International Journal of Computer Science Education in Schools, April 2022, Vol. 5, No. 3
ISSN 2513-8359

The Development and Validation of the Programming Anxiety Scale

Osman Gazi YILDIRIM[1]
Nesrin OZDENER[2]

[1] National Defense University Army NCO Vocational College/Turkey
[2] Marmara University/Turkey

DOI: 10.21585/ijcses.v5i3.140

Abstract

The main goal of the current study is to develop a reliable instrument to measure programming anxiety in university students. A pool of 33 items based on extensive literature review and experts' opinions were created by researchers. The draft scale comprised three factors applied to 392 university students from two different universities in Turkey for exploratory factor analysis. The number and character of the underlying components in the scale were determined using exploratory factor analysis. After exploratory factor analysis, confirmatory factor analysis was conducted on the draft scale using a sample of 295 university students. Confirmatory factor analysis was carried out to ensure that the data fit the retrieved factor structure. The internal consistency coefficient (Cronbach's alpha) was calculated for the full scale and each dimension for reliability analysis. For convergent validity, the factor loading of the indicator, the average variance extracted, composite reliability, and maximum share variance values were calculated. Additionally, convergent validity was tested through (1) comparison of mean values of factors and total programming anxiety depending on gender and (2) correlation analysis of factors, total programming anxiety, and course grade of students. The Fornell & Larcker criterion and the Heterotrait-Monotrait correlation ratio were utilized to assess discriminant validity. According to analysis results, the Programing Anxiety Scale (PAS) comprised 11 items in two factors: classmates and self-confidence. Similarly, results revealed that The PAS has good psychometric properties and can be used to assess the programming anxiety of university students.

Keywords: computer programming, programming anxiety, scale development, scale validation

1. Introduction

With the development of the internet and mobile technologies in the last two decades, breakthroughs have been experienced in many fields of computer science such as big data, artificial intelligence, blockchain, bioinformatics, wearable technologies, cloud computing, 3D printers, robotics, and virtual reality. Responsibilities of computer science such as automating the processes, facilitating communication, providing better products and services, assisting the world to be more productive have caused human beings to be more dependent on software (Santos, Tedesco, Borba, & Brito, 2020). As a result, all developed and developing countries are required to raise qualified individuals who can maintain the software used and produce practical solutions to new problems encountered in the future (Demirer & Sak, 2016). One of the conditions for the success of this task is to provide students with programming skills, which is considered one of the requirements of being a well-educated and knowledgeable citizen (Al-Makhzoomy, 2018; Kert & Uğraş, 2009). However, according to several studies, most computer science students regard programming courses as complicated and intimidating (Bennedsen & Caspersen, 2007; Connolly, Murphy, & Moore, 2009; Jenkins, 2002; Owolabi, Olanipekun, & Iwerima, 2014; Robins, Rountree, & Rountree, 2003; Wiedenbeck, Labelle, & Kain, 2004). Moreover, studies show that programming courses have high dropout and failure rates (Bennedsen & Caspersen, 2007; Luxton-Reilly et al., 2019).

Over the last decade, numerous research has been undertaken on the factors affecting learner success in programming courses. Previous studies indicate that programming background (Bunderson & Christensen, 1995; Byrne & Lyons, 2001), mathematical knowledge (Butcher & Muth, 1985; Wilson & Shrock, 2001), problem-solving skills (Gibbs, 2000; Hostetler, 1983), learning styles (Byrne & Lyons, 2001; Tan, Ting, & Ling, 2009), expectations of students for course outcome (Rountree, Rountree, & Robins 2002), comfort level (Bergin & Reilly, 2005) and self-efficacy (Ramalingam & Wiedenbeck, 1998) impact achievement of students in programming courses. Similarly, programming anxiety is also a significant predictor of achievement in programming (Connolly et al., 2009; Maguire, Maguire, & Kelly, 2017).

1.1 Related Work

Connolly, Murphy, and Moore (2007) define programming anxiety as a situation-specific psychological state caused by negative experiences or expectations in a computer programming situation. Connolly et al. (2007) also claim that programming anxiety is caused by the incorrect self-assessment of students' abilities when learning to program. According to Scott (2015), students often encounter programming anxiety at the initial stages of programming courses because programming courses involve concepts and materials that are "*radically novel*" (Dijkstra, 1989). Moreover, as beginning programming courses have become more abstract over the last few decades, programming anxiety has increased (Connolly et al., 2009). This particular content can evoke intense negative feelings (Huggard, 2004) such as confusion, frustration, and boredom (Bosch, D'Mello, and Mills, 2013) which is described as a phenomenon called "*programming trauma*" (Huggard, 2004).

Since learners' self-belief plays a fundamental role in intellectual development (Berland & Lee, 2011; Pajares, 1992), Jiang, Zhao, Wang, and Hu (2020) believe that this trauma happens when students lose their self-efficacy in programming, which negatively affects learning outcomes. Connolly et al. (2007) propose a cognitive model to explain how programming anxiety influences students' emotional, behavioral and physiological reactions (see Figure 1). The mental model asserts that students' automatic thoughts are activated in programming situations, directly influenced by their core and intermediate beliefs. Eventually, automatic thoughts affect their emotional, behavioral, and physiological reactions. According to Connolly et al. (2007), a fear of programming may commence caused by core beliefs for a student sensitive to programming anxiety. Then, intermediate thoughts of students could emerge as a fear of what other students might think about their performance and ability. Finally, automatic thoughts arise in programming situations and trigger negative thoughts and reactions.

In addition to the cognitive model for programming anxiety, Rogerson and Scott (2010) also depict an iceberg model to explain factors affecting fear of programming. According to the iceberg model, the fear of programming is induced due to the nature of programming. Rogerson and Scott (2010) cite those internal factors such as motivation, attitude, self-efficacy, and attribution often have a part in building negative perceptions of programming. At the same time, peers, teaching methodology, timing, lectures, and tutors constitute external factors.

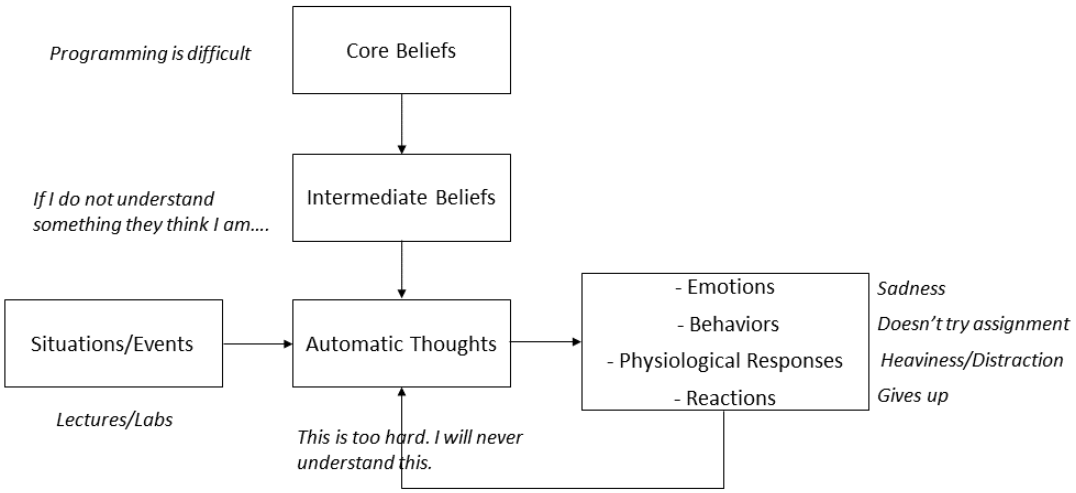

Figure 1. Proposed Cognitive Model for Programming Anxiety (Connolly et al., 2007)

The number of studies on programming anxiety has risen dramatically in the last decade. Some of these studies examined factors associated with programming anxiety, while others investigated the impact of programming anxiety on student performance and motivation. According to S Sinožić and Orehovaki (2018), the absence of programming experience, fear of programming, and a misperception of programming languages as very complex are all powerful determinants of programming anxiety among novices. Similarly, unfamiliar subjects in programming courses make students avoid programming, and programming makes them feel uncomfortable (Olipas, Leona, Villegas, Cunanan & Javate, 2021). According to studies, learners' programming anxiety levels aggregated as they were presented to programming concepts and principles. (Campbell, 2018; Dasuki & Quaye, 2016).

