# Implicit Theories and Self-efficacy in an Introductory Programming Course

F. Boray Tek \*, Kristin S. Benli\*, Ezgi Deveci<sup>†</sup>

\*Department of Computer Engineering, Isik University, Istanbul, Turkey <sup>†</sup>Departmenf of Physchology, Işık University, İstanbul, Turkey

#### Abstract

Contribution: This study examined student effort and performance in an introductory programming course with respect to student-held implicit theories and self-efficacy.

Background: Implicit theories and self-efficacy shed a light into understanding academic success, which must be considered when developing effective learning strategies for programming.

Research Questions: Are implicit theories of intelligence and programming, and programming-efficacy related to each other and student success in programming? Is it possible to predict student course performance using a subset of these constructs?

Methodology: Two consecutive surveys (N=100 and N=81) were administered to non-CS engineering students in Isik University.

Findings: Implicit theories and self-beliefs are interrelated and correlated with effort, performance, and previous failures in the course and students explain failure in programming course with "programming-aptitude is fixed" theory, and also that programming is a difficult task for themselves.

#### **Index Terms**

Implicit theories, self-efficacy, programming-efficacy, mindset for programming, programing-aptitude

### I. INTRODUCTION

Large numbers of today's university students are required to enroll in introductory programming courses and significant percentages of these students struggle, repeat, and even switch programs due to difficulties with these courses. Consensus among university programming instructors is that programming is a very difficult task with a history of poor outcomes and no quick solutions [1], [2]. University level programming courses generally have the highest dropout rates [3]. Mostly, because that students who may have positive beliefs in their abilities across general subjects are frequently challenged with radical novelty in the course [4], [5].

Another factor is that programming is an area that requires motivation and practice, and above all, abstract thinking and a deep understanding, rather than surface memorization. Though Calculus also builds on similar skills, students are more acquainted with its problems, approach, language, and practice. On the other hand, programming requires studying a new language and a new approach to solving problems. In addition, the necessity of implementing and testing solutions using a computer produce many additional problems. Hence, for freshman students, programming requires harder work and deliberate practice more than other subjects.

On the other hand, modern psychology identifies new psychological constructs which shed a light into understanding academic success, which must be considered when developing effective learning strategies [6]. Self-efficacy [7], resilience, grit [8], implicit theories (a.k.a mindset) [9] are shown to affect student motivation and effort directly. Thus, academic achievement is not only the result of *cognitive processes* -that acquire, organize, reuse formal information- but also strongly affected by *motivational* processes that is related with expectation of outcomes for self-effort, and affective processes that regulate emotional states and elicit emotional responses, e.g. stress or anxiety when faced with difficulty [7].

Bandura [7] defines perceived self-efficacy as self-belief of a person in his/her capacity of accomplishing a *specific* task at certain performance levels. Self-efficacy will determine how much effort a person will expend, how it confronts obstacles, and how resilient it will be during the course of that activity. The higher the self-efficacy the greater the effort, persistence, and resilience; which are central factors to learning success, more than previous experience [10]. Bandura emphasizes that a self-efficacy measure not specific to a task will be limited in explanatory and prediction power [11]. Thus, several self-efficacy scales were developed to measure subject-specific efficacy of students in different domains such as general academic success, learning engineering [12], computer-use [13], mathematics [14], and programming [15].

On the other hand, several studies have incorporated Sherer's [16] generalized self-efficacy scale to measure a person's self-belief of capacity to perform any novel or difficult tasks. Though it is not specific to a task, generalized self-efficacy scale is extensively used; validated and translated into 33 languages [17]) in clinical practice and behavioral analysis. However, Bandura does not support a task-independent efficacy scale, he states that self-efficacy in one field can affect another -even unrelated- field. Moreover, the degree of specificity in one context can be very different than another. In other words, a task can always be analyzed in different specificity levels, hence there can be two different self-efficacy scales which are taskspecific, but one more general than the other. Thus, as Sherer [16] and later Schwarzer [18] conceptualized, a person can have a global belief in his/her capacity and performance in any novel or challenging task. When it comes to succeeding in a

Corresponding author: F. Boray Tek (email: boray.tek@isikun.edu.tr).

2

foundational programming course, the question becomes whether a domain specific or the generalized self-efficacy measure is explanatory. Hence, the first question of the current study is whether the generalized self-efficacy or programming-efficacy is more explanatory and predictive for student performance in an introductory programming course, and also whether two measurements are inter-related, or not?

Along similar lines are the implicit theories [9], [19] that people hold about skills and abilities, and that play a role in how individuals explain the challenges that they are faced with in any learning situation. The implicit theories identified by Dweck and her colleagues are incremental and entity theories – more commonly referred to as fixed and growth mindsets. Entity theorists tend to believe that intellect is fixed. Their goals are more performance-oriented, demonstrating competence and mastery. Exerting effort, perhaps through remediation, is considered a sign of weakness. As a result, entity theorists are more likely withdraw effort or procrastinate which will provide them excuses for underperformance.

