Volume 7, No: 3
February 2026
ISSN 2513-8359

International Journal of Computer Science Education in Schools

Editors

Dr Filiz Kalelioglu

Dr Yasemin Allsop

www.ijcses.org

International Journal of Computer Science Education in Schools

Nov 2026, Vol 7, No 3

DOI: 10.21585/ijcses.v7i3

Table of Contents

International Journal of Computer Science Education in Schools, February 2026, Vol. 7, No. 3
ISSN 2513-8359

Teaching Computer Science: Analysing the Key Factors Affecting Educators' Professional Motivation[1]

Özcan TOY[1]

Serhat Bahadır KERT[2]

[1]**Yıldız Technical University, Türkiye, ozcantoy@gmail.com**

[2]**Yıldız Technical University, Türkiye, sbkert@yildiz.edu.tr**

DOI: 10.21585/ijcses.v7i3.243

Abstract

Computer science education is essential and presents both pedagogical and technological problems. Computer science educators must possess a passion for computing and education. This study investigated the professional motivation of computer science educators. A robust and reliable scale has been developed to evaluate the elements influencing the professional motivation of computer science instructors. The study used a quantitative correlational survey model. The scale was created utilizing data from 798 computer science scholars across Turkey's provinces. Data was gathered in three stages. The Exploratory Factor Analysis (EFA) involved 246 instructors, the Confirmatory Factor Analysis (CFA) included 366 teachers, and the final application encompassed 186 teachers. The data analysis software utilized was SPSS version 25.0 and AMOS version 24.0. The findings indicate that CFA was employed to examine a structure comprising 18 elements and two factors. The results indicated that instructors' motivation did not significantly vary based on gender, alma mater, years of experience, or location of assignment. A notable disparity was detected in the management factor based on the educational level of the teachers (primary, secondary, or high school). Independent samples t-tests revealed no significant difference in motivation scores based on gender ($t[184]=.102$; $p>0.05$). ANOVA results indicated no significant differences based on years of professional experience ($p=0.068$; $p<0.05$) or city of assignment ($p=0.199$; $p<0.05$). ANOVA indicated a significant impact of educational level ($p=0.058$; $p<0.05$) on the management-based factor. Post hoc comparisons (Tukey HSD) revealed that high school teachers exhibited considerably greater management-related motivation than their counterparts at the primary and secondary levels.

Keywords: Computer science, teacher, motivation, professional motivation, exploratory factor analysis (EFA), confirmatory factor analysis (CFA), ANOVA.

[1] This research has been derived from the first author's master thesis

1. Introduction

In the last thirty years, computer science has become an essential discipline in technology and education (Popovich et al., 2008). The widespread use of computers is increasingly viewed as a vital skill in the digital age, requiring the integration of computer science courses into educational programs. This integration has amplified the importance of computer science educators' motivation. The motivation of educators profoundly impacts the quality of computer science training. Teacher motivation is a vital factor impacting educational quality, as it influences lesson design, student engagement, and overall educational efficacy (Yavuz & Karadeniz, 2009). Teacher motivation is a determinant that directly impacts students' achievement in academic pursuits. Motivated educators execute their classes with more efficiency, while those lacking motivation view lessons mostly as a burden (Mabula, 2013). Teacher motivation is seen as an essential factor for success in education. Modern motivation research employs more sophisticated frameworks than merely distinguishing between intrinsic and extrinsic motivation; notably, Self-Determination Theory differentiates between autonomous motivation (such as intrinsic interest and alignment with professional values) and controlled motivation (such as external pressures), providing a more comprehensive foundation for scale development and interpretation (Ryan & Deci, 2000). Previous studies frequently differentiated between intrinsic motivation (internal fulfilment) and extrinsic motivation (external rewards). Contemporary frameworks, such as Self-Determination Theory (SDT), construct motivation on a continuum ranging from autonomous to controlled forms, offering a more complex understanding (Ryan & Deci, 2013). Nonetheless, in the rapidly evolving and continuously changing field of computer science education, the matter of teacher motivation has not been sufficiently investigated. The framework of computer science education requires enhanced technical proficiency and continual updates compared to other disciplines (Ni et al., 2023). Therefore, computer science educators must have a compelling motivation to engage in continuous professional development and to provide their students with the most current material (Ni et al., 2023; Yadav et al., 2017). The motivation of computer science educators is essential for enhancing educational quality and promoting student success in this field. An examination of national and international literature regarding teacher motivation in computer science education highlights the importance and shortcomings of the subject. Although there is no targeted research on teacher motivation in computer science education within the national literature, there exist extensive studies regarding teacher motivation in general. The research highlights various factors

affecting teacher motivation, such as the leadership styles of school administrators and the personal expectations of teachers (Coşkun, 2009; Duman, 2014; Elibol, 2013; Ertuğrul, 2021; İşgörür, 2020; Kulpcu, 2008).