Several studies have also connected programming anxiety to academic performance, perceived self-efficacy, encountering errors when developing programs, gender, peers, test anxiety, mathematics, and computer anxiety. For example, Olipas et al. (2021) found a negative association between participants' academic performance and programming anxiety in a study of 348 students. Hsu and Gainsburg (2021) and Wilfong (2006) explain that self-efficacy plays a vital role in performance in programming courses, and self-efficacy has a mediating effect on the relationship between anxiety and performance. Results of a systematic review of the literature conducted by Nolan and Bergin (2016) illustrate correlates of programming anxiety as programming as a subject, test anxiety, computer anxiety (volume of computer usage), and using mathematics frequently in coding. Additionally, the students' incapacity to debug their programs increase their programming anxiety (Dasuki & Quaye, 2016; Nolan & Bergin, 2016).

Some researchers mention the effects of peers on programming anxiety. According to Nolan and Bergin (2016), when programming students learn to program in a laboratory with many peers, this circumstance can be stressful. Falkner, Falkner, and Vivian (2013) explored how collaborative practices in programming courses can cause fear and tension in learners. They concluded that working in groups prevented students from feeling comfortable in classes. There are also studies in the programming literature on the effects of gender on programming anxiety. According to Olipas and Luciano's (2020) study, female students show more programming anxiety than male students. Chang (2005) also explored a possible association between the perceived complexity of programming tasks and programming anxiety with 307 participants. According to the findings, there was a strong association between these two variables, indicating that as the perceived complexity of programming assignments increased, so did students' perceived programming anxiety levels.

Many studies in the literature state that programming anxiety is one of the factors that cause students to fail and lose interest in programming courses. It is reported that programming anxiety is critical in determining students' success in a programming course (Connolly et al., 2007; Figueroa & Amoloza, 2015; Kinnunen & Malmi, 2006; Nolan, Bergin & Mooney, 2019; Owolabi et al., 2014; Scott, 2015). With self-beliefs being the case, Kinnunen and Simon (2012) assert that learners' self-beliefs are developed due to the experiences students have while they engage in programming activities rather than the resulting quality of the programs they write. As a consequence of negative experiences and self-appraisals, learners lack the time or have no motivation to program (Kinnunen & Malmi, 2006; Scott, 2015). Similarly, Maguire et al. (2017) assert that programming anxiety causes a lack of confidence and plays a crucial role in discouraging students from carrying out programming independently. Results of the study of Özmen and Altun (2014) show that while students with a low level of programming anxiety spend extra time on programming and code more qualified programs, students with a high level of anxiety devote limited time on programming practices and avoid learning programming. Similar results have been cited by Scott (2015), concluding that programming anxiety inhibits time spent practicing programming and decreases course participation (Bergin & Reilly, 2005). Scott and Ghinea (2014) investigated the possible adverse effects of programming anxiety on students' programming practice. Participants of the study were 239 university students. The findings revealed that students are frequently concerned when undertaking debugging activities.

In the light of all these studies in the literature, it is essential to measure the programming anxiety levels of students with reliable and valid instruments to determine students' anxiety levels and help learners overcome their anxiety and frustration in programming courses. However, despite anxiety's critical role in programming, research on anxiety scale development has been deficient. Information about these measurement instruments is summarized in Table 1.

Table 1. A Summary of the Relevant Scales

Name of the Scale/Survey	Factors	The Total Number of Items
Programming Anxiety Survey (Figueroa & Amoloza, 2015)	- Not Applicable	6
The Computer Programming Anxiety Questionnaire (Connolly et al., 2009)	- Gaining Initial Computing Skills - Sense of Control - Computer Self Concept - State of Anxiety in Computer Situations	15
The Computer Programming Anxiety Scale (Choo & Cheung, 1991)	- Errors - Significant Others - Confidence	19

As presented in Table 1, three scales are prepared to measure primarily programming anxiety. All of the scales are based on self-reported data. In addition to these scales, it was noted that computer anxiety or information technology (IT) anxiety scales were adapted for measuring programming anxiety in several studies (see Olipas & Luciano, 2020; Scott & Ghinea, 2014, and Orehovacki, Radosevic & Konecki, 2012). Furhermore, Demir (2021) recently adapted Choo and Cheung's (1991) programming anxiety scale into Turkish.

1.2 Purpose of the Study

Studies show that reducing anxiety can enhance academic performance and achievement (Hattie, 2008). The same is true when it comes to improving the efficiency of programming courses. It is vital to identify learners' programming anxiety and work closely with students with high anxiety to develop the learning outcomes of programming courses at the highest level. In this sense, there is a need for reliable measurement tools designed to measure programming anxiety to make meaningful conclusions from the analysis. As a result, the current research aims to create a proper and reliable tool to measure programming anxiety in university students.

2. Method

The Computer Programming Anxiety Scale was developed and validated in three phases, illustrated in Figure 2. In summary, dimensions of the draft scale were identified, and the item pool was generated in the first phase. In the second phase, content and phase validity were assessed. In the last stage, exploratory and confirmatory factor analysis was conducted, and construct validity was evaluated.

2.1 Phase 1: Identifying dimensions & item generation

Clark and Watson (1995) recommend beginning scale development by clearly conceptualizing the target construct and clarifying its breadth and scope. The researchers conducted a comprehensive literature review and content analysis to identify different dimensions of programming anxiety. With this respect, models and explanations related to programming anxiety were examined to develop a clear conceptualization. Furthermore, related constructs including computer anxiety, math anxiety, and test anxiety were investigated. The Computer Programming Anxiety Scale (Choo & Cheung, 1991) was used to identify programming anxiety dimensions. At the same time, scales developed to measure students' anxiety, such as programming anxiety, computer anxiety, test anxiety, math anxiety, and foreign language learning anxiety, were investigated. Depending on the studies on programming anxiety, three dimensions were proposed: (1) classmates, (2) self-confidence, and (3) errors. The "Classmates" subscale measures students' anxiety in the presence of more proficient students. The "Programming confidence" subscale measures students' feelings of inadequacy while programming. "Errors" subscale measures students' anxiety when confronted with errors during programming.

Next, a pool of 33 items was constructed to capture negative emotions during program development and debugging. The rationale underpinning including as many items as possible in the draft scale was that the number of items at the start should be twice as numerous as the final scale (Nunnally, 1994). To obtain precise and unambiguous items that reflect the specified conceptual definitions, item wording rules suggested by

Carpenter (2018) were applied. This cyclical item development procedure yielded a total of 33 items, each rated on a 5-point scale from 1 ("*never true*") to 5 ("*always true*"). In this regard, "*seldom true*" was scored as 2, "*sometimes true*" was 3, and "*often true*" was 4.

