034654317727727.
[48] R. J. Little, "Calibrated Bayes: A Bayes/frequentist roadmap," *Am. Stat.*, vol. 60, pp. 213–223, Aug. 2006, doi: 10.1198/000313006X117837.
[49] A. Smith and A. Gelfand, "Bayesian statistics without tears: A sampling-resampling perspective," *Am. Stat.*, vol. 46, no. 2, pp. 84–88, 1992, doi: 10.2307/2684170.
[50] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*, 3rd ed. Boca Raton, FL, USA: CRC Press, 2013.
[51] M. Kutner, C. Nachtsheim, J. Neter, and W. Li, *Applied linear statistical models*, 5th edition. Boston, MA: McGraw-Hill, 2004.
[52] J. Curran, *Bolstad: Functions for elementary Bayesian inference*. 2018.
[53] A. D. Martin *et al.*, *MCMCpack: Markov Chain Monte Carlo (MCMC) package*. 2018.
[54] D. H. Jonassen, *Learning to solve problems: A handbook for designing problem-solving learning environments*. New York, NY, USA: Routledge, 2011.
[55] M. D. Mumford, "Process-based measures of creative problem-solving skills: II. Information encoding," *Creat. Res. J.*, vol. 9, no. 1, p. 77, Jan. 1996, doi: 10.1207/s15326934crj0901_7.
[56] N. Rountree, J. Rountree, and A. Robins, "Predictors of success and failure in a CS1 course," *SIGCSE Bull*, vol. 34, no. 4, pp. 121–124, Dec. 2002, doi: 10.1145/820127.820182.
[57] B. C. Wilson and S. Shrock, "Contributing to success in an introductory computer science course: A study of twelve factors," in *SIGCSE '19*, New York, NY, USA, 2001, pp. 184–188, doi: 10.1145/364447.364581.
[58] C. Watson, F. Li, and J. L. Godwin, "No tests required: Comparing traditional and dynamic predictors of programming success," in *SIGCSE '14*, New York, Jan. 2014, pp. 469–474, doi: http://doi.acm.org/10.1145/2538862.2538930.
[59] C. Senko and B. Dawson, "Performance-approach goal effects depend on how they are defined: Meta-analytic evidence from multiple educational outcomes," *J. Educ. Psychol.*, vol. 109, no. 4, pp. 574–598, May 2017.
[60] I. Roll, R. S. J. d Baker, V. Aleven, and K. R. Koedinger, "On the benefits of seeking (and avoiding) help in online problem-solving environments," *J. Learn. Sci.*, vol. 23, no. 4, pp. 537–560, Oct. 2014, doi: 10.1080/10508406.2014.883977.
[61] E. D. Deemer, M. Yough, and S. A. Morel, "Performance-approach goals, science task preference, and academic procrastination: Exploring the moderating role of competence perceptions," *Motiv. Emot.*, vol. 42, no. 2, pp. 200–213, Apr. 2018, doi: 10.1007/s11031-017-9649-z.
[62] A. Elliot and H. McGregor, "A 2 × 2 achievement goal framework.," *J. Pers. Soc. Psychol.*, vol. 80, no. 3, pp. 501–519, Mar. 2001.
[63] M. Vansteenkiste, J. Simons, W. Lens, B. Soenens, L. Matos, and M. Lacante, "Less is sometimes more: Goal content matters," *J. Educ. Psychol.*, vol. 96, pp. 755–764, 2004.
[64] J. L. Meece, E. M. Anderman, and L. H. Anderman, "Classroom goal structure, student motivation, and academic achievement," *Annu. Rev. Psychol.*, vol. 57, no. 1, pp. 487–503, Jan. 2006, doi: 10.1146/annurev.psych.56.091103.070258.
[65] S. M. Fulmer and J. C. Frijters, "A review of self-report and alternative approaches in the measurement of student motivation," *Educ. Psychol. Rev.*, vol. 21, no. 3, pp. 219–246, Sep. 2009, doi: 10.1007/s10648-009-9107-x.

Brian R. Belland earned a BA in French from the College of Wooster in 1999, and a PhD in Educational Technology from Purdue University in 2008. He is an associate professor of Educational Psychology at Penn State. He published 1 book, 43 articles, and 9 book chapters. His honors include an NSF CAREER award, and 6 best paper awards.

ChanMin Kim received a BA in special education from Ewha Women's University, Seoul, Korea in 1998, and a PhD in Instructional Systems from Florida State University in 2007. She is an associate professor of Learning, Design, and Technology and Educational Psychology at Penn State. She published 41 articles and 11 book chapters. She received multiple research, design, and software programming awards.

Anna Y. Zhang earned a BS in Statistics from, and is pursuing a PhD in Educational Psychology at Penn State. Her awards include the Matthew Rosenshine Fund for Excellence in Statistics and a Dean's Graduate Assistantship.

Afaf A. Baabdullah earned a BA in Computers & Education in 2001. She is pursuing a Ph.D. in Learning, design, and technology at Penn State, and is faculty in the department of Curriculum and Instruction at King Saud University.

Eunseo Lee received a BA in English in 2009 from Yonsei University. She is pursuing a PhD in Educational Psychology at Penn State.