11-24-2018

The Relationship between Learner Characteristics and Student Outcomes in a Middle School Computing Course An Exploratory Analysis Using Structural Equation Modeling

Tuba Ketenci
Georgia Institute of Technology

Brendan Calandra
Georgia State University

Lauren Margulieux
Georgia State University

Jonathan Cohen
Georgia State University

Recommended Citation

Ketenci, T., Calandra, B., Margulieux, L. & Cohen, C. (2019). The Relationship between learner characteristics and student outcomes in a middle school computing course: An exploratory analysis using structural equation modeling. Journal of Research on Technology in Education. 51(1), 63-76. https://doi.org/10.1080/15391523.2018.1553024

The Relationship between Learner Characteristics and Student Outcomes in a Middle School Computing Course

An Exploratory Analysis Using Structural Equation Modeling

See below for published version:

Ayer, T., Calandra, B., Margulieux, L. & Cohen, C. (2019). The Relationship between learner characteristics and student outcomes in a middle school computing course: An exploratory analysis using structural equation modeling. *Journal of Research on Technology in Education. 51*(1), 63-76. https://doi.org/10.1080/15391523.2018.1553024

Abstract

This study, which took place during a 7-week middle school computing course, used structural equation modeling to examine the overall cumulative relationship between self-efficacy, interest, and prior computing experience and students' computer science learning outcomes. The findings indicated that 52% of the variance of student success, measured by a computational thinking quiz and rubric-based evaluations of participants' computing artifacts, was related to the aforementioned learner characteristics. These findings have implications for theory and practice and suggest that future research and instructional design practice in K-12 computing education should take these learner characteristics into account.

Introduction

Computer science (CS) in K-12 schools is receiving considerable attention internationally (Balanskat & Engelhardt, 2015; Grover & Pea, 2013; Kafai & Burke, 2017). Take Ericson's Analysis of 2017 AP CS exam participation as an example: "AP CS A exam takers (60,519 students) grew by 11.2 percent year-over-year in 2017, and 61.8 percent of them passed the exam" (Guzdial, 2018). The need for CS education is also evident in high profile initiatives such as the $4 billion in funding proposed for the White House's Computer Science for All initiative ("Computer science for all," 2016) and the 2017 Presidential memo to increase access to STEM and CS Education through $200 million per year of support (Increasing Access to High-Quality STEM Education, 2017). This movement to support CS education can be seen not only at the national but also at the state level ("Computer science mania," 2018). In addition, the National Science Foundation has demonstrated its dedication to providing opportunities for all students to learn CS ("Computer science is for all students," 2018), and the National Academies of Sciences, Engineering, and Mathematics have likewise shown commitment ("Workshop on the growth of computer science," 2016).

This attention, and the fact that many state school systems are now requiring CS as part of their core curriculum (Balanskat & Engelhardt, 2015), demands that careful attention be paid to CS instructional designs for K-12 (Dawson, Allen, Campbell, & Valair, 2018). Another issue of equal importance to the field, the profession, and society in general involves a need to focus on broadening participation in CS learning, especially of women and underrepresented minorities (Margolis, Estrella, Goode, Holme, & Nao, 2008).

Literature Review

One of the major issues pointed out in the literature that can negatively affect more global participation in CS learning may have to do with past use of instructional designs that either do not necessarily connect with all students' prior knowledge and experience (Araujo, Bittencourt, & Santos, 2018), or that present only shallow curricula to underrepresented students (Margolis et al., 2011), thus potentially perpetuating an elitist culture associated with the computing world, and perpetuating a cycle of exclusivity (Lachney, 2018; Scott, Sheridan & Clark, 2015). One might assume then that cognitive and affective learner characteristics associated with prior knowledge and experience might be important design considerations for CS curricula and learning environments (Lishinski, Yadav, Good, & Enbody, 2016). Indeed, factors such as self-efficacy, interest in computing, and prior computing experience should be considered especially when designing more inclusive CS learning environments, which might attract and retain students who in the past have not always been included in CS education because they feel it either does not speak to them, or it is beyond their ability (Watson, Li, & Godwin, 2014; Kafai, Searle, Martinez, & Brayboy, 2014).

While some studies have examined demographic differences among students such as age, race, socioeconomic status, and gender in their instructional designs (e.g., Margolis et al., 2011; Hatley, 2016), fewer have explicitly considered self-efficacy, interest, and/or prior experience, which has produced multiple calls for further research (Almstrum, Hazzan, Guzdial, & Petre, 2005; Robins, 2015). In one of these studies, Lishinski et al. (2016) investigated the relationships between college students' gender, interest, goal, self- efficacy, and performance, finding via structural equation modeling (SEM) that the best indicator for performance was self-efficacy. Wiedenbeck (2005) also employed SEM via path analysis and studied the effect of self-efficacy on non-major students' CS performance, again finding that

self-efficacy was a significant predictor of student success. Haungs et al. (2012) developed a college level curriculum which provided a certain level of flexibility to their students. The curriculum included different learning tracks (e.g., robotics, gaming, music, mobile apps), from which students were able to choose according to their personal interest. Haungs et al. (2012) argued that their intervention, which was designed based on student interest and personal choice, increased academic performance and retention. Beyer (2014) investigated the relationship between students' grades in a computer science class and predictors such as self-efficacy, interests, and prior experiences. She discovered that students in her study with an interest in computer science were more likely to stay in the course, and more likely to earn higher grades.

