# On the Domain-Specificity of Mindsets: The Relationship Between Aptitude Beliefs and Programming Practice

Michael James Scott and Gheorghita Ghinea

***Abstract*—Deliberate practice is important in many areas, including learning to program computers. However, beliefs about the nature of personal traits, known as *mindsets*, can have a profound impact on such practice. Previous research has shown that those with a *fixed mindset* believe their traits cannot change and tend to reduce their level of practice when they encounter difficulty. In contrast, those with the *growth mindset* believe their traits are flexible and tend to maintain regular practice despite the level of difficulty. However, focusing on mindset as a single construct focused on intelligence may not be appropriate in the field of computer programming. Exploring this notion, a self-belief survey was distributed to undergraduate software engineering students. It was revealed that beliefs about *intelligence* and *programming aptitude* formed two distinct constructs. Furthermore, the mindset for programming aptitude had greater utility in predicting software development practice and a follow-up survey showed that it became more fixed throughout instruction. Thus, educators should consider the role of programming-specific beliefs in the design and evaluation of introductory courses in software engineering. Particularly, the need to situate and contextualize the growth messages that motivate students who experience early setbacks.**

***Index Terms*—Self-Theories, Implicit Beliefs, Programming, Practice, Domain-Specific, Mindsets, Dweck.**

## I. Introduction

MANY struggle to learn programming [18]. Introductory courses, in particular, have a history of poor outcomes at the tertiary level. A recent study showed that many learners fail to grasp the fundamentals after such a course [16]. Thus, it is important to explore how students learn successfully.

Addressing the cognitive-affective barriers which reduce the deliberate practice that students engage in offers one line of enquiry [30]. This is important because maintaining an ongoing, reflexive, and self-regulated learning process is critical to the acquisition of expertise [13]. It has been said that at least ten years of such practice is needed to develop substantial proficiency in software engineering [13, 34].

Consequently, educators often situate high levels of scaffolding and formative feedback within the introductory programming lab [19]. Despite such efforts, however, many beginners do not practice regularly. Often, such students claim they experience apprehension and discomfort when they attempt to do so [28]. These emotional responses can prompt students to stop working on difficult assignments [22] and it has been reported that such affective factors worsen over a course of programming instruction [24].

Nevertheless, not all students react this way when they encounter problems. For example, some perceive compilation errors as a challenge to be overcome rather than as an indication of failure [26]. Among the potential reasons for such conflicting perspectives is the different ways in which students reflect upon their learning [21, 22]. These differences may correspond to students' self-beliefs [26], presenting an opportunity for educators to nurture particular mindsets.

## II. Mindsets: Are They Domain-Specific?

According to the self-theories proposed by Dweck [12], individuals hold beliefs about the nature of their personal traits, referred to as their *mindset*, which can be classified into one of two core beliefs. Those with the *fixed mindset* believe their traits are an entity that cannot be changed. Conversely, those with the *growth mindset* believe their traits are flexible and can be enhanced through effort.

These beliefs have implications for the way that students engage in self-regulated learning. This is because the learning strategies that students apply depend on whether they believe such strategies are necessary for learning and are effective at addressing problems [11]. As a result, those with a fixed mindset tend to adopt a helpless response when they encounter difficulty. In contrast, those with a growth mindset tend to persevere, adopting a mastery-orientated strategy [8, 35].

In order to nurture a growth mindset, educators embed growth messages such as "the brain is like a muscle, it develops through exercise" into their teaching practice. However, this advice is often framed in terms of intelligence [21, 26, 32]. Do students generalize such messages? The human mind can be conceived in terms of *multiple intelligences* [14] and self-theories have been adapted for areas as varied as: shyness [2]; math-ability [15]; and

Michael James Scott and Gheorghita Ghinea are members of the People & Interactivity Group within the School of Information Systems, Computing & Mathematics at Brunel University (e-mail: michael.scott@brunel.ac.uk).

willpower [20]. Therefore, it is conceivable that students do not associate their programming ability with a general sense of intelligence, but rather to a sense of programming aptitude.

Programming has been described as a discipline that presents “radical novelties” to beginners [9]. This is because new students often need to adapt their way of thinking to accommodate the abstract and intangible concepts that are applied in program creation [5, 30] as such thinking is seldom developed prior to the first programming course [18]. Hence, the discipline can feel distinct, potentially promoting a separate mindset for programming aptitude.