Conversely, according to Dweck, incremental theorists are more oriented toward goals that reflect a concern with skill acquisition. A significant difference between these two approaches is that incremental theorists believe that intellect is not fixed and that increased effort leads to increased learning. Thus, they seek alternative paths – including remediation – to achieve the end-goal of acquiring knowledge or a skill. In other words, students with a growth mindset try to adapt themselves to radical novelty that they may sense in a programming course [20], which may also suggest that they have increased self-efficacy and resilience. In contrast, students with fixed mindsets have been shown to spend less time delving deeply into instructor feedback that may point them toward success in the course [21]. Moreover, whereas a growth mindset may be strengthened, studies have reported increases in a fixed mindset over time as well e.g. [5], [2]. A student who believes that programming is a special talent, which may or not be linked to general intelligence, may easily feel a setback by the novelty and difficulty of programming. Dweck also identifies that implicit theories or mindset of a person can change from domain to domain. For example, a person can believe that English can be improved by studying hard, whilst believing mathematics or arts is only possible with a special talent that comes from genes. Therefore, mindset measurement scales were also introduced for different domains, such as programming [5] [20], mathematics [22] and even politics [9].

In applying these concepts to the introductory programming context, the notions of fixed and growth mindsets, along with self-efficacy may be instrumental in helping to understand variation in student persistence and success in foundational courses. Connections have been suggested between other subjects (e.g. mathematics and science) and success in programming [23], as well as between academic self-efficacy and attitudes toward programming [24], [25]. Some studies have shown that students may have high general self-efficacy or growth mindset, but they may struggle with domain-specific efficacy or mindset, such as programming [26], [27], [20], [2].

Ramalingham et al. [15] established a mutual relationship between programming-efficacy and performance in a programming course. Their work was adapted and applied to Turkish engineering students in two different descriptive studies [27], [28] which measured students' programming-efficacy in a programming course based on C++ and Java, respectively. Scott and Ghinea [2] have identified the connection between success and failure in introductory programming courses and students' implicit theories of programming-aptitude.

The field examined in the present study is the intersection of implicit theories and self-efficacies, and their joint relation to success and failure in a university-level introductory programming course. The paper compares students' generalized theories about intelligence with domain-specific theories about programming aptitude; and generalized self-efficacies with programming-efficacies in their relation to desire to learn, effort, and performance (and failure). The study explores the following questions: Does student mindset for intelligence and programming are interrelated? Also does specific or general efficacies and mindsets are related? Is it possible to predict student performance using a subset of these measures? At the other end, is it possible that the difficulty of programming and potential failure in the course leads to a belief of fixed programming aptitude and lower programming-efficacy? The latter discussion is enriched by comparing repeating students to first-timers. Repeating students beliefs and theories are particularly important because they may be suffering from lower self-efficacy and a fixed mindset from the previous experience. In addition, this mindset can be contagious for new students which will be somehow susceptible to repeating (and experienced) students' reviews on the course.

### II. METHOD

The current study employs different scales to measure student implicit theories (hereafter mindsets) about 1) intelligence (generalized), 2) programming-aptitude (domain-specific); and different scales to measure 3) self-efficacy (generalized), 4) programming-efficacy (domain-specific). Performance is measured with several variables such as re-enrollment, course grade (and failure), general GPA (Grade Point Average). Student desire to learn is self-reported; whereas effort is measured by their self-reported study frequency. Correlations are examined to explore the relationship among different scales/variables; regression is performed to reveal whether student performance can be predicted; t-tests explored the belief differences among students who are first-timers and repeaters and who have failed and passed the course. The following questions and hypotheses will be tested:

H1- Student general intelligence mindset and general self-efficacy scores and domain-specific programming-aptitude and programming-efficacy scores are correlated with each other and with their effort and performance (course grade, repeats,

GPA). Particularly, students who believe in improvable programming aptitude theory study more, perform better and express a positive desire to learn programming.

H2- Programming-aptitude and -efficacy scores can be used to predict student course performance.

H3- Students who fail the course tend to associate failure with fixed programming aptitude; and have lower self-efficacy. Moreover, re-enrolled students believe in fixed programming aptitude theory more than the first-time enrolling students.

H4- As the course progresses, their mindsets will shift from growth to fixed; and efficacies will decrease.

## A. Course and Data Collection

Introduction to Programming (CSE-101) is a mandatory course for non-CS engineering and IT majors at Işık University. The objective of this course is to introduce the basic concepts of programming, and encourage solving problems by computer programming. The 14-week course introduces the fundamentals of programming, where only the basic constructs {*variables, selections, loops, arrays, methods*} are introduced but object-oriented concepts (e.g. classes, instances) are left out. In addition to a three-hour per week lecture session where programming constructs are introduced, and problems and algorithms are discussed/solved, a laboratory session (two hours per week) introduces small problems/solutions to provide hands-on experience on the Java language platform. The course performance and final grade are calculated over 3x Exams (80%), 5x Quizzes (10%), 12 weeks of laboratory attendance&practice (10%). Exams had 4 or 5 programming questions focusing in different constructs and problem-solving strategies. Quizzes were the open-book problems that students solved in pairs, during lectures. Laboratory sessions included usually four to five solved problems, and two more to be solved by the students. Attending and trying to solve problems in the laboratory was sufficient to get laboratory grade for the week.