While studies of the relationship between students' prior experience and CS learning outcomes in higher education are mixed (e.g., Beyer, 2014 vs. Watson et al., 2014), in a recent study in K-12 CS, Blanchard, Gardner-McCune, & Anthony (2018) found that a group of elementary and middle school students' perceptions of programming in general, and of difficulty working with different constructs related to Scratch-based game development was influenced by their prior experience. Also Grover, Pea, and Cooper (2016) found that a group of middle school students' prior experience with computing technologies did predict higher grades in an online CS course. In the current study, we also examined the relationships between learner characteristics and CS learning outcomes among a group of middle school students. The goal of this exploration, however, was to build upon studies such as Grover et al. (2016) to determine whether these factors should be considered in future studies, and also to inform the development of more inclusive CS instructional designs for K-12.

The main research question in the current study is as follows: What is the relationship between middle school students' overall learner characteristics and their performance in a CS course?

Theoretical Frame

The theoretical approach that guides this study incorporates tenets of constructionist learning theory pioneered by Papert (1991). Constructionism gets its roots from the pioneering work of Piaget's (1952) on constructivist learning theory, which argues that learning is a process in which students construct knowledge through experiencing things and reflecting on those experiences (Driscoll, 1994). Constructivism has changed the instructional design of classroom teaching and more broadly educational practice significantly, from teacher-led instruction to student-centered design (Yadin, 2011). Taking constructivism one step further, Papert and Harel (1991) argued that learning occurs in the process of constructing socially or personally relevant public artifacts that connect previous experiences with the new ones, while interacting with others. In other words, constructionist learning theory suggests that students learn deeply when they create an artifact that involves understanding the system and the concept (Newstetter, 2000) because, as Barron and Hammond (2008) stated, learning by building requires "students to set constraints, generate ideas, create prototypes, and develop plans through storyboarding or other representational practices" (p. 7). Constructionist learning contexts in computing provide opportunities for students to master the discipline content while creating artifacts of their own choice (Bers, 2008) and foster problem-solving, reasoning, and computational thinking skills (Kafai & Burke, 2015). Students' artifacts might be many things, such as LEGO machines, apps, or virtual objects (Bruckman & Resnick, 1995). What is important is that their artifacts are meaningful to themselves and others around them (Bruckman & Resnick, 1995).

Method

Motivated by the positive potential that constructionist approaches to learning CS may have, the purpose of this study was to examine the relationship that constructionistic learner characteristics (i.e., self-efficacy, interest, and prior experience with creative computing) might have with middle school student outcomes related to computational thinking. Data collection and analysis were guided by the following question: Was there any relationship between a set of learner characteristics and middle school students' performance on two measures of computational thinking after participating in a series of constructionist computer science activities?

This study employed quantitative methods for data collection and analysis using structural equation modeling (SEM) as a framework for presenting and interpreting the results. The authors chose SEM as a statistical tool to analyze the relationship between the constructs in this study, because SEM allows for analysis of all variables in the model simultaneously rather than separately.

Context

This study was conducted within a private, STEM-focused school located in a large Southeastern city. The course in question included middle school-level lessons on Web design, graphic design, and coding basics. The course took place during the regular school day in a computer lab, and it was a required part of the curriculum at the school. The computer lab included enough computers for students to either work individually or collaboratively. This study focused on the implementation of a collaborative, problem-based instructional design within the course for helping participants to learn programming using MIT's App Inventor (http://ai2.appinventor.mit.edu). App Inventor is a block-based programming language that uses a graphical programming environment and enables students

to create applications (apps) for Android mobile devices. Part of the rationale for choosing this learning tool was that App Inventor is a "gender neutral and truly democratic" (Grover & Pea, 2013, p. 726) tool to teach computing. The instructional design included self-paced, direct instruction (Kirschner, Sweller, & Clark, 2006; Anderson, Reder, & Simon, 1999) to introduce App Inventor features and some computing concepts, plus less structured computational problem-solving activities (Guzdial, 2009) created to promote student's deeper CS learning in the form of computational thinking (Grover, 2016; Kafai & Burke, 2015).

Participants

A convenience sample of 142 middle school students participated in the study (Etikan, Musa, & Alkasim, 2016). Thirty-one of the students were excluded from the final analysis for not completing all assessments. Female students accounted for 56.8% of participants; males, 29.7%; and 13.5% of the students did not provide their gender information in the survey. Class size ranged from 18 to 21 students and 7 classes participated in this study under the supervision of 2 teachers who had been assigned to teach the course. One of the teachers had a computer science background, and the second had a degree in industrial engineering. The first author conducted a two-hour teacher orientation that focused more on the introduction to the curriculum, intervention, and data collection procedures than the concepts in the curriculum as the teachers already had experience with App Inventor, and thus were able to facilitate lessons. After the initial orientation, the researcher held weekly, online progress checks with the teachers. Finally, the researcher was available online during the days classes were taking place in case teachers had any real-time questions or concerns related to the study.

Procedure

The App Inventor intervention took place in a computing class one day a week for 50 minutes over a period of 7 weeks, for a total of 6 hours of class time over a single semester. During the first week of the course, students were given an orientation during which they took the adapted version of Barron et al.'s (2009) survey, described below. From weeks 2 to 5 of the intervention, students completed self-paced activities designed to help them learn the features of App Inventor. Activities were distributed to students based on gradually increasing difficulty, and decreasing amounts of scaffolding. The first set of activities involved students developing pre-designed apps using App Inventor by following step-by-step directions. These activities intentionally exposed participants to computational concepts and practices such as conditionals, testing, and debugging (Brennan & Resnick, 2012). Upon the completion of these initial activities, students were given a problem-based activity with fewer direct instructions, as recommended by Guzdial (2009). These activities were similar in design to the performance-based assessment that they would ultimately be required to complete, in that students were presented a problem and asked to design a solution using App Inventor, following Grover's (2016) recommendation. They were also given a chance to pick their own app design and/or development problem to solve.