2.2 Phase 2: Development of the scale

2.2.1 Content validity

Content validity of the draft scale was tested by interviewing five experts, three of whom were from the field of instructional technology, one from the field of Turkish language, and one from the field of psychological counseling and guidance. An expert opinion form was created in this phase. The experts were requested to rate each scale item using this form on a four-point rating scale (1 = not relevant; 2 = item requires so much revision that it is no longer relevant; 3 = item is suitable but needs minor changes; 4 = highly relevant). Data gathered from the expert opinion was used to quantify the content validity process and calculate Content Validity Index (I-CVI; Polit, Beck, & Owen, 2007). The I-CVI was calculated by dividing the experts who provided a 3 or 4 by the total number of experts for each item (Lynn, 1986). Nine items with an I-CVI-score of less than one were excluded from the draft scale using Lynn's (1986) criteria. In addition, based on the experts' recommendations, two of the retained items were revised to simplify the language. This operation yielded 24 items in the final pool (classmates: 8 items; self-confidence: 9 items; errors: 7 items). Table 2 depicts the item pool on the draft scale.

2.2.2 Face validity

The questionnaire's face validity was assessed quantitatively. To evaluate the qualitative face validity, nine college students enrolled in a programming course were interviewed face to face and participants rated the items based on clarity and relevancy. Despite some minor errors, all of the interviewees concurred on the clarity and comprehensibility of all of the items.

2.2.3 Translation of the Scale

The Programming Anxiety Scale items were created and written in Turkish at first. The data were collected utilizing this original scale. The translation of the original scale to English was carried out after the data collection process. A mixed translation strategy utilizing the back-translation method and the committee approach that was distinct from Jones, Lee, Phillips, Zhang & Jaceldo (2001) was used in the translation process. The researchers initially translated each item in the original version into English. Next, three Turkish/English bilingual professors thoroughly inspected each translated item. With the help of the multilingual teacher group, any necessary modifications to problematic items were performed. The bilingual experts agreed on the translated and original versions of the scale.

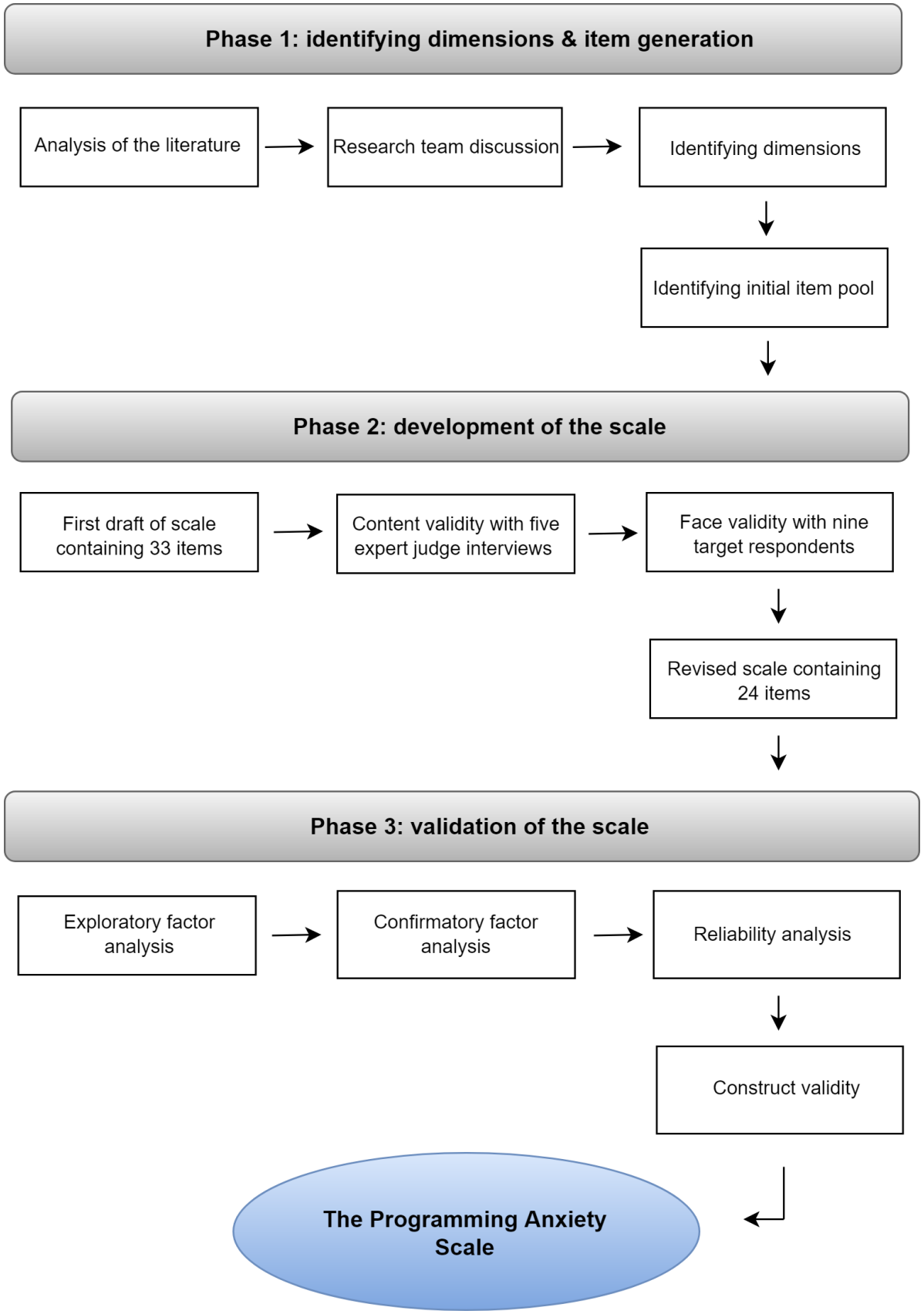

Figure 2. An overview of phases of the PAS development. (Adapted from Zarouali, Boerman & de Vreese, 2021)

Table 2. Item Pool of the Draft Scale

Factor	Items Code	Item
Classmates	Item1	I am concerned about not being able to stay calm like my classmates while coding a program.
	Item2	I feel humiliated if a classmate easily debugs an error on which I worked so hard.
	Item3	Believing that I cannot reach the level of my friends who have taken programming lessons before makes me anxious.
	Item4	The presence of my classmates who have previously taken programming courses makes me nervous.
	Item5	It makes me anxious when many of my classmates can write the code that I cannot write.
	Item6	I get worried if my classmates comprehend a programming topic and I don't.
	Item7	I feel tense when my friends talk about programming topics that I don't understand.
	Item8	I am concerned about not being able to write a program and being ridiculed by my classmates.
Self-confidence	Item9	I think I do not understand programming well.
	Item10	It makes me anxious to feel that I memorize programming topics instead of learning the logic.
	Item11	It makes me anxious to feel that I quickly forget what I have learned in programming lessons.
	Item12	I have doubts about creating the steps (algorithm) necessary for the solution while coding the program.
	Item13	I have concerns about my programming abilities.
	Item14	I feel confused when the program lines become complicated.
	Item15	I don't trust myself in writing programs.
	Item16	I get nervous when we talk about programming.
	Item17	It scares me that there are too many topics to learn in the programming lesson.
Errors	Item18	I get worried when I can not understand error messages.
	Item19	The number of errors in my program makes me worried.
	Item20	I am worried about encountering errors in my programs.
	Item21	I feel worried when my program fails to run.
	Item22	Debugging programs is a major worry for me.
	Item23	I get worried about debugging my software over and over again.
	Item24	It makes me worried to think that my codes will have bugs.