This has implications for the design and evaluation of teaching practice. A “saying is believing” exercise required students to “describe a time when (they) learned something other than programming (...) but with practice and perseverance (they) were able to succeed” [31, p. 176]. However, this was shown to lack practical impact. Perhaps this may be the case because students can hold a separate mindset for programming. In such cases, reflecting on past success in another discipline may not succeed.

Another intervention attempted a rich combination of mindset-informed training and feedback practices [7]. This was shown to have some success, but only for students who also received a programming-specific crib-sheet which contextually reinforced the growth belief. This could indicate an advantage in applying educational practices that are designed to change a more specific mindset, as opposed to a general mindset for intelligence [c.f. 35]. However, only a single measure was used in the study.

Following this line of reasoning, prior work [29] has shown that an adapted mindset scale formed two distinct subscales: items about programming aptitude and items about general intelligence. However, a significant correlation was found between scores on these subscales, raising several research questions (RQs) about the implications of separate mindsets for teaching practice in the software engineering context:

RQ1. Can students have a mindset for programming aptitude that is substantially different to their mindset for general intelligence?

RQ2. Does the mindset for programming aptitude have more utility for predicting programming practice compared to the mindset for general intelligence?

RQ3. Does the mindset for programming aptitude change differently to the mindset for general intelligence over a period of programming instruction?

## III. Hypotheses

To explore the relative merits of modeling separate mindsets, there is a need to establish a clear difference between them. The first research question examines two hypotheses: firstly, that a model with programming aptitude mindset and intelligence mindset as two distinct, but slightly correlated factors ($H_1$), demonstrates good fit to observed data [29]; secondly, for the notion of separate mindsets to have utility for educators, the classification of each mindset (being either fixed or growth) should not have a high level of consistency ($H_2$). The second research question then explores the impact of each mindset on programming practice behavior. It is hypothesized that both programming aptitude mindset and mindset for intelligence ($H_3$ and $H_4$) are related to programming practice behavior. Given their relation to resilience [8, 35], each relationship will be moderated by early performance ($H_5$ and $H_6$), such that those achieving high grades will not be as strongly influenced by their mindset. However, each will have a different level of explanatory power on programming practice behavior ($H_7$).

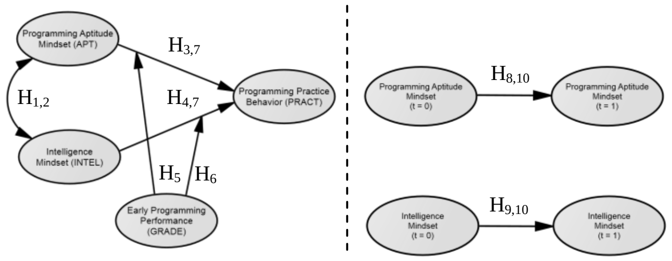
Programming Aptitude Mindset (APT)
Intelligence Mindset (INTEL)
Programming Practice Behavior (PRACT)
Early Programming Performance (GRADE)
$H_{1,2}$
$H_{3,7}$
$H_{4,7}$
$H_5$
$H_6$
Programming Aptitude Mindset (t = 0)
Programming Aptitude Mindset (t = 1)
$H_{8,10}$
Intelligence Mindset (t = 0)
Intelligence Mindset (t = 1)
$H_{9,10}$

Fig. 1. The Impacts of Mindset for Programming Aptitude (APT) and Mindset for Intelligence (INTEL) on Programming Practice Behavior (PRACT), as moderated by Early Programming Performance (GRADE) (*Left)*, Alongside Change in Each Mindset Across an 8-Week Period of Instruction (*Right)*.

The third research question investigates change in mindset over time. As there could be elements of programming instruction that induce a fixed mindset [7, 26], it is hypothesized that mindset towards programming aptitude will become more fixed over a period of programming instruction ($H_8$). Mindset towards intelligence may also change ($H_9$), but less so than programming aptitude ($H_{10}$).

## IV. Data Collection

A two-wave survey was conducted in 2012-13 to examine these hypotheses. Participants were recruited from two core programming modules at the authors' institution. The study was promoted via: pre-registered email; institutional email; notices on BlackBoard Learn; and through a course-related Facebook Group.

The questionnaires were distributed using SurveyMonkey and were available for 11 days across the 8th and 16th week of the semester, respectively. Participation was voluntary. In order to identify programming assessments corresponding to each respondent, student identification numbers were either obfuscated and encoded into hyperlinks or reported. The sampling frame consisted of 296 first and second year undergraduate students on programming modules within the authors' institution. To be eligible, students had to be at least 18 years of age and had to have submitted their first three lab assignments, the deadlines for which were prior to the date the survey was conducted. There were 73 students who completed all of the items in the first wave of the survey. Thus, the initial response rate was 24%. However, there was some attrition between the first and second wave of the survey, with only 63 students responding to both. Thus, the attrition rate was 14%.