Unlike the step-by-step App Inventor activities, the problem-based activities gave students the flexibility to incorporate their home cultures and daily lives into their computing learning experience during class time if they so choose. Most students worked in self-organized teams of two or three to design their mobile apps, and some of the students worked individually. During the 6th week, the students completed a performance-based assessment in which they were asked to customize an app that gives health recommendations to a user according to the user's selection. At the end of the intervention, the teacher shared students'

digital artifacts with the researchers for performance-based evaluation. One month later, students took the CT quiz, described below.

Measures

In response to Grover's (2016) call for multiple complimentary assessments that attend to both cognitive and non-cognitive aspects of learning CS, two sets of measures were used in this study. One set included two measures to assess learning outcomes, and the other set assessed three cognitive and affective mediating variables (learner characteristics).

CS learning. The latent variable "CS learning" was measured by using two assessment instruments: a computational concepts and practices performance-based grade, and a quiz score.

Computational concepts and practices performance score. This score was calculated using Sherman and Martin's (2015) rubric to analysis apps that students had made. Sherman and Martin originally developed the rubric to evaluate CS and non-CS majors' mobile apps created using App Inventor. They found out that the rubric was able to detect important and subtle differences in mobile computational thinking projects (Sherman & Martin, 2015). The rubric consists of 16 criteria for assessing students' CT in programming with App Inventor. Some of the criteria are specific to App Inventor (e.g., use of accelerometer and orientation sensors and screen interface) and some of them are more general (e.g., events and algorithms). Students can get a score between 1 and 4 for each criteria of the rubric. In the present study, the authors used all thirteen criteria in the rubric to create a performance grade with a maximum score of 64. Across the sample (N= 109), performance scores ranged from 16 to 29 (M= 21.65, SD= 4.05).

The performance-based scores were the scored by two graduate research assistants with CS and educational technology backgrounds using Sherman and Martin's (2015) rubric to evaluate student artifacts. Given the number of subsections in the CT rubric (Shermin & Martin, 2015), the research team adopted a procedure for inter-rater reliability that had been employed in a prior, similar study (Israel et al., 2017). Agreement among raters for each subsection in this study ranged between 90% and 100%, thus was deemed within an acceptable range.

See Figures 1 and 2 for an example of a student artifact called the Health Calculator. This app did not have any design problems; however, the code blocks were not complete. In this case the student created an event with "when button click," but did not complete it, thus the app was not reacting to user input.