2.3 Phase 3: Validation of the scale

2.3.1 Sample

The draft scale was validated primarily in two phases. Since using the same data set for exploratory factor analysis (EFA) and confirmatory factor analysis (CFA) is not generally accepted as the correct method in the literature (Fokkema & Greiff, 2017), data were collected from two distinct sample groups of university students (first and second sample). While the data from the first sample was used to investigate the scale's underlying factor structure in the EFA phase, the data from the second sample was used in the CFA phase to cross-validate the EFA results. All subjects were recruited using a convenience sampling method.

The first sample comprised 392 university students (29 females, 363 males) taking programming courses at two different universities in Turkey. The female and male participants' mean age was 20.30 and 19.46 years, respectively. The first sample consists of students from the Department of Electronics Technology (82.4), Computer Education and Instructional Technology (CEIT) (10.2%), and Computer Technology (7.4). While 81.89% (n=321) of the students were experienced in a programming, 18.11% (n=71) were novices.

The second sample consisted of 295 college students recruited voluntarily from different programs (32 females, 263 males). The female and male participants' mean age was 21.12 and 19.89 years, respectively. The second sample consists of students from the Department of Electronics Technology (69.04), CEIT (16.01%), and Computer Technology (14.95). While 76.61% (n=226) of the students were experienced, 23.39% (n=69) had no prior programming experience. Data from both samples were collected using Google Forms.

2.3.2 Analytical Strategy

The PAS was validated using Boateng, Neilands, Frongillo, Melgar-Quiñonez, & Young's (2018) scale development recommendations. The number and character of the underlying components in the scale were determined using EFA, which CFA followed to ensure that the data fit the retrieved factor structure. The construct validity was then examined after a reliability analysis. The Kaiser-Meyer-Olkin Measure of Sampling Adequacy (KMO) and Bartlett's test were implemented to assess the adequacy of the study group (Tabachnick & Fidell, 2001). The tests showed that the data was suitable for EFA. In addition, considering Hair, Black, Babin, & Anderson's (2010) suggestions on the cutoff value for factor loadings and commonalities, these values were determined as .50 and .30, respectively. Items that loaded only one factor without any cross-loadings were kept. The research employed SPSS 22 software for EFA. For CFA, Maximum likelihood estimation was adopted to calculate the structure parameters using AMOS 22 software.

The internal consistency coefficient (Cronbach's alpha) was calculated for the full scale and each dimension for reliability analysis. For convergent validity, the factor loading of the indicator, the average variance extracted (AVE), composite reliability (CR), and maximum share variance (MSV) values were calculated (Hair, Hult, Ringle, & Sarstedt, 2014). These calculations were conducted using Gaskin's (2016) AMOS MasterValidity Plugin. Furthermore, convergent validity was assessed through (1) comparison of mean values of factors and total programming anxiety depending on gender and (2) correlation analysis of factors, total programming anxiety, and course grade of students. The Fornell & Larcker criterion and the Heterotrait-Monotrait (HTMT) correlation ratio were utilized to examine discriminant validity (Ab Hamid, Sami & Sidek, 2017). HTMT was calculated using Excel 2016 software. Results of EFA, CFA, reliability, and validity analysis were presented in the *manuscript's results section*.

3. Results

3.1 Exploratory Factor Analysis

The data's appropriateness for factor analysis was assessed before doing EFA. For this reason, Mahalanobis distance values for probable multivariate outliers were determined. Thirty-nine instances were eliminated from the analysis because their Mahalanobis values were above the necessary chi-square value of 54.05 (df = 26, alpha=.001) (Pallant, 2007). In addition, KMO was used to determine the adequacy of the sample size, and Bartlett's Test of Sphericity was used to determine whether the datum was suitable for factor analysis. The KMO sampling adequacy metric was .951, higher than the acceptable value of .60. Barlett's Test of Sphericity was also statistically significant (x^2=3790.593, p=.000), demonstrating that the data was considerably factorable (Pallant, 2007).

In the first stage of EFA, principal axis factoring was employed for 24 items with direct oblimin rotation. The

direct oblimin rotation approach was chosen because of the associated factors (Costello & Osborne, 2005; Gorsuch, 1983). After the EFA process, the correlation matrix was investigated for multicollinearity issues. Correlations in the .80's or .90's (Field, 2018) were examined, and six items (Item18, Item19, Item20, Item21, Item23, and Item24) in the "errors" factor with a correlation coefficient greater than .8 were excluded from the scale. Next, EFA was conducted again for the remaining 18 items. Four items (Item2 and Item6 in classmates and Item13 and Item16 in self-confidence) with high factor loadings in multiple factors were excluded (Burns & Machin, 2009; Howard, 2016). EFA was executed with the remaining 14 items. Based on the cutoffs for the eigenvalues and inspection of the scree plot, a two-factor model was identified that explained 69.4% of the variance in programming anxiety. The two-factor model was confirmed by a parallel analysis with 5000 randomly generated data matrices through Parallel Analysis Web Application (Patil, Surendra, Sanjay, & Donavan, 2017). These factors were labeled: (1) Classmates and (2) Self-Confidence. Table 3 includes the complete list of factor loadings. Internal consistency reliabilities (i.e., Cronbach's alpha coefficient) for the full scale and the subscales were .95, .90, and .94, respectively.

Table 3. Factor Loadings of the PAS

Items Code	Item	Classmates	Self-Confidence
Item1	I am concerned about not being able to stay calm like my classmates while coding a program.	.633	
Item3	Believing that I cannot reach the level of my friends who have taken programming lessons before makes me anxious.	.806	
Item4	The presence of my classmates who have previously taken programming courses makes me nervous.	.918	
Item5	It makes me anxious when many of my classmates can write the code that I cannot write.	.643	
Item7	I feel tense when my friends talk about programming topics that I don't understand.	.634	
Item8	I am concerned about not being able to write a program and being ridiculed by my classmates.	.877	
Item9	I think I do not understand programming well.		.818
Item10	It makes me anxious to feel that I memorize programming topics instead of learning the logic.		.726
Item11	It makes me anxious to feel that I quickly forget what I have learned in programming lessons.		.851
Item12	I have doubts about creating the steps (algorithm) necessary for the solution while coding the program.		.844
Item14	I feel confused when the program lines become complicated.		.909
Item15	I don't trust myself in writing programs.		.780
Item17	It scares me that there are too many topics to learn in the programming lesson.		.852
Item22	Debugging programs is a major worry for me.		.822

3.2 Confirmatory Factor Analysis

The 14-item Programming Anxiety Scale's two-factor model was subjected to CFA using second sample data. Before the analysis, data were screened for missing values and outliers. With this respect, 13 respondents were detected unengaged in the scale evidence by getting the same response for every item. Thus, 14 cases were

deleted from the sample. Next, CFA was conducted with 282 samples using maximum likelihood estimation. After CFA, three items (i.e., Item1, Item9, and Item11) had standardized parameter estimates smaller than the recommended value of .50 (Hair et al., 2010).