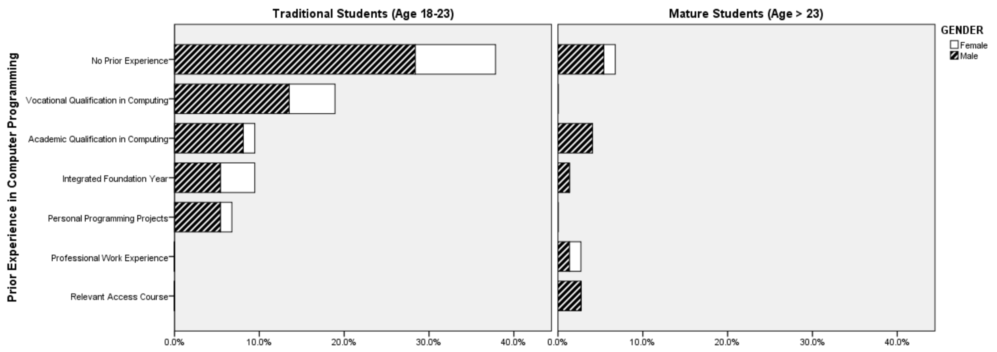

Fig. 2. A summary of the age, gender and prior programming experience of the survey respondents.

## V. Participant Characteristics

Participants were first and second year undergraduate students following the sequential pathway for "Computer Science (Software Engineering)"[1]. The descriptive statistics, summarized in Figure 2, show that approximately 24.3% of the respondents were female, while the average age was 20 years ($\bar{x}$ = 20.48, $\sigma$ = 2.42, max = 30), with 17.6% of respondents being over the age of 23 at entry.

As the response rate was low and the early programming scores for the sample indicated that many were performing at a high merit level ($\bar{x}$ = 6.61, $\sigma$ = 1.71, max = 9.00), there was concern about response bias. However, performance did not significantly differ to the cohort ($\bar{x}$ = 6.35, $\sigma$ = 1.58, $t$[72] = 1.291, $p$ = .201). Furthermore, the proportion of mature (age > 23) ($\chi^2$ = 2.647, $p$ = .103) and female students ($\chi^2$ = 1.372, $p$ = .241) was typical of the cohort.

Admission to the pathway required at least 300 UCAS Points[2] (University & College Admission System Points), with a strong preference for STEM subjects (science, technology, engineering, and mathematics). Prior programming experience was not required (44.6%). However, students without a relevant STEM qualification, or the required points, could opt to pursue a relevant foundation course (9.6%).

In the first year, students on the pathway would attend an "Introductory Java Programming" module in order to learn object-orientated design and the fundamental constructs of the Java language. This was conducted through a sequence of laboratory-based assignments and a group project, where students would program robots to complete short scripted tasks. For example: maze navigation; obstacle avoidance; or communication in Morse Code.

In the following year, students explored algorithms and data structures as part of an "Algorithms and their Applications" module. This involved the implementation and analysis of classic algorithms (e.g. sorting, searching, graph traversal, meta-heuristics) as a series of lab-based tasks. As before, this was paired with a group-based Android development project.

[1] The course description is available here: www.brunel.ac.uk/courses/undergraduate/computer-science-software-engineering-bsc

[2] To convert many international qualifications and grades to UCAS Points, refer to: www.ucas.com/how-it-all-works/explore-your-options/entry-requirements/tariff-tables

## VI. Measurement

The questionnaire measured three latent variables: mindset for intelligence (INTEL); mindset for programming aptitude (APT); and programming practice behavior (PRACT). Common factor analysis techniques were used to generate these scores, rather than principle components analysis, to explore how the underlying structure of items' shared variance reflected the latent variables of interest (see [1] for a discussion). Early programming performance (GRADE) was measured using the first three assignments in each module.

### A. *Mindset towards Intelligence (INTEL)*

To measure mindset for intelligence, items were drawn from Dweck's mindset scale [12]. Five items were used, including three statements that endorsed the fixed belief, such as "my intelligence is something about me that I can't change very much" and two statements that endorsed the growth belief, such as "I can always substantially change how intelligent I am". These were presented as a 7-point Likert Scale, ranging from strongly disagree to strongly agree. The order in which items were displayed was randomised alongside items measuring mindset for programming aptitude. Composite scores were generated using a regression method based on the factor score matrix generated by a maximum-likelihood analysis. A high score indicated a fixed belief.