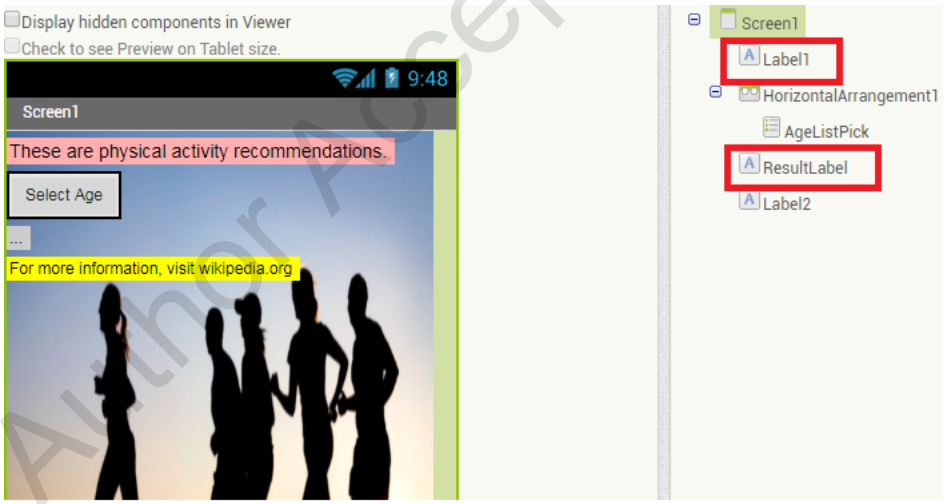

Figure 1. Sample design screen

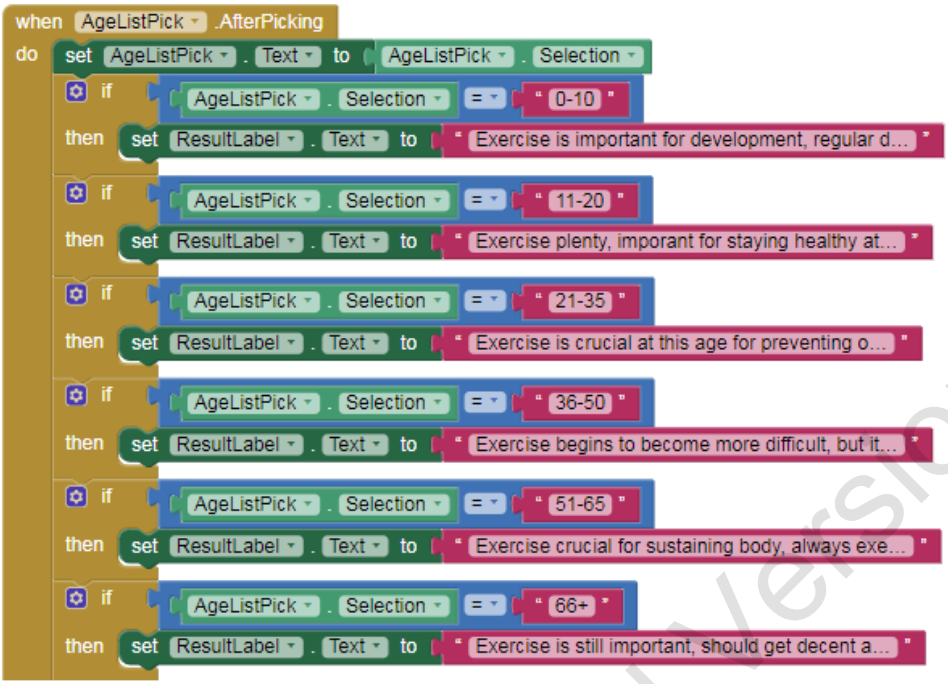

Figure 2. *Sample code block*

Computational concepts and practices quiz score. This was calculated via a custom-built, multiple choice assessment called the CT Quiz. The authors of this paper developed the CT quiz (Authors, 2018) based closely on Brennan and Resnick's (2012) computational thinking framework, which is widely used in K-12 computing education studies and outlines the CS concepts and practices through visual block-based programming. The dimensions of the computational thinking framework are concepts, practice, and perspectives (Brennan & Resnick, 2012). Computational concepts include the use of sequences, loops, parallelism, events, conditionals, operators, and data while programming. The computational practices are abstracting and modularizing, reusing and remixing others' projects, and being incremental and iterative. The final dimension is computational perspectives, which is when students express themselves, connect with others, and question the limitations of their design through computing. Each question in the CT Quiz refers to one of the CT concepts outlined in the framework. Items in the instrument were designed to allow

students to demonstrate CT abilities by answering multiple choice questions about App Inventor. For example, the question in Figure 3 below asks students to troubleshoot an App Inventor code sequence. Though students answered 18 CT quiz items, only 11 were used for this analysis after reliability testing (α = .802). While the maximum quiz score was 11, student scores in this study ranged from 1 to 7 (N= 71, M= 4.75, SD= 1.391).

Question: You realize that your code for your app above is doing the calculation wrong for your fitness level. It should multiply the height by two and then divide by 10. Please choose the appropriate code block below to solve the problem.

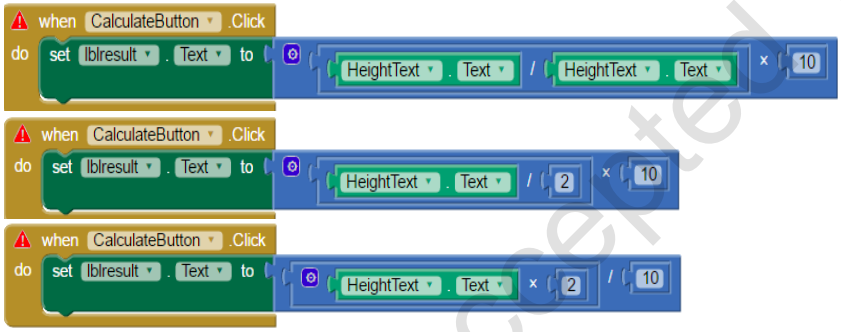

Figure 3. Sample question from the computational concepts and practices quiz.

Learner characteristics. Another latent variable in the proposed model is learner characteristics measured through self-report items on a survey. Learner characteristic "refers to the existing factors that influence the student engagement before the learning takes place" (Biggs & Tan, 2001, p. 136). Biggs and Tan defined learner characteristics as the latent variable in their 3P (Presage, Process, and Product) model of learning that includes personal differences, including prior experience and knowledge, self-efficacy, and interest. We employed Barron et al.'s (2009) survey to gather learner characteristics for our sample population. Thus, our learner characteristic latent variable is the composite of student's self-

efficacy in computing, interest in learning computing technology, and prior experience with creative computing technologies.

Self-efficacy. Students' self-efficacy was measured with the response to one self-efficacy-specific question. The item for self-efficacy was as follows: "I feel confident in my ability to learn how to build phone apps." The answer choice was presented in a five-item Likert scale from 1 to 5, indicating strongly disagree to strongly agree respectively.

Interest. Students' interest in learning about computing technology was measured based on the scheme presented in Barron et al.'s survey (2009). The Cronbach's alpha for the items in their study was 0.83, establishing and indicating sufficient reliability of the items. In our test, we asked students the three interest-specific five-item scale questions listed in Barron et al. and one additional question regarding their interest in App Inventor. The additional question measured student interest in the program, with choices including "I cannot wait to get started!" and "I am not very interested," and "I do not want to do it." The Cronbach's alpha is 0.85 in our study, which is very close to that in Barron et al.

Prior experience with creative computing. Students' prior experience with thirteen creative production activities were derived from student responses to a set of questions from Barron et al.'s survey (2009). Barron et al. picked activities that involved some aspects of design, personal expression, and/or the application of more advanced concepts related to computing. Students' involvement with these activities are measured by a series of questions about the extent of their prior experience with creative production. Questions about experience (e.g., "How often have you coded a website using HTML?") were presented in a five-item Likert scale item with "I have never done this," "I've done this once in my life," "I've done this a few times," "I've done this a lot," and "I don't know what this is." After

that, students' depth and breadth of experience were calculated as follows to create the four types of student profiles of experience.

Depth of prior experience with creative computing technologies. Students' depth of prior experience predictor was calculated according to Barron et al.'s (2009) description in their journal article. The number of activities that each student participated more than five times are calculated and served as a measure of the depth of students' experience with creative computing technologies. The median split method is employed to determine the students with a high and low depth of prior experience. Thus, this is a binary variable.