Furthermore, AVE values for factors below .50 were calculated, indicating the absence of convergent validity. After removing these three items, 11 item structure of the PAS was re-subjected to CFA. The diagram regarding the factor structure of programming anxiety with new item codes (See Appendix) and the parameter estimates was given in Figure 2.

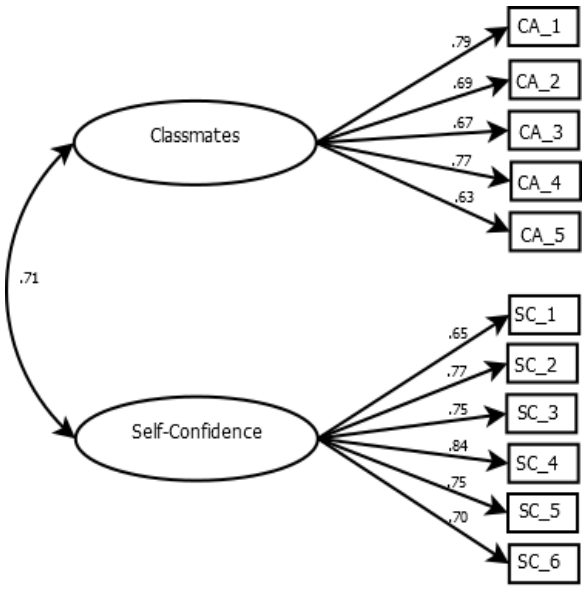

Figure 2. Standardized Coefficients for the Two-Factor Model of PAS

According to Hu and Bentler (1999), researchers employ numerous goodness-of-fit metrics to analyze a model. In this study, the Chi-square goodness of fit test, Root Mean Square of Error of Approximation (RMSEA), Standardized Root Mean Square Residuals (SRMR), Goodness of Fit Index (GFI), Normative Fit Index (NFI), and Comparative Fit Index (CFI) were employed to assess model fit (Brown, 2006; Browne & Cudeck, 1993; Hair et al., 2010; Kline, 1998). Table 4 presents the fit statistics for the confirmative factor analysis.

Table 4. Values of the Goodness-of-Fit Test for Programming Anxiety

X^2	X^2/df	p-value	RMSEA	SRMR	GFI	NFI	CFI
114.226*	2.79	.000	.071	0.032	.93	.93	.95

* $p<0.01$

When Table 4 was analyzed, the chi-square value (X^2 = 114.226, X^2/ df = 2.79, p = .000) was found to be significant. RMSEA value of .071 indicate good adaptation (Brown, 2006; Browne & Cudeck, 1993). GFI, CFI, and NFI values greater than .90 indicate a good fit (Hair et al., 2010; Kline, 1998). The CFA results indicate that the structural model is a good fit. As shown in Fig. 2., each item loaded significantly on its particular dimension and was relatively large (.50 and above).

3.3 Assessment of Reliability and Validity

To assess convergent validity, standardized factor loadings, CR, and AVE were calculated (Hair et al., 2010). Cronbach alpha values for factors and the whole scale were calculated for reliability analysis (Taber, 2018). The condition that factor loading values greater than .5, Cronbach alpha values greater than .7, and AVE and CR values greater than .5 and .7 were taken into account (Taber, 2018; Hair et al., 2010). Table 5 illustrates that all constructs are reliable since they fulfill the above criteria. Furthermore, each construct's Cronbach's alpha exceeds the recommended value. The Cronbach's alpha value of the whole scale was calculated as .901, which

fulfilled the reliability criterion.

Table 5. Evaluation of the Measurement Model

Factors	CR	AVE	MSV	Cronbach Alpha
Classmates	.835	.506	.498	.843
Self-Confidence	.881	.554	.498	.882

Furthermore, convergent validity was assessed through (1) comparison of mean values of factors and total programming anxiety depending on gender (Table 6) and (2) correlation analysis of factors, total programming anxiety, and course grade of students (Table 7). As shown in Table 6, female students had more programming anxiety than male respondents. Gender differences in programming anxiety were also investigated using a t-test. The test results showed that while there was no significant difference between male and female students in self-confidence factor ($t = .767$, $p>.05$), there was a significant difference in classmates factor ($t =4.058$, $p=.000$) and total programming anxiety ($t = 2.518$, $p=.012$). Correlation analysis results (Table 7) revealed that classmates ($r=-.194$) and self-confidence ($r=-.315$) factors, as well as total programming anxiety ($r=-.284$), were negatively associated with course grade, $p<.01$. All of these correlations were found as weak (Schober, Boer & Schwarte, 2018; Senthilnathan, 2019).

Table 6. Gender differences in programming anxiety

Gender		Factors				Total Programming Anxiety	
		Classmates		Self-Confidence			
	N	M	SD	M	SD	M	SD
Female	30	16.00	4.69	16.87	4.75	32.87	8.45
Male	252	11.96	5.19	15.96	6.27	27.91	10.34

Table 7. Correlation analysis results

	Classmates	Self-Confidence	Total Programming Anxiety	Course Grade
Classmates	—			
Self-Confidence	.688**	—		
Total Programming Anxiety	.898**	.938**	—	
Course Grade	-.194**	-.315**	-.284**	—

**. Correlation is significant at the 0.01 level (2-tailed).

The Fornell & Larcker and the HTMT criteria were used to test the scale's discriminant validity. Table 8 summarizes the results of the Fornell & Larcker criteria. In Table 8, each AVE's square root was given on the diagonal, and the correlation coefficients (off-diagonal) for each construct were displayed in the corresponding rows and columns. Fornel and Larcker (1981) state that the AVE values' square root should be higher than the correlations between the components included in the analysis. As shown in Table 8, this condition was satisfied, and the model met the Fornell & Larcker criterion for discriminant validity. In addition to the Fornell & Larcker criterion, discriminant validity was assessed through the HTMT coefficient. The HTMT coefficient was calculated as .705 in this model. According to Henseler, Ringle, and Sarstedt (2015), the HTMT coefficient should be less than .90 if the components to be evaluated are hypothetically close to one other. The HTMT coefficient was found to be below the threshold levels.

Table 8. The square root of the average variance extracted (AVE) and correlations matrix

	Factors	
Factors	Classmates	Self-Confidence
Classmates	**.711**	
Self-Confidence	.705	**.745**

4. Discussion and Conclusion

In computing education research, accurate measurement is critical (Scott, 2015). On the other hand, few measurement tools are available to computer education researchers (Scott & Ghinea, 2014). The current research fills a gap in the literature by developing and validating a measurement tool to assess the computer programming anxiety of university students. The development and validation of the programming anxiety scale in the current study were carried out in harmony with the scale studies recently published in several fields (Nasir, Adil, & Kumar, 2021; Rosario-Hernández, Rovira-Millán, & Blanco-Rovira, 2022; Sun et al., 2022; Zarouali, Boerman, & de Vreese, 2021). EFA, DFA, reliability, and validity analysis resulted in a scale including 11 items, five items for classmates, and six for self-confidence. The minimum obtainable score from the Programming Anxiety Scale is 11, while the maximum score is 55. As the score obtained from the scale increases, programming anxiety also increases. Choo and Cheung's (1991) Computer Programming Anxiety Scale served as a guide to develop the present scale.