### B. *Mindset towards Programming Aptitude (APT)*

To measure mindset for programming aptitude, items were also drawn from Dweck's mindset scale [12]. However. these were adapted to the programming context. Five items were used, including three statements endorsing the fixed belief. For example: "I have a fixed level of programming aptitude, and not much can be done to change it". The remaining two items endorsed the growth belief. For example: "I believe I am able to achieve a high level of programming aptitude, with enough practice". The items were presented as a 7-point Likert scale, with responses ranging from strongly disagree to strongly agree. The items were presented randomly alongside items measuring mindset for intelligence. Composite scores were generated using a regression method based on a factor score matrix produced by a maximum-likelihood analysis. A high score on this scale indicates a fixed belief.

### C. *Regularity of Programming Practice (PRACT)*

A self-report measure of programming practice was created for this survey using a 7-point and a 4-point item. These were presented as Guttman-type items, questioning "in a typical week of study I find myself writing code {during the closed-labs / at least {1-5} day(s) a week / every day}" and "in a typical session I concentrate on programming for {up to 30 minutes / at least 30 minutes / at least one hour / at least two hours}". Thus, providing an indication of frequency of practice and the typical duration of each practice session.

Responses to these items were parceled into a single composite score using principal axis factoring. Note, as a retrospective self-report measure, this should not be interpreted as actual practice. Caution should be exercised due to the potential for self-report biases [10].

### D. *Early Programming Performance (GRADE)*

As the core programming modules used the same assessment structure, early programming performance was measured using existing assessment data. Assignments were assessed as code reviews by a team of Ph.D. students covering the modules; typically, with good consistency (ICC = 0.73, 6 submissions). Grades reflected the functional coherence of solutions, the presence of common pitfalls, and a judgment on quality according to a rubric. They were recorded as 1 (pass), 2 (merit), and 3 (distinction). The results of the first three assessments were added together to form a composite score.

## VII. Data Analysis

The data was analyzed using PASW 18.0.3 and AMOS 21.0.0 for Windows. All cases were included. Cases with missing data were removed list-wise. All reported *p*-values are two-tailed with significance determined at the .05 level.

### A. *Replication of the Two-Mindsets Factor Structure (RQ1)*

As with the previous study [29], a maximum-likelihood factor analysis showed that a two factor model had greater fit ($\chi^2$ = 43.094, *df* = 34, *p* = .136) than a single factor model ($\chi^2$ = 69.619, *df* = 35, *p* = .000). Furthermore, the items used to measure mindset towards intelligence (α= .73) and mindset towards programming aptitude (α = .61) demonstrated adequate reliability. Fit indices are summarized in Table 1:

TABLE 1. FIT INDICES & CRITERIA FOR TWO-FACTOR MODEL

| Fit Indices | 1-Factor Model | 2-Factor Model | Adequate Fit Criteria [18] |
|---|---|---|---|
| SRMR | .108 | .078 | < .08 |
| CFI | .728 | .928 | > .90 |
| RMSEA | .117 | .061 | < .08 |
| Bollen-Stein *p* | .015 | .313 | > .05 |

*Note:* SRMR: Standardized Root Mean Square Residual; CFI: Comparative Fit Index; RMSEA: Root Mean Square Error of Approximation; N = 73

### B. *Consistency Between Different Mindsets (RQ1)*

Participants were classified using a two-step clustering procedure that was applied separately to APT and INTEL. Each showed the expected two-cluster solutions, based on Log-likelihood distance and Bayesian Information Criterion. The average silhouette coefficient was used to evaluate the clustering solutions. This yielded values greater than 0.7 for both analyses, indicating that the solutions were "good". The results are shown in Table 2 below:

TABLE 2. MINDSET CLASSIFICATION FOR INDIVIDUAL STUDENTS

| | Fixed Intelligence | Growth Intelligence | κ | *p* |
|---|---|---|---|---|
| Fixed Programming Aptitude | 10 | 13 | | |
| Growth Programming Aptitude | 11 | 39 | .220 | .060 |

*Note: κ:* Cohen's kappa; N = 73; Agreement = 67.1%

The correlation between factor scores was significant (*r* = .248, *p* = .034). However, the level of agreement between each classification scheme, based on the kappa statistic, indicated only "fair agreement" (*p* = .060) [23]. It can be seen that 23 students were classified as fixed APT (31.5%), while 21 students were fixed INTEL (28.7%). Inconsistency occurred in 24 cases (32.9%), where students held different beliefs for the two domains. This was most prominent for those with fixed APT, where 13 of the 23 cases maintained growth INTEL (56.5%). However, 11 of the 50 cases with a growth APT also had inconsistent beliefs (28.8%).