Breadth of prior experience with creative computing technologies. The breadth of the prior experience is calculated similarly to Barron et al.'s (2009) method, by summing up the number of activities each student had participated in at least once. The range could be from 1 to 13. The median split method is used to determine the high breadth vs. low breadth groups. Thus, this is a binary variable. As described in Barron et al.'s article, students with low breadth and low depth are categorized *beginner*. Students with low breadth and high depth are categorized *specialist*. Those with high breadth and low depth are categorized *explorer*, and, finally those with high breadth and depth are categorized *generalist*. The Cronbach's alpha for prior experience with creative computing questions was acceptable, $\alpha = .86$. Across the sample ($N = 97$), 42% of the students were classified as beginners, 28% as specialists, 8% as explorers, and 20% as generalists.

Data Analysis

Participants' CT Quiz scores and performance-based scores were used for this analysis. Structural equation modeling (SEM) was used in this study to consider the overall

relationship that prior experience, self-efficacy, and interest had with student learning within the particular computational, problem-solving context described above.

The model proposed in Figure 4 includes our hypotheses as follows: computational concepts and practices performance score and computational concepts and practices quiz grade measure CS learning.

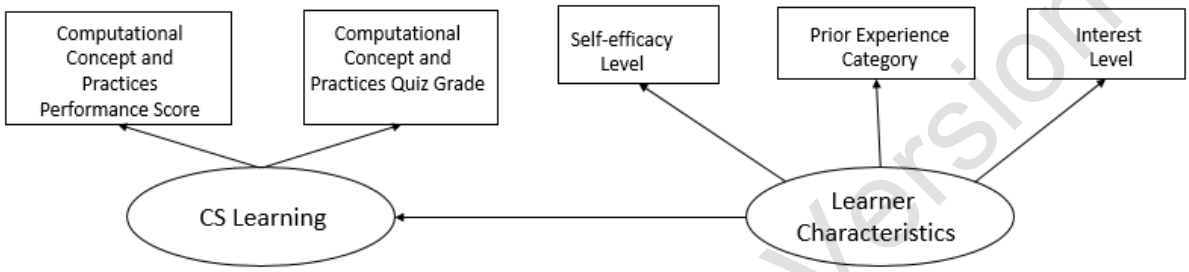

Figure 4. Model for relation between CS Learning and Learner Characteristics.

Mplus 7.0 (Muthen & Muthen, 2012) was used for SEM with two latent and five observed variables. This was done in order to validate the hypothesized structure. Given the continuous nature of the items, Robust Maximum Likelihood estimation (RML) was also used.

In this study, several absolute and incremental fit indices were used such as the Comparative Fit Index (CFI), the Chi-square test, and Standardized Root Mean Squared Residuals (SRMR) to evaluate the overall discrepancy between observed and model-implied (co)variances (Hooper et al., 2008). According to Hu and Bentler (1998), a root means square error of the approximation comparative fit index value of 0.95 or higher indicates a close fit and values up to 0.90 indicate a reasonable fit. In addition, the Chi-square test result should not be significant to have a good model fit (Bagozzi & Yi, 1988). The value of SRMR should be less than 0.05 for well-fitting models (Diamantopoulos et al., 2000). However, values as high as 0.08 are considered as within acceptable range (Hu & Bentler, 1998). We did not report the root mean square error of

the approximation (RMSEA) value in the current study because the use of this model fit the index in the studies with small sample size and degrees of freedom, and it is not recommended in the literature (Kenny, Kaniskan, & McCoach, 2015).

The sample size for the analysis was 142. The percentage of missing data for predictors varied between the ranges of 0% to 31%. We conducted a missing value analysis. The Missing Completely at Random (Little, 1998) test shows that the missing data was not systematic, $\chi 2$ (14) = 17.255, $p > 0.05$. When the robust maximum likelihood estimation is used, the Method of Full information Maximum Likelihood (FML) is adopted to deal with the missing data. FML is the process used to estimate a likelihood function for each case based on the variables that are present so that all the available data are used as a type of maximum likelihood (Enders & Bandalos, 2001).

Results & Discussion

The present study examined the relationship of a set of learner characteristics on a group of middle school students' CS learning outcomes during a 7-week computing course using MIT's App Inventor. Findings showed that the SRMR value for the model proposed was 0.055, and the CFI value was 0.935. The SRMR and CFI values indicate adequate model fit, accurately representing the observed relationships among the variables introduced in the model. The result of a Chi-square test was not statistically significant showing a good model fit, $\chi 2$ (4) = 8.47, $p = 0.07$. Mplus also suggested only one modification for model fit improvement such as adding error covariance between the "interest" and "quiz" variables. However, we did not make that change because those variables do not represent similar constructs. Furthermore, as shown in Figure 5, all the factor loadings were statistically significant at the 0.01 level, with standardized factor loadings ranging from 0.44 to 0.95.

The standardized path coefficients are presented in Figure 5. All the path coefficients were statistically significant. Each coefficient represents the standard model result meaning that the change in Y associated with one resulted in a standard deviation increase in X. For example, in our research, we found that the students' CS learning improved by .72 standard deviations given a change of one standard deviation in the overall learner characteristics. The unstandarized result showed that one-point increase in the overall learner characteristic resulted in 1.5 point increase out of 18 for the middle school students that the study took place. Since our sample consisted of students from a private middle school population in a major Southeastern metropolitan area, its generalizibity is limited. Similar studies should be conducted in different contexts to generalize the results to all middle school students in the U.S.