In the current study, no factor related to errors was found, while there was a factor for error anxiety in the study of Choo & Cheung (1991) and Demir (2021), in which Choo & Cheung's (1991) scale was adapted into Turkish. Although seven items were included in the draft scale for this factor, six of these items showed high multicollinearity. They were removed from the draft scale due to the exploratory factor analysis, and one item was kept. Although the inclusion of an item about error anxiety shows that debugging is a source of anxiety for students (Dasuki & Quaye, 2016; Nolan & Bergin, 2016), it is worth examining why it is not included as a factor. Not having an "errors" factor may indicate that encountering errors is a source of anxiety regardless of the number of errors and the time spent for debugging, which was utilized as parameters in Choo & Cheung (1991) and Demir (2021). In other words, encountering errors in programs may exist as a single source of anxiety, regardless of the frequency of encountering errors, the number of errors, or the time it takes to debug. The fact that Demir's (2021) study does not contain any information about CFA makes it impossible to make inferences about whether the three-factor model shows a good fit in the Turkish version of the scale and make comparisons about the error factor.

Within the scope of the current study, the only item related to error anxiety was included in the self-confidence factor. One of the reasons for this result may be the differences in perception of debugging as a process in programming activities between the novices and the relatively more experienced individuals. From this point of view, while debugging may be perceived as an independent process for novice programmers, debugging may be perceived as an integral process of programming and an element of self-efficacy perception. Considering a high degree of relationship between self-confidence and self-efficacy perception (Blanco et al., 2020; Malureanu, Panisoara & Lazar, 2021; Tsai, 2019), it is not surprising that an item in the error factor is included in the self-confidence factor. It is evident that enhanced debugging skills develop a programmer's confidence, and fear of making mistakes may be related to programming skills (Ahmadzadeh, Elliman, & Higgins, 2005; Connolly et al., 2009; Nolan & Bergin, 2016). From this point of view, the experience of the participant group of Choo & Cheung's (1991) study on programming was not explained in detail. The participant group was specified only as of grade 12 level. However, participants in both the EFA and CFA stages of the current study were relatively more experienced with programming than the participants in Choo & Cheung's (1991) study. They have developed at least one project and were familiar with the debugging process. This result may indicate that perceptions of debugging are related to programming experience.

Another reason may be the attitudes of the participants towards programming. Choo & Cheung's (1991) participant group consisted of grade 12 junior high school students. In contrast, the current study participants comprised university students who perceived programming as a profession. The students' awareness that the programming profession will include debugging may have caused them to perceive debugging as a personal

competence and an aspect of their career. The last reason for this result may be the software development environments (Integrated Development Environments-IDEs) and the resources and materials that can be facilitated for debugging. Scratch was used in Demir (2021) as the program development environment. On the contrary, Visual Studio was used in the current study. The nature of the errors encountered in Scratch and Visual Studio shows differences. The IDE used by Choo & Cheung (1991) was not specified. However, the number of resources found on debugging in the 1990s and today's resources are very different. Today, the internet is used for interpretations of error messages, and previous experiences of other people are utilized for solutions. Nowadays, even IDEs that translate error messages into users' native language exist. Therefore, the characteristics of IDEs and the resources may alter the perception of debugging anxiety.

The validity of the developed scale was tested by comparing the results of studies examining the relationship between programming anxiety and theoretically related variables in the literature. In the current study, female students had more programming anxiety than male respondents. The t-test result also revealed that while there was no significant difference between male and female students in self-confidence, there was a substantial difference in classmates factor and total programming anxiety. This result is consistent with Olipas and Luciano's (2020) study. Similarly, consistent with the former findings ((Connolly et al., 2007; Figueroa & Amoloza, 2015; Kinnunen & Malmi, 2006; Nolan et al., 2019; Owolabi et al., 2014; Scott, 2015), factors of the PAS and total programming anxiety was correlated with course grades of students. These results indicate that the developed scale is valid and reliable.

Another issue is related to the fact that this scale was developed specifically for programming anxiety. Since this scale was designed specifically to measure programming anxiety, it differs from scales adapted to programming anxiety, such as computer anxiety, computer attitude, and IT anxiety (Chang, 2005; Olipas & Luciano, 2020; Owolabi et al., 2014). There were insufficient instruments to assess programming anxiety in the literature, and the current study offered a psychometrically reliable scale. With the help of the present scale, programming anxiety levels in students can be measured, methods and techniques that can reduce students' anxiety can be developed, and special attention can be paid to students with high programming anxiety. In addition, situations that increase programming anxiety in students can be investigated. The PAS developed in the current study is recommended for study groups that have somewhat experience in creating, coding, and debugging programming projects. Future research may concentrate on a different group of college students from various cultures and countries.

Acknowledgments

This study was conducted in the framework of Osman Gazi YILDIRIM's Ph.D. dissertation at Marmara University/TURKEY, supervised by Professor Nesrin OZDENER DONMEZ.

References

Ab Hamid, M. R., Sami, W., & Sidek, M. M. (2017, September). Discriminant validity assessment: Use of Fornell & Larcker criterion versus HTMT criterion. In *Journal of Physics: Conference Series* (Vol. 890, No. 1, p. 012163). IOP Publishing.

Ahmadzadeh, M., Elliman, D., & Higgins, C. (2005). An analysis of patterns of debugging among novice computer science students. *In Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education* (pp. 84-88).

Al-Makhzoomy, A. K. (2018). *Effect of Game Development-Based Learning on the Ability of Information Technology Undergraduates to Learn Computer and Object-Oriented Programming* (Doctoral dissertation, Wayne State University).

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGcSE Bulletin, 39*(2), 32-36.

Bergin, S., & Reilly, R. (2005). Programming: factors that influence success. *In Proceedings of the 36th SIGCSE technical symposium on Computer science education* (pp. 411-415).

Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning (IJGBL), 1*(2), 65-81.

Blanco, Q. A., Carlota, M. L., Nasibog, A. J., Rodriguez, B., Saldaña, X. V., Vasquez, E. C., & Gagani, F. (2020).

Probing on the relationship between students' self-confidence and self-efficacy while engaging in online learning amidst COVID-19. *Journal La Edusci*, *1*(4), 16-25.

Boateng, G. O., Neilands, T. B., Frongillo, E. A., Melgar-Quiñonez, H. R., & Young, S. L. (2018). Best practices for developing and validating scales for health, social, and behavioral research: a primer. *Frontiers in public health*, *6*, 149.

Bosch, N., D'Mello, S., & Mills, C. (2013, July). What emotions do novices experience during their first computer programming learning session?. *In International Conference on Artificial Intelligence in Education* (pp. 11-20). Springer, Berlin, Heidelberg.

Brown, T. A. (2006). *Confirmatory factor analysis for applied research*. New York: Guilford Publications.

Browne, M. W., & Cudeck, R. (1992). Alternative ways of assessing model fit. *Sociological methods & research, 21*(2), 230-258.

Bunderson, E. D., & Christensen, M. E. (1995). An analysis of retention problems for female students in university computer science programs. *Journal of Research on Computing in Education, 28*(1), 1-18.

Burns, R. A., & Machin, M. A. (2009). Investigating the structural validity of Ryff's psychological well-being scales across two samples. *Social indicators research*, *93*(2), 359-375.

Butcher, D. F., & Muth, W. A. (1985). Predicting performance in an introductory computer science course. *Communications of the ACM, 28*(3), 263-268.

Byrne, P., & Lyons, G. (2001, June). The effect of student attributes on success in programming. *In Proceedings of the 6th annual conference on Innovation and technology in computer science education* (pp. 49-52).