### C. *Impact of Each Mindset on Practice Behavior (RQ2)*

Two linear regression analyses compared the independent impact of APT and INTEL on PRACT. Assumptions of residual normality, independence, and homoscedasticity were verified prior to each analysis. The model exploring APT was significant (*p* = .003) and is described below in Table 3.

TABLE 3. PROGRAMMING APTITUDE MINDSET REGRESSION MODEL

| Construct | β | $\sigma_{\bar{x}}$ | t | *p* |
|---|---|---|---|---|
| Programming Aptitude Mindset (*APT*) | -.249 | .109 | -2.225 | .025 |
| *APT*GRADE* | .245 | .101 | 2.254 | .027 |
| Early Programming Performance (*GRADE*) | .248 | .108 | 2.281 | .026 |

*Note:* Adjusted $R^2$ = .141; N = 73; *F*[3,70] = 4.985, *p* = .003

The relationship, illustrated in Figure 3, reveals that those with fixed APT and low GRADE tended to practice less than their peers. However, students with high GRADE were not as strongly influenced by their APT.

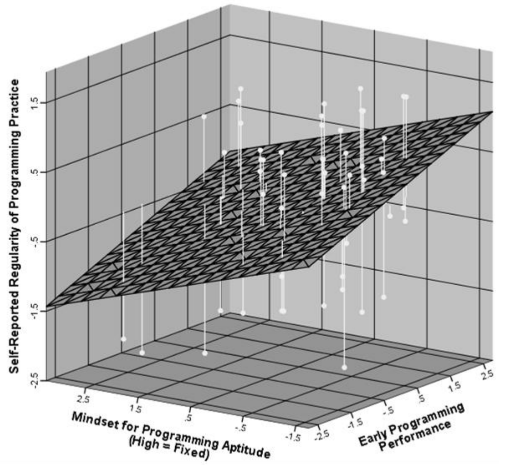

Fig. 3. Illustrating the influence of programming aptitude mindset and performance on early assignments on students' programming practice.

This interaction, however, was not found in the model exploring INTEL, shown in Table 4:

TABLE 4. INTELLIGENCE MINDSET REGRESSION MODEL

| Construct | β | $\sigma_{\bar{x}}$ | t | *p* |
|---|---|---|---|---|
| Intelligence Mindset (*INTEL*) | -.253 | .114 | -2.222 | .030 |
| *INTEL*GRADE* | .135 | .107 | 1.194 | .237 |
| Early Programming Performance (*GRADE*) | .283 | .113 | 2.490 | .015 |

*Note:* Adjusted $R^2$ = .089; N = 73; *F*[3,70] = 3.385, *p* = .023

The expected interaction with GRADE was not significant (p = .237). Nevertheless, using INTEL to predict PRACT was significant (*p* = .023). Thus, both models had utility for predicting PRACT. However, the comparison shown below in Table 5 reveals several differences:

TABLE 5. MODEL SELECTION CRITERIA

| Model | *Adjusted $R^2$* | *AIC* | *PC* | *BIC* |
|---|---|---|---|---|
| Programming Aptitude Mindset Model | .141 | -9.037 | 0.895 | -2.166 |
| Intelligence Mindset Model | .089 | -4.791 | 0.948 | 2.081 |

*Note:* AIC: Alkaike Information Criterion; PC: Prediction Criterion; BIC: Bayesian Information Criterion

It can be seen that the regression model using the mindset scores for programming aptitude explained a larger proportion of variance (Adjusted $R^2$ = .141, $\Delta R^2$ = .052). Furthermore, there was an noticeable improvement in fit ($\Delta AIC$ = 4.246, $\Delta AIC$ > 2 [4]). Thus, the data shows that the APT model has greater utility for predicting PRACT.