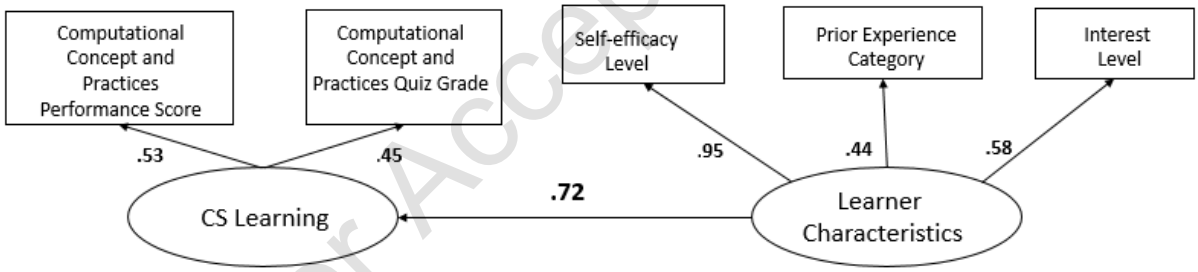

Figure 5. A standardized solution of the structural equational modeling.

The R^2 value suggests that the overall combination of learner characteristics accounted for 52% of the variance in the students' CS learning in this study, which is generally considered a moderate effect size (Moore, Notz, & Flinger, 2013; Zikmund & William, 2000). Future studies may use advanced command available in MPlus to account for the nesting of the data which is not taken into consideration in this study.

The value of the path coefficients shows the relative strength of the relationships between the students' self-efficacy, initial interest in the program, and prior creative

production experience. The strongest coefficient was self-efficacy, underscoring the pivotal role of self-efficacy in the learning process and providing evidence of its influence on learner characteristics that are important to build CS learning. The students' initial interest in the curriculum was also a good measure of learner characteristics and exceeded that of prior experience on the overall subject and can be measured through self-reported items. We were surprised to find that prior experience with creative computing projects was not as strong as interest in predicting learner characteristics that impact the learning which was assessed at the end of the study. While this model fits well, another structural model should be built with the addition of factors measured during the instruction, such as collaboration among students, and time spent on the task (Watson et al., 2014).

The two subscales of CS learning, computational concepts, and practices performance grade and quiz, were significantly correlated with each other but lower than the recommended cut-off of 0.85 for distinguishing for an additional factor model (Kenny, 2012). This means that they are similar but distinct enough to remain separate. In addition, the subscales of learner characteristics—self-efficacy, interest, and prior experience—were also significantly correlated with each other at the alpha level .05. Please see Table 1 for further information.

Table 1

Correlations among Subscales of Learner Characteristics (Self-efficacy (1), Interest (2), Prior Experience (3)) and CS Learning (Grade (4) & Quiz (5))

	1	2	3	4	5	Mean	SD	Cronbach's α
Self-efficacy	-	.556**	.402**	.307**	.406**	3.69	.92	NA
Interest	-	-	.334**	.174	.073	3.79	1.01	.853
Prior Exp.	-	-	-	.139	.249*	1.09	1.17	.862

Grade	-	-	-	-	.233*	21.65	4.049	NA
Quiz Score	-	-	-	-	-	8.61	2.51	NA

Note. $n = 111$. * $p < .05$, ** $p < .01$.

Performance grades and the CT quiz are good measures of CS learning

Measuring CS learning has been a challenge for educators and researchers (Grover, 2016; Grover, Pea, & Cooper, 2015). While some previous studies have built models on a one-outcome variable, e.g., students' grade on a project or exam (Roman-Gonzalez et al., 2016), the model presented in this paper included two different types of assessment scores. Data from both instruments reinforced the results, and aligned with previous research suggesting multiple assessment types can be useful for understanding CS learning in K-12 CS education (Brennan & Resnick, 2012; Grover, 2016). Thus, this study contributes to the literature by presenting what could become a reliable set of measures for assessing middle school students' CS learning. This is important because, as Grover and Pea (2013) noted, without an appropriate assessment, computing education has little chance to be an integral part of K-12 education.

Conclusion

This study was intended to help the CS education community to further understand how learner characteristics were related to CS learning outcomes in one particular context by examining a group of middle school students working through direct instruction and guided practice/problem solving activities using MIT's App Inventor. The findings showed that 52% of the variation in students CS learning could be explained by the differences in students' learner characteristics.

While acknowledging the limitations listed in the previous sections, the model proposed in this paper suggests that prior experience, self-efficacy, and interest in the content were valid measures for students' learner characteristics. Based on our results, we recommend that: a) this study be replicated with a larger population, b) future studies take the same approach, but apply it in varied contexts and with a diverse array of K-12 students and measure performance and multiple points, and c) future studies test the model presented here in a variety of K-12 CS learning environments. Although more research is necessary to make general recommendations, based on our findings, we would recommend that CS instructional designs for K-12 include relevant computing activities that: a) build upon students' prior knowledge and experience, b) attract them to and help them to remain interested in computing. We also suggest instructional designs that help to increase students' self-efficacy which may increase access to CS learning for all students and improve CS learning outcomes. It may be of interest that the activity design in this study of direct instruction followed by guided practice, as well as presenting activities for students to choose from with gradually increasing levels of difficulty, was meant to help bolster student agency and self-efficacy.

Overall, these results highlight the strong relationship that individual characteristics, especially self-efficacy, had on a group of middle school students' CS learning outcomes, demonstrating that these characteristics were part of a relationship with CS learning. These findings should be of interest to researchers and practitioners involved in CS education.

Acknowledgments

This research was supported by grant from the National Science Foundation (grant number: 1433280). Any opinions, findings, and conclusions expressed are those of the

authors and do not necessarily reflect the views of the sponsoring agencies. We are grateful for the participation of the students and teachers involved.

References

Almstrum, V. L., Hazzan, O., Guzdial, M. & Petre, M. (2005). Challenges to computer science education research. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05)*. pp. 191–192. New York, NY: ACM.