The study was conducted in Spring 2016 in two parts: one in the 8th week (pre-survey), and the other in the 13th week (post-survey) of the semester. The surveys were prepared in Turkish (the course is taught in English); created with Google Docs; then filled out online by the students who attended laboratory sessions. The pre-survey was administered one week before the first midterm exam and the post-survey as conducted approximately two weeks before the final exam. Before the pre-survey, the course had completed topics until nested loops, whereas methods and arrays were completed in post-survey, only the basic sorting algorithms were left before the post-test. Prior to the pre-survey, students have already seen lots of mini programming problems, and so they have already faced with the challenge. Prior to the post-survey, the students had received 60% of the course grades. Hence, they were almost able to predict their performance. To evaluate student self-beliefs and theories it is important that they have some experience and intuition about the task and challenge, however, still, it could be useful if the study included another test in the first week of the semester as it was also pointed by the participant data distribution (see the next section).

#### B. Participants

Table I summarizes some demographic information about the participants. In total, there were 193 Turkish students in three different course sections. The pre- (first row) and post- (second row) surveys were completed by 100 and 81 students, respectively. 65 participants completed both the pre- and post-surveys (third column). The participants were mostly male and the vast majority of the respondents were engineering students, excluding computer and software majors.

More than half of the respondents were taking a programming course for the first time. The remainder had failed and re-enrolled, not because they insist on learning programming, rather because the course is mandatory. Figure 1 shows the grade distribution among the participating students of the pre and post surveys versus all students. While the majority of the failed students did not participate -perhaps because of the late application of the surveys-, the participants of both surveys had balanced distributions among different performance levels.

Survey	Total	Gender		Age		Department		#Repo	Course	Avg.		
	Participants	Female	Male	Avg.	Max	Eng.	IT	First-Timer	Repeater	Pass	Fail	GPA
Pre	100	39.0%	61.0%	20.81	27	93.0%	7.0%	62.0%	38.0%	70.0%	30.0%	1.73
Post	81	33.3%	66.6%	20.90	29	88.8%	11.1%	66.6%	33.3%	77.7%	22.2%	1.92
Pre&Post	65	38.5%	61.5%	20.75	27	92.3%	7.7%	69.2%	30.8%	81.5%	18.5%	1.99

TABLE I Demographic information

### C. Scales and Measures

The scales and observed variables are explained below. Readers can find the complete survey, its English translation, and a factor analysis of newly adapted scales online (see Supplementary material).



Fig. 1. Letter grade distribution of (pink-light) all CSE101 (Spring 2016) students, (green-medium) respondents of the pre-survey, (blue-dark) respondents of the post-survey.

1) Mindset for Intelligence (MNDINT): The first part of the survey examined students' mindsets about intelligence using a subset of Dweck's scale [9]. Four items were taken from the original scale and translated into Turkish. Then they were back-translated by a translator and checked by another native English instructor. Two of the items supported the belief in a fixed intelligence theory, e.g. "People have a certain amount of intelligence, and they can't really do much to change it". Whereas two others supported the belief in a growth intelligence, e.g. "No matter which intelligence level he or she has, a person can change it even if just a bit.". These questions were scored on a six-point scale as in the original. The principal components analysis (PCA) revealed that the four items were loading to only one factor between .66 and .85 (explained variance = 58.85). The literature [29] has different findings about the number of factor loadings of mindset scale, perhaps due to the use of different subsets. Our results support Dweck's [9]. Testing the reliability, Cronbach's alpha value was .76; and item-total correlations were between .71 and .82. Cr. alpha is a commonly used measure in the analysis of homogeneity and unidimensionality of scales of more than one items. However, the reliability of Cr. alpha is known to be adversely affected by the number of items in a test [30]. The suitability of the data for factor analysis can be examined by the Kaiser-Meyer-Olkin (KMO) partial correlation coefficient and the Barlett's test of sphericity. The KMO coefficient gives information about whether the data matrix is suitable for factor analysis and whether the data structure is suitable for factoring. For factorability, it is expected that KMO will be higher than .60. The Barlett's test examines whether the test items are correlated with themselves, and have some level of correlation with other items. The Kaiser-Meyer-Olkin (KMO) measure of sampling adequacy was .65, and Bartlett's test of sphericity was significant ( $\chi^2(6) = 105.39, p < .001$ ).

2) Mindset for Programming Aptitude (MNDPRG): Following Scott and Ghinea [2] the authors adapted Dweck's mindset scale to the programming context to measure whether students believed: programming skills can be improved by practice; or it is a fixed skill. Simply, the word "intelligence" in the intelligence mindset scale was replaced with the words "programming-aptitude". For example, the first question was, "People have a certain level of programming-aptitude, and they can't really do much to change it". Again, items were scored on a six-point scale. PCA revealed that the four items were loading to one factor between .66 and .79 (explained variance = 55.61). Cronbach's alpha value was .73; and item-total correlations changed between .70 and .78. The KMO measure of sampling adequacy was .61; and Bartlett's test of sphericity was significant  $(\chi^2(6) = 87.69, p < .001)$ .