### D. Change in Belief for Each Mindset Over Time (RQ3)

A series of paired t-tests examined whether students' mindsets had changed between the first wave of the survey and the second wave. The results are shown in Table 6:

TABLE 6. PAIRED T-TESTS FOR CHANGES IN MINDSET

| Mindset | $\bar{x}_{w=0}$ | $\bar{x}_{w=1}$ | $\bar{x}_\Delta$ | $\sigma_\Delta$ | t | *p* | *d* |
|---|---|---|---|---|---|---|---|
| Intelligence | -0.15 | -0.09 | .059 | .508 | 0.927 | .358 | n.s |
| Programming Aptitude | -1.16 | -0.90 | .267 | .856 | 2.475 | .016 | 0.62 |

*Note:* N = 63, *df* = 62, w: survey wave (8 weeks between each wave), *d*: Cohen's d effect size

There was a non-significant decrease in mean INTEL score (*p* = .358). Thus, students' INTEL remained stable across the eight week period. However, there was a significant increase in APT score (*p* = .016). This suggests that students' beliefs towards programming aptitude had become more fixed, with "medium" effect (*d* = 0.62) [6]. In context, however, the mean difference ($\bar{x}_\Delta$ = .267) does not represent a large shift for the entire cohort. Only 30.2% of the respondents came to believe more strongly in a fixed perspective, with only 18% of cases changing distinctly from the growth belief.

### E. Summary of Adjustments for Multiple Hypothesis Testing

As multiple hypotheses were explored, *p*-values have been adjusted to control for the false discovery rate (*FDR* = .05) using the Benjamini-Hotchberg Procedure [3]. Note, $H_7$ and $H_{10}$ are not associated with a null hypothesis significance test. These adjustments are shown in Table 7.

TABLE 7. SUMMARY OF FINDINGS AND ADJUSTED p-VALUES

| RQ | $H_n$ | Hypothesis | Observation | $\tilde{p}$ | Conclusion |
|---|---|---|---|---|---|
| 1 | $H_1$ | APT ↔ INTEL | r = .248 | .048 | Reject Null |
| | $H_2$ | κ (APT ↔ INTEL) ≠ 0 | κ = .220 | .075 | *Accept* Null |
| 2 | $H_3$ | APT → PRACT | β = -.249 | .048 | Reject Null |
| | $H_4$ | INTEL → PRACT | β = -.253 | .048 | Reject Null |
| | $H_5$ | (APT * GRADE) → PRACT | β = .245 | .048 | Reject Null |
| | $H_6$ | (INTEL * GRADE) → PRACT | β = .135 | .263 | *Accept* Null |
| | $H_7$ | $\lvert AIC$ (APT) - $AIC$ (INTEL)$\rvert > 2.0$ | $\Delta AIC = 4.2$ | -- | -- |
| 3 | $H_8$ | $\bar{x}_\Delta$ (APT) ≠ 0 | $\bar{x}_\Delta$ = .267 | .048 | Reject Null |
| | $H_9$ | $\bar{x}_\Delta$ (INTEL) ≠ 0 | $\bar{x}_\Delta$ = .059 | .358 | *Accept* Null |
| | $H_{10}$ | $\lvert d$ (APT) - $d$ (INTEL)$\rvert > 0.2$ | $\Delta d$ = 0.62 | -- | -- |

*Note:* p̃: Benjamini-Hotchberg adjusted p-value.

## VIII. DISCUSSION

The literature on self-beliefs and motivation shows that mindsets can influence resilience [8, 35]. Students with growth beliefs tend to maintain practice when they encounter difficulty. Those with fixed beliefs do not. Thus, it is important that educators inspire growth beliefs as ongoing practice is important for developing expertise [13, 34]. However, mindset may not reflect a single general construct focused on intelligence. This study shows some evidence that students may develop domain-specific beliefs in the area of computer programming.

The first research question examined whether students' beliefs about their intelligence and their beliefs about their programming aptitude could be substantially different. Although a significant correlation was found, the classification schemes showed low levels of agreement. Most students with the fixed belief for programming aptitude had the growth belief for intelligence. This suggests that students can have markedly different mindsets across domains.

The second research question explored the relationships between each mindset and programming practice behavior. Although the results were modest, the regression model based on aptitude beliefs had a closer fit to the data and explained a greater proportion of the variance. Furthermore, while early performance in programming moderated the relationship between practice and aptitude beliefs, this was not found in intelligence model. These results seems to reinforce the notion that students do not associate their performance in computer programming with their sense of intelligence and suggest that the mindset towards programming aptitude could have greater utility for predicting programming practice.