Campbell, S. L. (2018). *An evaluation of an automated, interactive learning method for a database query language*. The University of Iowa.

Carpenter, S. (2018). Ten steps in scale development and reporting: A guide for researchers. *Communication Methods and Measures*, *12*(1), 25-44.

Chang, S. E. (2005). Computer anxiety and perception of task complexity in learning programming-related skills. *Computers in Human Behavior, 21*(5), 713-728.

Choo, M. L., & Cheung, K. C. (1991). On meaningful measurement: Junior college pupils' anxiety towards computer programming. *Journal of Educational Technology Systems, 19*(4), 327-343.

Clark, L. A., & Watson, D. (1995). Constructing validity: Basic issues in objective scale development. *Psychological Assessment, 7*(3), 309–319

Connolly, C., Murphy, E., & Moore, S. (2007). Second chance learners, supporting adults learning computer programming. *In international conference on engineering education–ICEE*.

Connolly, C., Murphy, E., & Moore, S. (2009). Programming anxiety amongst computing students—A key in the retention debate?. *IEEE Transactions on Education, 52*(1), 52-56.

Costello, A. B., & Osborne, J. (2005). Best practices in exploratory factor analysis: Four recommendations for getting the most from your analysis. *Practical assessment, research, and evaluation, 10*(1), 7.

Dasuki, S., & Quaye, A. (2016). Undergraduate students' failure in programming courses in institutions of higher education in developing countries: A Nigerian perspective. *The Electronic Journal of Information Systems in Developing Countries*, *76*(1), 1-18.

Demir, F. (2021). The effect of different usage of the educational programming language in programming education on the programming anxiety and achievement. *Education and Information Technologies*, 1-24.

Demirer, V. & Sak, N. (2016). Dünyada ve Türkiye'de programlama eğitimi ve yeni yaklaşımlar. *Eğitimde Kuram ve Uygulama, 12*(3), 521-546.

Dijkstra, E. W. (1989). On the cruelty of really teaching computing science. *Communications of the ACM, 32*(12), 1398-1404.

Falkner, K., Falkner, N. J., & Vivian, R. (2013, March). Collaborative Learning and Anxiety: A

phenomenographic study of collaborative learning activities. *In Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 227-232).

Field, A. (2018). *Discovering Statistics Using IBM SPSS Statistics*. Sage

Figueroa Jr, R. B., & Amoloza, E. M. (2015). Addressing Programming Anxiety among Non-Computer Science Distance Learners: A UPOU Case Study. *International Journal, 9*(1), 56-67.

Fokkema, M., & Greiff, S. (2017). How performing PCA and CFA on the same data equals trouble. *European Journal of Psychological Assessment*.

Fornell, C., & Larcker, D. F. (1981). Evaluating structural equation models with unobservable variables and measurement error. *Journal of marketing research*, *18*(1), 39-50.

Gaskin, J. (2016), "MasterValidity", Gaskination's Statistics. http://statwiki.gaskination.com

Gibbs, D. C. (2000). The effect of a constructivist learning environment for field-dependent/independent students on achievement in introductory computer programming. *ACM SIGCSE Bulletin, 32*(1), 207-211.

Gorsuch, R. L. (1983). *Factor analysis (2nd Edition*). Hillsdale, NJ: Erlbaum.

Hair, J., Black, W., Babin, B., & Anderson, R. (2010). *Multivariate data analysis (7th ed.).* Upper Saddle River, NJ: Prentice-Hall.

Hair, J. F., Hult, G. T. M., Ringle, C., & Sarstedt, M. (2014). A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM).

Hattie, J. (2008). Visible learning: A synthesis of over 800 meta-analyses relating to achievement. Routledge.

Henseler, J., Ringle, C. M., & Sarstedt, M. (2015). A new criterion for assessing discriminant validity in variance-based structural equation modeling. *Journal of the academy of marketing science*, *43*(1), 115-135.

Hostetler, T. R. (1983). Predicting student success in an introductory programming course. ACM SIGCSE Bulletin, 15(3), 40-43.

Howard, M. C. (2016). A review of exploratory factor analysis decisions and overview of current practices: What we are doing and how can we improve? *International Journal of Human-Computer Interaction*, *32*(1), 51-62.

Hsu, W. C., & Gainsburg, J. (2021). Hybrid and Non-Hybrid Block-Based Programming Languages in an Introductory College Computer-Science Course. *Journal of Educational Computing Research*, *59*(5), 817-843.

Hu, L. T., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural equation modeling: a multidisciplinary journal, 6*(1), 1-55.

Huggard, M. (2004). Programming trauma: can it be avoided. *Proceedings of the BCS Grand Challenges in Computing: Education*, 50-51.

Jenkins, T. (2002). On the difficulty of learning to program. *In Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences* (Vol. 4, No. 2002, pp. 53-58).

Jiang, Y., Zhao, Z., Wang, L., & Hu, S. (2020, August). Research on the Influence of Technology-Enhanced Interactive Strategies on Programming Learning. *In 2020 15th International Conference on Computer Science & Education (ICCSE)* (pp. 693-697). IEEE.

Jones, P. S., Lee, J. W., Phillips, L. R., Zhang, X. E., & Jaceldo, K. B. (2001). An adaptation of Brislin's translation model for cross-cultural research. *Nursing research, 50*(5), 300-304.

Kert, S. B. & Uğraş, T. (2009). Programlama eğitiminde sadelik ve eğlence: Scratch örneği. In *The First International Congress of Educational Research, Çanakkale, Turkey*.

Kinnunen, P., & Malmi, L. (2006, September). Why students drop out CS1 course?. *In Proceedings of the second international workshop on Computing education research* (pp. 97-108).

Kinnunen, P., & Simon, B. (2012). My program is ok–am I? Computing freshmen's experiences of doing programming assignments. *Computer Science Education, 22*(1), 1-28.

Kline, R. B. (1998). *Principles and practice of structural equation modeling*. New York: Guildford

Luxton-Reilly, A., Ajanovski, V. V., Fouh, E., Gonsalvez, C., Leinonen, J., Parkinson, J., ... & Thota, N. (2019). Pass rates in introductory programming and in other stem disciplines. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education* (pp. 53-71).

Lynn, M. R. (1986). Determination and quantification of content validity. *Nursing research*.

Maguire, P., Maguire, R., & Kelly, R. (2017). Using automatic machine assessment to teach computer programming. *Computer Science Education, 27*(3-4), 197-214.

Malureanu, A., Panisoara, G., & Lazar, I. (2021). The relationship between self-confidence, self-efficacy, grit, usefulness, and ease of use of elearning platforms in corporate training during the COVID-19 pandemic. *Sustainability*, *13*(12), 6633.

Nasir, M., Adil, M., & Kumar, M. (2021). Phobic COVID-19 disorder scale: Development, dimensionality, and item-structure test. *International Journal of Mental Health and Addiction*, 1-13.

Nolan, K., & Bergin, S. (2016, November). The role of anxiety when learning to program: a systematic review of the literature. In *Proceedings of the 16th koli calling international conference on computing education research* (pp. 61-70).

Nolan, K., Bergin, S., & Mooney, A. (2019, September). An Insight Into the Relationship Between Confidence, Self-efficacy, Anxiety and Physiological Responses in a CS1 Exam-like Scenario. In *Proceedings of the 1st UK & Ireland Computing Education Research Conference* (pp. 1-7).