3) Self-efficacy (EFCGEN): To construct and analyze (generalized vs specific) correspondence, as of the general intelligence mindset scale and the programming aptitude, the current study employed the generalized self-efficacy scale from Sherer [16] which was adapted to Turkish by Yıldırım [31]. The survey consists of 17 questions that are scored on a five-point scale. The set includes both positive and negative questions such as "I avoid facing difficulties"(-), "I give up easily"(-), "Failure increases my perseverance"(+). The scale is more general than the academic self-efficacy employed in relevant studies [25], [27]. Cronbach's alpha was .89.

4) Programming-efficacy (EFCPRG): There are existing adaptations of self-efficacy to the domain, such as the programmingefficacy scales that were adapted to C and Java language platforms [28], [27]. These scales are layered, and rate student self-efficacy in using different level programming constructs and solving problems of increasing difficulty. However, because the CSE101 course is not object-oriented, the authors could not utilize these programming-efficacy surveys directly. Moreover, a general vs. specific comparison required measuring students' programming beliefs in one-to-one correspondence with their general self-efficacy. Therefore, the authors adapted the latter to measure programming-efficacy from a self-assessment of

 TABLE II

 CORRELATIONS BETWEEN SCALES AND SINGLE OBSERVED VARIABLES

	Rng	Pr/Ps	Mean	SD	MNDINT	MNDPRG	EFCGEN	EFCPRG	EFT	REP	GRD	ATT	GPA	DES	INT∝PRG
MNDINT	[1-6]	Pr	3.62	1.04	-	0.22*	0.29**	0.14	0.02	-0.21*	-0.08	-0.11	-0.03	-	0.06
MNDINT	[1-6]	Ps	3.49	1.0	-	0.31**	0.11	0.16	0.16	-0.21	0.15	0.02	0.21	-0.08	-0.14
MNDPRG	[1-6]	Pr	4.48	0.94		-	0.32**	0.66***	0.53***	-0.28**	0.20*	-0.07	0.16	-	0.03
MNDPRG	[1-6]	Ps	4.11	.99		-	0.29*	0.65***	0.44***	-0.33**	0.48***	0.01	0.36***	0.33**	0.02
EFCGEN	[1-5]	Pr	3.73	0.54			-	0.38***	0.20*	-0.20*	0.19	0.06	0.23*	-	0.0
EFCGEN	[1-5]	Ps	3.62	0.61			-	0.46***	0.20	-0.17	0.21	0.16	0.31**	0.37***	0.09
EFCPRG	[1-5]	Pr	3.66	0.93				-	0.61***	-0.26*	0.25*	-0.13	0.27*	-	0.16
EFCPRG	[1-5]	Ps	3.58	0.79				-	0.50***	-0.36***	0.52***	-0.11	0.45***	0.44***	0.24*
EFT	[1-6]	Pr	4.35	0.54					-	-0.11	0.15	-0.01	0.07	-	0.18
EFT	[1-6]	Ps	4.26	1.03					-	-0.13	0.24*	-0.04	0.25*	0.22*	0.33**
REP	[1+]	Pr	1.79	1.21						-	-0.16	0.11	-0.34***	-	-0.04
REP	[1+]	Ps	1.65	1.12						-	-0.32**	-0.05	-0.40***	-0.20	-0.18
GRD	[0-4]	Pr	1.78	1.49						-	-	-0.05	0.79***	-	-0.01
GRD	[0-4]	Ps	1.99	1.49							-	-0.14	0.78***	0.32**	0.25*
ATT	[0 - 10]	Pr	4.9	3.2								-	-0.04	-	0.06
ATT	[0-10]	Ps	4.48	3.23								-	-0.08	0.34**	0.18
GPA	[0-4]	Pr	1.73	1.44									-	-	0.07
GPA	[0-4]	Ps	1.90	0.82									-	0.20	0.19
DES	[1-6]	Pr	-	-										-	-
DES	[1-6]	Ps	3.63	1.27										-	0.22*
<b>INT PRG</b>	[1-6]	Pr	3.27	1.41											-
<b>INT ~PRG</b>	[1-6]	Ps	3.47	1.43											-

Note: Rng: range, Pr. Pre, Ps: Post survey; MNDINT: Mindset intelligence; MNDPRG: Mindset Programming-aptitude; EFCGEN: General self-efficacy; EFCPRG: Programming-efficacy; EFC: Effort REP: Course repeats, GRD: Course grade, ATT: Laboratory attendance GPA: Cumulative GPA, DES: Desire to learn programming, INT $\propto$ PRG: Intelligence proportional to programming N<sub>Pr</sub>=100, N<sub>Ps</sub>=81; \*p < .05; \*\*p < .001

difficulty perspective. Starting with a subset of the original scale (EFCGEN), simply the keyword programming was inserted in the appropriate positions to form six questions that were scored on a five-point scale. Items included both positive and negative questions such as "*I am sure that I can learn programming*"(+), "*Programming is a difficult task for me*"(-), "*If a programming problem seems too complicated, I do not try solving it*"(-). PCA revealed that the six items were loading to one factor between .69 and .85 (explained variance = 60.03). Cronbach's alpha was .86, and item-total correlations were between .73 and .83. The KMO measure of sampling adequacy was .81, and Bartlett's test of sphericity was significant ( $\chi^2(15) = 233.22, p < .001$ ).