Anderson, J. R., Reder, L. M., & Simon, H. A. (1999). Applications and Misapplications of Cognitive Psychology to Mathematics Education.

Araujo, L. G. J., Bittencourt, R. A., & Santos, D. (2018, February). An Analysis of a Media-Based Approach to Teach Programming to Middle School Students. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 1005-1010). ACM.

Bagozzi, R. P., & Yi, Y. (1988). On the evaluation of structural equation models. *Journal of the Academy of Marketing Science*, *16*(1), 74-94.

Balanskat, A., & Engelhardt, K. (2014). *Computing our future: Computer programming and coding-Priorities, school curricula and initiatives across Europe*. European Schoolnet.

Barron, B., & Darling-Hammond, L. (2008). Teaching for Meaningful Learning: A Review of Research on Inquiry-Based and Cooperative Learning. Book Excerpt. *George Lucas Educational Foundation*.

Barron, Brigid, Sarah E. Walter, Caitlin Kennedy Martin, and Colin Schatz. (2010). Predictors of creative computing participation and profiles of experience in two Silicon Valley middle schools. *Computers& Education, 54*(1), 178-189.

Bers, M. U. (2008). Using robotic manipulatives to develop technological fluency in early childhood. *Contemporary Policy on Science Technology Early Child Education*, 105-225.

Bergin, S., & Reilly, R. (2005, February). Programming: factors that influence success. In *ACM SIGCSE Bulletin* (Vol. 37, No. 1, pp. 411-415). ACM.

Beyer, S. (2014). Why are women underrepresented in Computer Science? Gender differences in stereotypes, self-efficacy, values, and interests and predictors of future CS course-taking and grades. *Computer Science Education, 24*(2-3), 153-192.

Biggs, J., Kember, D., & Leung, D. Y. (2001). The revised two-factor study process questionnaire: R-SPQ-2F. *British Journal of Educational Psychology*, *71*(1), 133-149.

Blanchard, J. J., Gardner-McCune, C., & Anthony, L. (2018, February). How Perceptions of Programming Differ in Children with and without Prior Experience. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education.* (pp. 1099-1099). ACM.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association,* Vancouver, Canada (pp. 1-25).

Bureau of Labor Statistics. (2015b). Labor force statistics from the Current Population Survey [Data file]. Retrieved from http://www.bls.gov/cps

Bruckman, A., & Resnick, M. (1995). The MediaMOO project: Constructionism and professional community. *Convergence*, *1*(1), 94-109.

Computer science is for all students! (2018, May, 18). Retrieved from: https://www.nsf.gov/news/special_reports/csed/

Computer science for all. (2016, January, 30). Retrieved from: https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

Computer science mania sweeps the US. (2018, May, 18). Retrieved from https://medium.com/@codeorg/computer-science-mania-sweeps-the-us-b38e6b7518c3

Dawson, J. Q., Allen, M., Campbell, A., & Valair, A. (2018, February). Designing an Introductory Programming Course to Improve Non-Majors' Experiences. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 26-31). ACM.

Dede, C. (2005). Planning for neomillennial learning styles. Educause Quarterly, 28(1), 7-12.

Diamantopoulos, A., Siguaw, J. A., & Siguaw, J. A. (2000). Introducing LISREL: A guide for the uninitiated. *Sage*.

Driscoll, M. Psychology of learning for instruction. 1994. *Needham, MA: Allyn and Bacon Google Scholar*.

Enders, C. K., & Bandalos, D. L. (2001). The relative performance of full information maximum likelihood estimation for missing data in structural equation models. *Structural Equation Modeling, 8*(3), 430-457.

Etikan, I., Musa, S. A., & Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American Journal of Theoretical and Applied Statistics*, *5*(1), 1-4.

Goode, J., & Margolis, J. (2011). Exploring computer science: A case study of school reform. *ACM Transactions on Computing Education (TOCE)*, *11*(2), 12.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38-43.

Grover, S., & Pea, R. (2013, March). Using a discourse-intensive pedagogy and android's app inventor for introducing computational concepts to middle school students. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (pp. 723-728). ACM.

Grover, S. (2014). Foundations for advancing computational thinking: Balanced designs for deeper learning in an online computer science course for middle school students.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, *25*(2), 199-237.

Grover, S., Pea, R., & Cooper, S. (2016, February). Factors influencing computer science learning in middle school. *In Proceedings of the 47th ACM technical symposium on computing science education* (pp.552-557). ACM.

Guzdial, M. (2009, October, 9). Question Everything: How We Teach Intro CS is Wrong [Web log post]. Retrieved May 8, 2017, from https://computinged.wordpress.com/2009/10/02/questioneverything-how we-teach intro-cs-is-wrong/

Guzdial (2018). Analysis of 2017 AP CS exam participation from Barbara Ericson. RetrievedMarch, 14, 2018 from: https://computinged.wordpress.com/2018/01/08/analysis-of 2017-ap-cs-exam-participation-from-barbara-ericson

Haungs, M., Clark, C., Clements, J., & Janzen, D. (2012, February). Improving first-years success and retention through interest-based CS0 courses. *In Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (pp. 589-594). ACM.

Hatley, L. A. D. (2016). Communal learning versus individual learning: An exploratory convergent parallel mixed-method study to describe how young African American novice programmers learn computational thinking skills in an informal learning environment (Doctoral dissertation, George Mason University).

Hooper, D., Coughlan, J., & Mullen, M. (2008). Structural equation modelling: Guidelines for determining model fit. *Articles*, 2.

Hu, L. T., & Bentler, P. M. (1998). Fit indices in covariance structure modeling: Sensitivity to under parameterized model misspecification. *Psychological Methods*, *3*(4), 424.