Nonetheless, there exists a paucity of studies in the global literature regarding the motivation of computer science educators. The "Motivation to Teach Computer Science (MTCS)" scale, developed by Martin et al. (2023), is a significant and thorough evaluation of the motivations of computer science instructors. This measure evaluates instructors' motivations according to self-determination theory across four interconnected criteria. The study emphasized that teachers display a spectrum of motivation, ranging from external pressures to intrinsic drive. It has been established that efforts to enhance teacher motivation in computer science education must correspond with the instructors' requirements (Martin et al., 2023). These studies highlight the impact of teacher motivation on educational quality and emphasize the need to develop strategies to enhance teacher motivation in computer science education. The results, apparent in both national and international literature, highlight the necessity for comprehensive research on teacher motivation in computer science education in this study.

Presently, computer science education has been integrated into the curriculum at both the university level and from elementary school forward. Coding classes have been taught from an early age in the USA, Europe, and Far Eastern countries (Balanskat & Engelhardt, 2014). This situation constitutes substantial evidence of the swift global expansion of interest in computer science education. A primary justification for providing computer science education at a young age is the increasing demand for computer skills in the future workforce (Chen et al., 2017). In this context, it is essential for students to be introduced to computer science at a young age, allowing them to function as both consumers and creators in the technical domain (Grout & Houlden, 2014). Computer science education equips students with technical expertise while cultivating vital skills such as problem-solving, critical thinking, and creativity (Wing, 2006). While this study was executed inside the national framework of Türkiye, its contributions possess international significance. Motivation serves as a universal catalyst for teacher effectiveness, and the validated instrument created herein provides a framework that may be customized or evaluated in various nations.

This study is grounded in Self-Determination Theory (SDT), which conceptualizes motivation along a continuum from autonomous to controlled forms and emphasizes basic psychological needs (autonomy, competence, relatedness) as drivers of sustained professional engagement (Ryan & Deci, 2000). To account for organizational and management-related influences observed in our factor structure, we complement SDT with the Job Demands–Resources (JD-R) model, which highlights how job resources (e.g., administrative support, equipment, workload management) can foster motivation and buffer job demands (Bakker & Demerouti, 2007). Mapping our instrument and findings to these frameworks allows a more nuanced interpretation than a simple intrinsic–extrinsic dichotomy. The two-factor structure—encompassing both purpose-driven (SDT) and management-related (JD-R) elements—represents constructs that surpass national boundaries. This work offers a psychometrically robust instrument and empirical evidence from the burgeoning field of school computers, contributing to worldwide dialogues on teacher motivation and facilitating comparative research across many educational systems. This research aims to rectify a significant gap in the field by examining teacher motivation in computer science education. This study aims to address the following two principal research topics to accomplish its objective:

RQ1. Is the Perception Scale for Professional Motivation in Computer Science Education a valid and reliable instrument for measuring teachers' professional motivation?
RQ2. What are the underlying factors that shape the professional motivation of computer science teachers?
RQ3. To what extent do these motivational factors differ according to demographic and professional characteristics (e.g., gender, university of graduation, educational level, years of professional experience, and city of assignment)?

2. **Method**

This research utilized a quantitative methodology to examine teacher motivation in computer science education. The fundamental characteristic of quantitative research is that the data may be represented and analysed numerically (Karasar, 1994). The study aims to identify several factors affecting teacher motivation through quantitative analysis and to draw implications from the findings. The relational screening model was preferred in the study as one of the survey methodologies. This model's correlation type enables a more sophisticated analysis

of the relationship between variables. This study utilized the relational screening paradigm to evaluate the established scale and examine variations in teachers' professional motivation based on demographic factors like gender, educational background, years of experience, and location of assignment.