5) Effort&Practice (EFT): To measure student effort, the survey included three items on a six-point scale which questioned self-reported programming effort. The set includes "I learn programming by following and noting programs in lectures/labs", "I learn programming by writing programs on a computer myself". PCA revealed that the three items were loading to one factor .61 to .81 (explained variance = 54.6). Cronbach's alpha was .72. The KMO measure was .80, and Bartlett's test was significant ( $\chi^2(6) = 61.13, p < .001$ ).

#### D. Single Observed Variables

1) Course repeats (**REP**): indicates the number of previous enrollments in the course (survey question 35 in the supplementary material). More than one-third of the respondents had previously failed and re-enrolled in this mandatory course.

2) Course grade (**GRD**): is the end-of-term performance grade of each respondent ({F=0, DD=1.0, DC=1.5, CC=2.0, CB=2.5, BB=3, BA=3.5, AA=4.0}), calculated from the average described in Section II-A. Students receiving a grade below CC are advised to repeat, whereas students with an F must repeat the course.

3) Laboratory Attendance (ATT): rates the student attendance to laboratory sessions from 0 to 10. This is also included in the course grade.

4) Cumulative Grade Point Average (GPA): is over all courses that the student has taken, as recorded on their transcript. Usually, Engineering students enroll for CSE101 in their first year, the second term. However, as students can fail and re-enroll in different terms, the effect of CSE101 on the GPA may be high or low, but not zero.

5) Desire to learn (**DES**): rates student's answer to the question "How much do you want to learn programming?" (question 36, only in post-survey). Respondents answered on a five-point scale; two-thirds expressed that they very enthusiastic to learn programming.

6) Intelligence proportional to programming aptitude (INT  $\propto$  PRG): rates agreement with the statement "intelligence and programming aptitude are proportional", (question 37) on a six-point Likert scale. Mean of pre- and post-surveys denote that on average students slightly disagree with the proposition.

### III. DATA ANALYSIS

1) Hypothesis 1 (Correlations): Do Student general intelligence mindset and general self-efficacy scores and domainspecific programming-aptitude and programming-efficacy scores are correlated with each other and with their effort and performance (course grade, repeats, GPA). Do students who believe in improvable programming aptitude theory study more, perform better, and express a positive desire to learn programming?: Table II shows data from pre- (Pr) and post-surveys (Ps) with means (MN) and standard deviations (SD). The scores were normed with respect to the number of questions in

TABLE III Multiple Linear Regression Analysis of Grade

Model 1 (Course grade)											
Pre Survey	В	SE(B)	$\beta$	t	Sig.(p)						
MNDINT	47	.30	16	-1.58	.117						
MNDPRG	.23	.42	.07	.55	.578						
EFCGEN	.79	.60	.14	1.31	.193						
EFCPRG	.57	.44	.17	1.30	.196						
$R^2 = .1, R^2_{aa}$	$_{ij} = .00$	3									

each survey. The domain-specific scales: the mindset for programming (MNDPRG), programming efficacy (EFCPRG) and effort (EFT) are all positively inter-correlated; and also positively correlated with course grade and negatively correlated with course repeats. The desire to learn programming (DES) is positively correlated with the mindset for programming, general self-efficacy and more strongly with programming efficacy. The stronger correlations in pre-survey generally agree with the post surveys'. The results reveal that student who believe in an improvable programming aptitude (higher in MINDPRG) have higher programming efficacy. They also study more, get higher grades in the course; less likely to repeat the course; and express a positive desire for learning programming. An observation is that effort and the agreement for "intelligence and programming aptitude is proportional" statement ( $INT \propto PRG$ ) is significantly positively correlated. The scatter plot of the data (not shown) showed that the majority of the students report effort above average, whereas only a few completely disagree with "intelligence and programming aptitude" are proportional.

2) Hypothesis 2 (Performance prediction): Can programming aptitude and programming-efficacy scores (domain-specific scales) be used to predict student course performance (grade)?: A multiple linear regression analysis was conducted to predict student course grade by the pre-survey scale scores. Table III shows that none of the predictors had significant contribution to the model, despite a weak regression (F(4;95)=2.6, p=.041,  $R^2=.1$ ,  $R^2_{Adj}=.06$ ). Thus, grade prediction cannot be confirmed.

3) Hypothesis 3 (Previous fails and re-enrollment): Do students who fail the course tend to associate failure with fixed programming aptitude, and have lower self-efficacy? Do previously failed (and re-enrolled) students believe in fixed programming aptitude theory more than the first-time enrolling students?: Table IV shows independent-samples t-tests (from pre-survey  $N_{100}$ ) that compare MNDINT, MNDPRG, EFCGEN and EFCPRG scores for the first-timer and re-enrolled students. There was a significant difference in MNDPRG and EFCPRG scores (respectively,  $t_{(79)}=3.58$ , p<.001;  $t_{(98)}=3.137$ , p<.001). The lower values of programing mindset of re-enrolled students suggest that a past failure amplified their belief on fixed programming aptitude. The post-survey data shows that the difference was larger at the end of the term. In addition, t-tests from post-survey show that views of students who fail and pass differed significantly regarding mindset for programming, self-and programming efficacy: MNDPRG ( $t_{(79)}=-2.1$ , p<.001), EFCGEN ( $t_{(79)}=-4.45$ , p<.05) and EFCPRG ( $t_{(79)}=-5.54$ , p<.001), respectively.