Hushman, C. J., & Marley, S. C. (2015). Guided instruction improves elementary student learning and self-efficacy in science. *The Journal of Educational Research, 108*(5), 371-381.

Increasing Access to High-Quality Science, Technology, Engineering, and Mathematics (STEM) Education. (2017, September, 25). Retrieved from: https://www.whitehouse.gov/articles/president-trump-signs-presidential-memo-increase-access-stem-computer-science-education/

Israel, M., Wherfel, Q. M., Shehab, S., Melvin, O., & Lash, T. (2017, August). Describing Elementary Students' Interactions in K-5 Puzzle-based Computer Science Environments using the Collaborative Computing Observation Instrument (C-COI). In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 110-117). ACM.

Kafai, Y., Searle, K., Martinez, C., & Brayboy, B. (2014, March). Ethnocomputing with electronic textiles: culturally responsive open design to broaden participation in computing in American Indian youth and communities. *In Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 241-246). ACM.

Kafai, Y. B., & Burke, Q. (2015). Constructionist gaming: Understanding the benefits of making games for learning. *Educational Psychologist, 50*(4), 313-334.

Kafai, Y. B., & Burke, Q. (2017). Computational Participation: Teaching Kids to Create and Connect Through Code. In Emerging Research, Practice, and Policy on Computational Thinking (pp. 393-405). Springer, Cham.

Kenny, D. A. (2012). Multiple latent variable models: Confirmatory factor analysis. Retrieved from http://davidakenny.net/cm/mfactor.htm

Kenny, D. A., Kaniskan, B., & McCoach, D. B. (2015). The performance of RMSEA in models with small degrees of freedom. *Sociological Methods & Research, 44*(3), 486-507.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, *41*(2), 75-86.

K12CS. (2016). K12 Computer Science Framework. Retrieved May 8, 2017, from http://k12cs.org/wp-content/uploads/2016/09/K-12-Computer-Science-Framework.pdf

Lachney, M. (2018). Computational communities: African-American cultural capital in computer science education. *Computer Science Education*, 1-22.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., & Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads, 2*(1), 32-37.

Lent, R. W., Sheu, H. B., Gloster, C. S., & Wilkins, G. (2010). Longitudinal test of the social cognitive model of choice in engineering students at historically Black universities. *Journal of Vocational Behavior, 76*(3), 387- 394.

Liu, C.-C., Chen, W.-C., Lin, H.-M., & Huang, Y.-Y. (2017). A remix-oriented approach to promoting student engagement in a long-term participatory learning program. *Computers & Education*, *110*, 1–15. https://doi.org/10.1016/j.compedu.2017.03.002

Lishinski, A., Yadav, A., Good, J., & Enbody, R. J. (2016). Learning to Program: Gender Differences and Interactive Effects of Students' Motivation, Goals, and Self-Efficacy on Performance. In *ICER* (pp.211-220).

Little, J. R. (1998), "Missing data," in Encyclopedia of Biostatistics, eds. P. Armitage and T. Colton, Chichester, UK: John Wiley & Sons.

Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2008). *Stuck in the shallow end: Education, race, and computing*. MIT Press.

Margolis, J., Goode, J., & Bernier, D. (2011). The Need for Computer Science. *Educational Leadership*, *68*(5), 68-72.

Moore, D. S., Notz, W. I, & Flinger, M. A. (2013). The basic practice of statistics (6th ed.). New York, NY: W. H. Freeman and Company.

Muthén, L. K., & Muthén, B. O. BO 1998-2012. Mplus user's guide, 7.

National Science Foundation. (2017). Women, Minorities, and Persons with Disabilities Science and Engineering. Retrieved May 8, 2017, from https://www.nsf.gov/statistics/2017/nsf17310/static/downloads/nsf17310 digest.pdf

Newstetter, W. C., McCracken, W. M., & Eastman, C. M. (2001). Design knowing and learning: Cognition in design education. *Elsevier Science Limited.*

Papert, S. (1991). Situating constructionism. In S. Papert & I. Harel (Eds.), *Constructionism* (pp. 1–11). Norwood, NJ: Ablex.

Piaget, J. (1952). *The origins of intelligence in children*. New York: Basic Books.

Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2016, November). Does computational thinking correlate with personality? the non-cognitive side of computational thinking. *In Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality* (pp. 51-58). ACM.

Robins, A. (2015). The ongoing challenges of computer science education research. *Computer Science Education*, *25*(2):115-119.

Scott, K. A., Sheridan, K. M., & Clark, K. (2015). Culturally responsive computing: a theory revisited. *Learning, Media and Technology*, *40*(4), 412-436.

Sherman, M., & Martin, F. (2015). The assessment of mobile computational thinking. *Journal of Computing Sciences in Colleges*, *30*(6), 53-59.

Watson, C., Li, F. W., & Godwin, J. L. (2014, March). No tests required: comparing traditional and dynamic predictors of programming success. *In Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 469-474). ACM.

Weintrop, D. (2016). Modality matters: understanding the design of introductory programming environments. *Journal of Computing Sciences in Colleges, 32*(1), 6-6.

Wiedenbeck, S. (2005, October). Factors affecting the success of non-majors in learning to program. *In Proceedings of the first international workshop on Computing education research* (pp. 13-24). ACM.

Workshop on the Growth of Computer Science Undergraduate Enrollments. (2016, August, 14). Retrieved from: http://sites.nationalacademies.org/cstb/currentprojects/cstb_173432

Yadin, A. (2011). Reducing the dropout rate in an introductory programming course. *ACM Inroads*, *2*(4), 71-76.

Zikmund, William G. (2000). *Business research methods* (6th ed). Fort Worth: Harcourt College Publishers.