Nunnally, J. C. (1994). *Psychometric theory* 3E. Tata McGraw-Hill Education.

Olipas, C. N. P., & Luciano, R. G. (2020). Understanding The Impact Of Using Countdown Timer On The Academic Motivation And Computer Programming Anxiety Of IT Students: The Case Of A State University In The Philippines. *International Journal of Scientific and Technology Research*, *9*.

Olipas, C. N. P., Leona, R. F., Villegas, A. C. A., Cunanan Jr, A. I., & Javate, C. L. P. (2021). The Academic Performance and the Computer Programming Anxiety of BSIT Students: A Basis for Instructional Strategy Improvement.

Orehovacki, T., Radosevic, D., & Konecki, M. (2012, June). Acceptance of Verificator by information science students. In Proceedings of the ITI 2012 34th *International Conference on Information Technology Interfaces* (pp. 223-230). IEEE.

Owolabi, J., Olanipekun, P., & Iwerima, J. (2014). Mathematics ability and anxiety, computer and programming anxieties, age and gender as determinants of achievement in basic programming. GSTF Journal on Computing (JoC), 3(4), 109.

Özmen, B., & Altun, A. (2014). Undergraduate Students' Experiences in Programming: Difficulties and Obstacles. *Turkish Online Journal of Qualitative Inquiry, 5*(3).

Pajares, M. F. (1992). Teachers' beliefs and educational research: Cleaning up a messy construct. *Review of educational research, 62*(3), 307-332.

Pallant, J. (2007). *SPSS survival manual: A step by step guide to data analysis using SPSS version 15* (3rd ed.). New York: Open University Press.

Patil Vivek H, Surendra N. Singh, Sanjay Mishra, and D. Todd Donavan (2017). *Parallel Analysis Engine to Aid in Determining Number of Factors to Retain using R [Computer software],* available from https://analytics.gonzaga.edu/parallelengine/.

Polit, D. F., Beck, C. T., & Owen, S. V. (2007). Is the CVI an acceptable indicator of content validity? Appraisal and recommendations. *Research in nursing & health*, *30*(4), 459-467.

Ramalingam, V., & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research, 19*(4), 367-381.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion.

Computer science education, 13(2), 137-172.

Rogerson, C., & Scott, E. (2010). The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education: Research, 9*(1), 147-171.

Rosario-Hernández, E., Rovira-Millán, L. V., & Blanco-Rovira, R. A. (2022). Development and Validation of the Job Satisfaction Brief Scale. *Revista Caribeña de Psicología*, e6191-e6191.

Rountree, N., Rountree, J., & Robins, A. (2002). Predictors of success and failure in a CS1 course. *ACM SIGCSE Bulletin, 34*(4), 121-124.

Santos, S. C., Tedesco, P. A., Borba, M., & Brito, M. (2020). Innovative approaches in teaching programming: A systematic literature review. In *Proceedings of the 12th International Conference on Computer Supported Education* (Vol. 1, pp. 205-214).

Schober, P., Boer, C., & Schwarte, L. A. (2018). Correlation coefficients: appropriate use and interpretation. *Anesthesia & Analgesia*, *126*(5), 1763-1768.

Scott, M. (2015). *Self-Beliefs in the Introductory Programming Lab and Games-based Fantasy Role-Play* (Doctoral dissertation, Brunel University).

Scott, M. J., & Ghinea, G. (2014, July). Measuring enrichment: the assembly and validation of an instrument to assess student self-beliefs in CS1. *In Proceedings of the tenth annual conference on International computing education research* (pp. 123-130).

Senthilnathan, S. (2019). Usefulness of correlation analysis. *Available at SSRN 3416918*.

Sinožić, S., & Orehovački, T. (2018, May). Using analytic hierarchy process to select the most appropriate tool for learning programming. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 0524-0529). IEEE.

Sun, J., Jiang, X., Gao, Y., He, C., Wang, M., Wang, X., ... & Zhang, L. (2022). Subhealth Risk Perception Scale: Development and Validation of a New Measure. *Computational and Mathematical Methods in Medicine*, 2022.

Tabachnick, B. G., & Fidell, L. S. (2001). *Using multivariate statistics (4th ed.)*. Needham Heights, MA: Allyn & Bacon

Taber, K. S. (2018). The use of Cronbach's alpha when developing and reporting research instruments in science education. *Research in science education*, *48*(6), 1273-1296.

Tan, P. H., Ting, C. Y., & Ling, S. W. (2009, November). Learning difficulties in programming courses: undergraduates' perspective and perception. In *2009 International Conference on Computer Technology and Development* (Vol. 1, pp. 42-46). IEEE.

Tsai, C. Y. (2019). Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy. *Computers in Human Behavior*, *95*, 224-232.

Watson, C., & Li, F. W. (2014, June). Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 39-44).

Wiedenbeck, S., Labelle, D., & Kain, V. N. (2004). Factors affecting course outcomes in introductory programming. *In PPIG* (p. 11).

Wilfong, J. D. (2006). Computer anxiety and anger: The impact of computer use, computer experience, and self-efficacy beliefs. *Computers in human behavior*, *22*(6), 1001-1011.

Wilson, B. C., & Shrock, S. (2001). Contributing to success in an introductory computer science course: a study of twelve factors. *ACM SIGCSE Bulletin, 33*(1), 184-188.

Zarouali, B., Boerman, S. C., & de Vreese, C. H. (2021). Is this recommended by an algorithm? The development and validation of the algorithmic media content awareness scale (AMCA-scale). *Telematics and Informatics*, *62*, 101607.

Appendix

The Programming Anxiety Scale in its Original and Translated Form

Original Items (in Turkish)	Translated Items (in English)
CA_1 Daha önce programlama dersi alan arkadaşlarımın seviyesine yetişemeyeceğime inanmak beni kaygılandırır.	Believing that I cannot reach the level of my friends who have taken programming lessons before makes me anxious.
CA_2 Daha önce programlama dersi alan sınıf arkadaşlarımın varlığı beni tedirgin eder.	The presence of my classmates who have previously taken programming courses makes me nervous.
CA_3 Benim yazamadığım bir kodu çoğu sınıf arkadaşım yazabilmesi beni kaygılandırır.	It makes me anxious when many of my classmates can write the code that I cannot write.
CA_4 Arkadaşlarım benim anlamadığım programlama konularında konuştuğunda gergin hissederim.	I feel tense when my friends talk about programming topics that I don't understand.
CA_5 Program yazamayıp sınıf arkadaşlarımın önünde gülünç duruma düşeceğimden endişelenirim.	I am concerned about not being able to write a program and being ridiculed by my classmates.
SC_1 Programlamayı iyi anlayamadığımı düşünürüm.	I think I do not understand programming well.
SC_2 Program yazarken çözüm için gerekli olan basamakları (algoritmayı) doğru oluşturabileceğim konusunda şüphelerim var.	I have doubts about creating the steps (algorithm) necessary for the solution while coding the program.
SC_3 Program satırları karmaşık olmaya başladığında aklımın karıştığını hissederim.	I feel confused when the program lines become complicated.
SC_4 Program yazma konusunda kendime güvenmem.	I don't trust myself in writing programs.
SC_5 Programlama dersinde öğrenilecek çok fazla konunun olması beni korkutur.	It scares me that there are too many topics to learn in the programming lesson.
SC_6 Programlarımda hatalarla karşılaşmak benim için büyük bir endişe kaynağıdır.	Debugging programs is a major worry for me.