4) Hypothesis 4 (Change during the course): Does student mindset shift from growth to fixed; and efficacies decrease during the course.: Table II shows that scores for all five scales dropped from pre-survey to the post-survey. For the students who have attended both pre and post-survey, a paired-samples t-test was conducted to compare MNDINT, MNDPRG, EFCGEN and EFCPRG scores (N<sub>65</sub>). Though, all the mean values dropped from the pre- to post-survey, the only significant change was for MNDPRG ( $M_{pre}$ =4.46, SD<sub>pre</sub>=.97 and  $M_{post}$ =4.2, SD<sub>post</sub>=.96; t<sub>(64)</sub>=2.5; p=.015;) and marginally for EFCGEN( $M_{pre}$ =3.75, SD<sub>pre</sub>=.6 and  $M_{post}$ =3.59, SD<sub>post</sub>=.67; t<sub>(64)</sub>=2.4, p=.052).

			MNDINT			MNDPRG			EFCGEN			EFCPRG		
		М	SD	t	Μ	SD	t	М	SD	t	Μ	SD	t	
Pre-enrollment Pre-survey N=100	First-timer (n=62) Re-enrolled (n=38)	3.71 3.49	0.93 1.19	1.023	4.74 4.06	0.81 0.99	3.580***	3.81 3.62	0.45 0.66	1.614	3.84 3.26	0.83 0.98	3.137**	
Performance Post-survey N=81	Failed (n=46) Passed (n=35)	3.42 3.59	0.99 1.02	741	3.7 4.6	0.92 0.88	-2.101*	3.46 3.76	0.71 0.53	-4.45***	3.22 4.06	0.67 0.68	-5.54***	

TABLE IV INDEPENDENT GROUP T-TESTS BETWEEN FIRST TIME AND RE-ENROLLING STUDENTS; FAILED AND PASSED STUDENTS

\*p < .05, \*\*p < .01, \*\*\*p < .001.

### IV. DISCUSSION

Mindset for programming aptitude (MNDPRG) represents the belief in improvable programming aptitude, whereas programmingefficacy (EFCPRG) is student's self-belief of his/her capacity to learn programming, or in other words, his/her assessment of the challenge of programming from his/her self-perceived capacity. The generalized scales represent the belief in improvable intelligence (MNDINT), and task-independent self-efficacy (EFCGEN), respectively. The correlation analysis of data from surveys has several implications. First, as expected, domain-specific measures MNDPRG and EFCPRG are more relevant to programming than generic scales. The generic scales had some correlation with their domain-specific counterparts. Second, MNDPRG and EFCPRG are highly correlated. Therefore student's belief in whether programming aptitude can be improved by effort is somehow related to his/her belief in self-capacity to learn programming, and vice-versa. Furthermore, both measures correlate significantly with self-reported effort (EFT). Hence, for a significant portion of the survey students, a growth mindset for programming and higher programming-efficacy means higher effort. However, the weak correlation between self-reported effort and course grade shows that not all hard-working students get higher grades (GRD), despite the positive correlation between GRD, MNDPRG and EFCPRG. This may be the reason for the correlation among (for some of) the students who self-report higher effort&practice but also agree with "programming aptitude and intelligence is proportional". Confirming Scott and Ghinea's [2], effort&practice is very related to mindset for programming. The current paper shows that both are related with programming-efficacy.

The analysis also revealed a correlation between self-expressed desire to learn (DES) and programming mindset and particularly with programming-efficacy. Furthermore, data supports that students who believe in improvable programming aptitude theory express much desire to learn programming, study more, and get higher course grades; and they think programming is a manageable challenge for themselves. Since both mindset and efficacy are related with goal settings, this correlation among programming efficacy and desire may also indicate a self-revision of goals in the face of difficulty, such that "Programming is difficult for me, I do not want to learn programming!". Students who declared less desire also had lower programming efficacy and are aligned towards fixed programming aptitude theory. However, whether these correlations indicate cause and effect or attribution relation can be answered after discussing other results.

Seeing the correlations, what is the possibility of predicting the course grade with mindset and efficacy measures. The regression analysis in the pre-survey has shown prediction would not be significant, at least for all students. This may have proven the obvious fact that course grade is a complex outcome that depends on many factors. However, it may still be possible to explore prediction for a subset of students, such as repeaters, which was not explored here. The lack of predictive power of implicit theories in course grade or GPA, align with Kornilova's [25] findings that implicit theories of intelligence had no predictive power on GPA. However, as Dweck shows [22] the effect on achievement may reveal itself over time as it acts indirectly by impacting goal orientations.

The decreasing self- and programming-efficacies and decreasing belief in an improvable programming aptitude show that all students of the course (sampled by the participants) revise their theories and self-beliefs negatively upon facing the challenges, which is parallel with the attribution theory and relevant research [5], [2], [25].

Programming mindset and efficacy scores of students who have failed the course were significantly lower than the successful ones. Therefore the strong correlations of these two factors with the course grade, but lack of significant prediction result, supports the attribution theory more than a predictive capacity. In other words, students who had trouble coping with the programming challenge and foreseen an upcoming failure, attribute it to the limited aptitude and also to that programming is a difficult task for themselves.