The third research question asked whether beliefs changed across an eight week period of instruction. Although beliefs about intelligence did not change, it is a concern that nearly one-third of respondents came to believe more strongly in the fixed perspective of programming aptitude. The cause, in this case, is unclear. The literature suggests that many aspects of programming instruction [7, 26] and feedback style [25, 27] could have contributed to the change. However, there could be differences in source as well as sensitivity. Thus, factors that affect domain-specific beliefs should be further explored.

This study has several limitations, notably threats to external validity as students were recruited from two classes at a single institution and the number of students encountering early difficulties was low. Furthermore, the sample size constrained statistical power, so interaction effects could not be investigated. It should also be noted that the reliability of the mindset measure was marginally adequate, suggesting a need for further scale development (see [33]).

## IX. Conclusion and Future Work

This study reveals some evidence that mindset for *programming aptitude* is not only distinct from mindset about *intelligence*, but that it may also have a stronger relationship with programming practice. This suggests a discipline-specific perspective may be appropriate when extending self-theory research into the software engineering context. As such, educators should emphasize the malleability of programming skill directly by, for example, contextually situating growth messages within relevant programming materials (e.g. code review rubrics [7]). Moreover, future work should examine measures of programming aptitude mindset and further investigate mindset interventions.

## References

[1] Beavers, A.S. et al. "Practical Considerations for Using Exploratory Factor Analysis in Education Research", *Practical Assessment, Research &* Evaluation, vol. 18, no. 6, pp. 1-13, 2013.

[2] Beer, J.S. "Implicit Self-Theories of Shyness", *J. Personality & Social Psychology*, vol. 83, no. 4, pp. 1009-1024, 2002.

[3] Benjamini, Y. and Hochberg, Y. "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing", *J. Royal Statistical Society*, ser. B, vol. 57, no. 1, pp. 289-300, 1995.

[4] Burnham, K. P., and D. R. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach, 2nd Ed*. Springer-Verlag, New York, 2002.

[5] Caspersen, M.E. and Bennedset, J. "Instructional Design of a Programming Course - A Learning Theoretic Approach ". In *Proc. 3rd International Workshop on Computing Education Research*, Atlanta, GA, 2007, pp. 111-122.

[6] Cohen, J. "A Power Primer", *Psychological Bulletin*, vol. 112, no. 1, pp. 155-159, 1992.

[7] Cutts, Q., Cutts, E., Draper, S., O'Donnell, P. and Saffrey, P. "Manipulating Mindset to Positively Influence Introductory Programming Performance". In *Proc. 41st ACM Tech. Symp. on Computer Science Education*, Milwaukee, WI, 2010, pp. 431-435.

[8] Diener, C.I. and Dweck, C.S. "An Analysis of Learned Helplessness: Continuous Changes in Performance, Strategy and Achievement Cognitions Following Failure", *J. Personality & Social Psychology*, vol. 36, no. 5, pp. 451-462, 1978.

[9] Dijkstra, E.W. "A Debate on Teaching Computing Science: On the Cruelty of Really Teaching Computing Science", *Communications of the ACM*, vol. 32, no. 12, pp. 1398-1404, 1989.

[10] Donaldson, S.I. and Grant-Vallone, E.J. "Understanding Self-Report Bias in Organizational Behavior Research", *J. Business & Psychology*, vol. 17, no. 2, pp. 245-260, 2002.

[11] Dweck, C.S. and Master, A. "Self-Theories Motivate Self-Regulated Learning" in *Motivation & Self-Regulated Learning: Theory, Research & Applications*, D.H. Schunk and B. Zimmerman, Eds. Lawrence Erlabaum, New York, NY, 2008, pp. 31-51.

[12] Dweck, C.S. *Self-Theories: Their Role in Motivation, Personality, and Development*. Psychology Press, Philadelphia, PA, 1999.

[13] Ericsson, K.A., Krampe, R. and Tesch-Romer, C. "The Role of Deliberate Practice in the Acquisition of Expert Performance", *Psychological Review*, vol. 100, pp. 393-394, 1993.

[14] Gardner, H.E. *Multiple Intelligences: New Horizons in Theory and Practice*. Basic Books, New York, NY, USA, 2006.

[15] Good, C., Rattan, A. and Dweck, C.S. "Why Do Women Opt Out? Sense of Belonging and Women's Representation in Mathematics". *J. Personality & Social Psychology*, vol. 102, no. 4, pp. 700-717, 2012.

[16] Guzdial, M. "From Science to Engineering", *Communications of the ACM*, vol. 54, no. 2, pp. 37-39, 2011.