Hong et. al. [19] proposed that implicit theories can set up frameworks which can be filled with attributions in the case of failure. In the current case, these attributions may feed the fixed programming aptitude theory in a cyclic way, and contribute to future failures for repeating students. The significantly lower MNDPRG and EFCPRG scores of re-enrolled students support this conclusion. Thus, some of the prospective students for the next term would have pre-attributed the failure to their fixed programming aptitude.

There are several limitations of this study which may infer external validation of its results. First of all, while sample sizes seem sufficient for statistical analysis, the participants were from the same institution. Second, a major portion of the failed students did not even participate in the first survey. So our results may be biased to reflect beliefs of persistent students. Third, some of the scales such as effort relied only on self-reported values to indicate study behavior. An objective measure such as externally counted "number of programs written, number of questions attempted" could be more objective. Gore [32] showed that predictive power of efficacy depend on when efficacy beliefs are measured and types of efficacy questions. Hence, one can argue that the application of the pre-survey on the 8th week (before the first midterm) may have affected measurements of programming mindset and efficacy, since students may have already foreseen their own performance. However, to measure self-efficacies (in a specific task), the person must be exposed to the task. Therefore this study had delayed application of the pre-survey also that did not employ detailed programming efficacy questionnaires such as developed in [15], [28], which included questions such as "I understand OOP or Java Applets".

### V. CONCLUSION

The comparison of implicit theories of intelligence, programming aptitude and self-efficacy in relation to student effort and performance in programming has confirmed and also enriched some of the earlier findings [2] that students develop domain-specific implicit theories and self-beliefs. In particular, the current study shows that students who believe in improvable programming aptitude and have higher programming-efficacy study more, and get higher grades. On the other side, students who are re-enrolled to the course support a fixed programming aptitude theory more and have lower programming efficacy. Thus they come pre-attributed the previous failure(s) to "programming is a difficult task for me, related to the pre-acquired skills/genes more than effort". Therefore, belief in a fixed programming-aptitude and having a low programming-efficacy

significantly increases the likelihood of course fails and thus repeats. Fortunately, it is possible to detect some of those students with the measures employed in this study. Therefore, an intervention must target increasing programming-efficacy and the false belief that "the programming aptitude is fixed"; and show to the student that programming skill is improvable by practice. Acknowledging students with the results that the failure/repeats and the fixed programming-aptitude and low programmingefficacy is correlated, can be the first step as an intervention to strengthen the belief in a growth programming aptitude. Bandura [11] emphasizes that mastery experience is the most effective source of self-efficacy. Hence, the course plans must include greater numbers of incremental steps (and feedback) in the learning process [33], [34], and support personalized learning perhaps via other media such as mobile apps [20] to prevent student early frustration and perception of programming as a "difficult task". Devising such interventions can be supported also by qualitative studies on students that believe in a fixed programming aptitude and have low programming-efficacy. The authors currently qualitatively study the sub-factors of fixed programming-aptitude belief and low-efficacy.

#### REFERENCES

- [1] M. E. Caspersen and M. Kolling, "Stream: A first programming process.," ACM Trans. on Computing Education, vol. 9, no. 1, p. 4, 2009.
- [2] M. J. Scott and G. Ghinea, "On the domain-specificity of mindsets: The relationship between aptitude beliefs and programming practice." IEEE Transactions on Education, vol. 57, no. 3, pp. 169-174, 2014.
- [3] J. Rountree, N. Rountree, A. Robins, and R. Hannah, "Observations of student competency in a cs1 course." in In Proc. of the 7th Australasian Conf. on Computing Education-Vol 42. Australian Computer Society Inc, 2005, pp. 145-149.
- [4] E. Djikstra, "A debate on teaching computer science: On the cruelty of really teaching computer science." Communications of the ACM, vol. 32, pp. 1398-1404, 1989.
- [5] Q. Cutts, E. Cutts, S. Draper, P. O'Donnell, and P. Saffrey, "Manipulating mindset to positively influence introductory programming performance," in In Proc. of the 41st ACM Symposium on Comp. Sci. Edu., 2010, pp. 431-435.
- [6] B. Oakley, A Mind For Numbers: How to Excel at Math and Science (Even If You Flunked Algebra). New York: Penguin, 2014.
- [7] A. Bandura, Self-efficacy, part in Encyclopedia of human behavior, V. Ramachaudran, Ed. Academic Press., vol. 4.
- [8] A. L. Duckworth, C. Peterson, M. D. Matthews, and D. R. Kelly, "Grit: Perseverance and passion for long-term goals," vol. 92, pp. 1087-1101.
- [9] C. S. Dweck, C. Chiu, and Y. Hong, "Implicit theories and their role in judgments and reactions: A world from two perspectives," Psychological Inquiry, vol. 6, pp. 267-285, 1995.
- [10] J. A. Rosen, E. J. Glennie, B. W. Dalton, J. M. Lennon, and R. N. Bozick, Noncognitive Skills in the Classroom: New Perspectives on Educational Research, Research Triangle Park, NC, 2010.
- [11] A. Bandura, Guide for constructing self efficacy scales, in Self-Efficacy Beliefs of Adolescents in Adolescence and Education, U. T. and P. F., Eds. Information Age Pub., vol. 4.
- [12] N. A. Mamaril, E. Usher, C. R. Li, D. R. Economy, and M. S. Kennedy, "Measuring undergraduate students' engineering self-efficacy: A validation study," Journal of Engineering Education, vol. 105, no. 2, pp. 366-395, 2016.
- [13] C. Murphy, D. Coover, and S. Owqn, "Development and validation of the computer self-efficacy scale." vol. 49, pp. 893-899, 1989.
- [14] N. Betz and G. Hackett, "The relationship of mathematics self-efficacy expectations to the selection of science-based college majors," Elsevier Journal of Vocational Behavior, vol. 23, pp. 329-345.
- [15] V. Ramalingam, D. LaBelle, and S. Wiedenbeck, "Self-efficacy and mental models in learning to program," In ACM SIGCSE Bulletin., vol. 36, no. 3, pp. 171-175, 2004.
- [16] M. Sherer, J. Maddux, B. Mercandante, and R. R. W., "Construct validation of the self-efficacy scale," vol. 53, pp. 899–902, 1983.
- [17] General self-efficacy scale (gse). [Online]. Available: http://userpage.fu-berlin.de/health/selfscal.htm
- [18] R. Schwarzer and M. Jerusalem, "Generalized self-efficacy scale." in Measures in health psychology: A user's portfolio. Causal and control beliefs, e. a. J. Weinman, Ed. Windsor, NFER-NELSON, 1995, pp. 35-37.
- [19] Y. Y. Hong, C. Y. Chiu, C. S. Dweck, D. M. S. Lin, and W. Wan, "Implicit theories, attributions, and coping: A meaning system approach," J. of Personality and Social psychology, vol. 77, no. 3, p. 588, 1999.
- [20] M. J. Scott and G. Ghinea, "Implicit theories of programming aptitude as a barrier to learning to code: are they distinct from intelligence?" in In Proc. of the 18th ACM Conf. on Innovation & Technology in Comp. Sci. Edu. ACM, pp. 347–347.
- [21] J. A. Mangels, B. Butterfield, J. Lamb, C. Good, and C. S. Dweck, "Why do beliefs about intelligence influence learning success? a social cognitive neuroscience model," Social cognitive and affective neuroscience, vol. 1, no. 2, pp. 75-86, 2006.
- [22] C. S. Dweck, "Mindsets and science achievement," in Carnegie Corp. of New York-Inst. for Adv. Study Commiss. on Math. and Sci. Edu., 2008.
- [23] P. Byrne and G. Lyons, "The effect of student attributes on success in programming." In ACM SIGCSE Bulletin, vol. 33, no. 3, pp. 49-52, 2001.
- [24] A. J. Gomes, A. N. Santos, and A. J. Mendes, "A study on students' behaviours and attitudes towards learning to program," in In Proc. of the 17th ACM Conf. on Innovation & Technology in Comp. Sci. Edu. ACM, 2012, pp. 132-137.
- [25] T. V. Kornilova, S. A. Kornilov, and M. A. Chumakova, "Subjective evaluations of intelligence and academic self-concept predict academic achievement: Evidence from a selective student population." Learning and Individual Differences, vol. 19, no. 4, pp. 596-608, 2009.
- [26] P. Kinnunen and B. Simon, "My program is ok-am i? computing freshmen's experiences of doing programming assignments," Computer Science Education, vol. 22, no. 1, pp. 1-28, 2012.
- [27] O. Korkmaz and H. Altun, "Adapting computer programming self-efficacy scale and engineering students' self-efficacy perceptions," Participatory Educational Research (PER)., vol. 1, p. 2031, 2014.
- [28] P. Askar and D. Davenport, "An investigation of factors related to self-efficacy for java programming among engineering students," The Turkish Online J. of Educational Technology, vol. 8, p. 1, 2009.
- [29] A. Diseth, E. Meland, and H. Breidablik, "Self-beliefs among students: Grade level and gender differences in self-esteem, self-efficacy and implicit theories of intelligence," Learning and Individual Differences, vol. 35, pp. 1-8, 2014.
- [30] M. Tavakol and R. Dennick, "Making sense of cronbach's alpha," Int. J. of Medical Education, vol. 2, pp. 53-55, 2011.
- [31] F. Yıldırım and I. İlhan, "Genel Özyeterlilik Ölçeği türkçe formunun geçerlilik ve güvenilirlik Çalışması," *Türk Psikiyatri Dergisi*, pp. 301–308, 2010. [32] G. P. A, "Academic self-efficacy as a predictor of college outcomes: Two incremental validity studies," *Journal of Career Assessment*, vol. 14, pp. 92-115.
- [33] T. Ahoniemi, E. Lahtinen, and T. Erkkola, ""fighting the student dropout rate with an incremental programming assignment"," in In Proc. of the Seventh Baltic Sea Conf. on Computing Education Res.-Vol 88, 2007, pp. 163-166.
- [34] J. Stamey and S. Sheel, "A boot camp approach to learning programming in a cs0 course." Journal of Computing Sciences in Colleges, vol. 25, no. 5, pp. 34-40, 2010.