[17] Hair, J., Black, B., Babin, B. and Anderson, R. *Multivariate Data Analysis, 7th ed.* Psychology Press, NJ, USA, 2009.

[18] Jenkins, T. "On the Difficulty of Learning to Program" [Online]. In *Proc. 3rd Ann. Conf. of the HEA Learning and Teaching Support Network: Centre for Information and Computer Sciences*, Loughborough, UK, 2002. Available: http://goo.gl/FZsib.

[19] Jenkins, T. "Teaching Programming - A Journey from Teacher to Motivator" [Online]. In *Proc. 2nd Ann. Conf. of the HEA Learning and Teaching Support Network: Centre for Information and Computer Sciences*, London, UK, 2001. Available: http://goo.gl/jZH3p.

[20] Job, V., Dweck, C.S. and Walton, G.M. "Ego Depletion - Is It All In Your Head? Implicit Theories About Willpower Affect Self-Regulation", *Psychological Sci.*, vol. 21, no. 11, pp. 1686-1693, 2010.

[21] Kinnunen, P. and Beth, S. "My Program is OK – Am I? Computing Freshman's Experience of Doing Programming Assignments", *Computer Science Education*, vol. 22, no. 1, pp. 1 – 28, 2012.

[22] Kinnunen, P. and Simon, B. "Experiencing Programming Assignments in CS1: The Emotional Toll". In *Proc. 6th Int. Workshop on Computing Education Research*, Aarhus, Denmark, 2010, pp. 77-86.

[23] Landis, J. and Koch G. "The Measurement of Observer Agreement for Categorical Data", *Biometrics*, vol. 33, no. 1, pp. 159-174, 1977.

[24] McKinney, D. and Denton, L.F. "Houston, We have a Problem: There's a Leak in the CS1 Affective Oxygen Tank", *ACM SIGCSE Bulletin*, vol. 36, no. 1, pp. 236-239, 2004.

[25] Mueller, C.M. and Dweck, C.S. "Praise for Intelligence Can Undermine Children's Performance", *J. Personality & Social Psychology*, vol. 75, no. 1, pp. 33-52, 1998.

[26] Murphy, L. and Thomas, L. "Dangers of a Fixed Mindset: Implications of Self-Theories Research for Computer Science Education", *ACM SIGCSE Bulletin*, vol. 40, no. 3, pp. 271-275, 2008.

[27] Rattan, A., Good, C. and Dweck, C.S. "It's Okay - Not Everyone Can Be Good at Math": Instructors with an Entity Theory Comfort (and Demotivate) Students", *J. Experimental Social Psychology*, vol. 48, no. 3, pp. 731-737, 2012.

[28] Rogerson, C. and Scott, E. "The Fear Factor: How it Affects Students Learning to Program in a Tertiary Environment", *J. Information Technology Education*, vol. 9, no. 1, pp. 147-171, 2010.

[29] Scott, M.J. and Ghinea, G. "Implicit Theories of Programming Aptitude as a Barrier to Learning to Code: Are They Distinct from Intelligence?". In *Proc. 18th Ann. ACM Conf. on Innovation and Technology in Computer Science Education*, Kent, UK, 2013.

[30] Scott, M.J. and Ghinea, G. "Educating Programmers: A Reflection on Barriers to Deliberate Practice". In *Proc. 2nd HEA Conf. on Learning and Teaching in STEM Disciplines*, Birmingham, UK, 2013, pp. 028P.

[31] Simon, B. *et al*. "Saying Isn't Necessarily Believing: Influencing Self-Theories in Computing". In *Proc. 4th Int. Workshop on Computing Education Research*, Sydney, Australia, 2008, pp. 173-184.

[32] Stump, G., Husman, J., Chung, W.-T. and Done, A. "Student Beliefs about Intelligence: Relationship to Learning". In *Proc. IEEE Frontiers in Education Conf.*, San Antonio, TX, 2009, pp. T4F-1.

[33] Tew, A.E. and Dorn, B. "The Case for Validated Tools in Computing Education Research", *Computer*, vol. 46, no. 9, pp. 60-66.

[34] Winslow, L.E. "Programming Pedagogy - A Psychological Overview", *ACM SIGCSE Bulletin*, vol. 28, no. 3, pp. 17-22, 1996.

[35] Yeager, D. and Dweck, C.S. "Mindsets That Promote Resilience: When Students Believe That Personal Characteristics Can Be Developed", *Educational Psychologist*, vol. 47, no. 4, pp. 302-314, 2012.