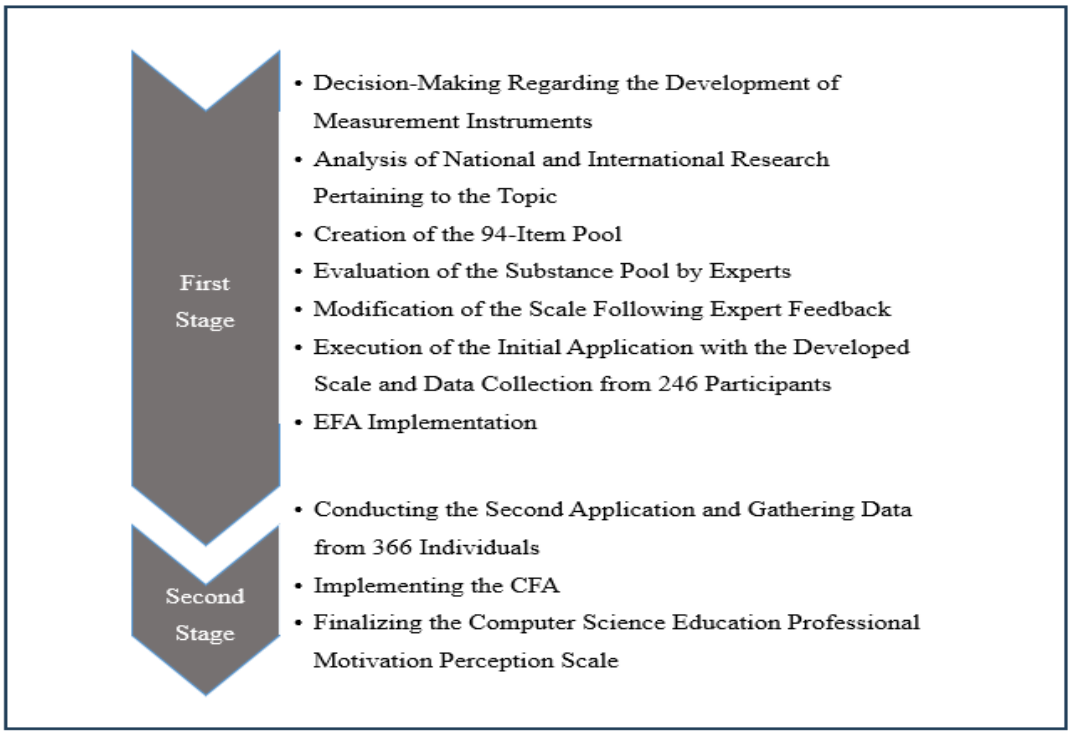


Figure 1. Operations Executed in Phases 1 and 2

This model enables the analysis of the relationship between teacher motivation and recognized effective elements, while predicting the prospective effects of these interactions on educational processes. Karasar (2003) contends that the relational screening model is an effective method for statistically evaluating the relationship between many variables. This study aims to clarify the factors affecting the motivations of computer science educators.

2.1 Participants

In study, data from extensive cohorts is employed to obtain critical information (Büyüköztürk, et. al., 2017). The study population consists of computer educators working throughout Turkey. This research utilized a sampling approach to obtain more accurate information about a specific demographic instead of including the entire population. A sample is defined as a subset that represents a certain part of the population, forming the foundation for the researcher’s inquiry (Büyüköztürk et al., 2017). This study utilized a convenience sampling method, a

variant of purposive sampling. This method entails choosing easily accessible individuals to facilitate and optimize the data collection process for the researcher (Yıldırım & Şimşek, 2003). As a result, readily accessible computer instructors in Turkey have formed the study group for the project. This technique has enabled the formation of a sample suitable for the research objectives, taking into account practical limitations such as study duration and accessibility. The scale was administered online through a survey link distributed to computer science educators in both public and private institutions throughout Türkiye. Nonetheless, the predominant portion of respondents originated from public schools, while private school educators were inadequately represented in the sample. The study utilized convenience sampling of easily accessible teachers; hence the findings cannot be confidently applied to all computer science educators in Türkiye. The sample may not accurately reflect the diversity of geographies, educational institutions, and available resources nationwide. Among the final scale participants, Sakarya exhibited the highest representation at 15.1% (f=28), whereas only one participant (0.5%) was sourced from various provinces including Adıyaman, Afyonkarahisar, Bingöl, Bitlis, Burdur, Edirne, Elazığ, Erzincan, Giresun, Karaman, Kırıkkale, Kırşehir, Konya, Mersin, Muğla, Muş, Rize, Tekirdağ, Trabzon, and Yalova. This disproportionate distribution further constrains the generalizability of the findings. Furthermore, despite the survey being disseminated to educators in both public and private institutions, the questionnaire lacked a question specifying the kind of institution, rendering it impossible to ascertain the precise representation of private school teachers in the sample. A scale development study was conducted during the project's initial phase. Subsequent to the scale's creation, the final application was implemented utilizing the acquired scale. Data were gathered from a total of 186 computer educators for the final application. Table 1 below displays the statistics regarding the gender variable of the individuals that participated in the final application.

Table 1. Data Regarding the Gender Variable of Participants in the Final Application

Gender	**Frequency (f)**	**Percentage (%)**
Male	115	61.8
Female	71	38.2
Total	186	100.0

2.2. Data Collection Tools

This study utilizes a Likert-type scale, designated as the "Perception of Professional Motivation in Computer Science Education Scale," developed by the researcher, as the instrument for data collection to achieve the research objective. Likert-type scales are tools commonly utilized in social sciences to evaluate individuals' attitudes regarding a specific subject. Tezbaşaran (1997) contends that Likert-type scales are often preferred for their capacity to measure equal intervals. The scale utilized in this study was deemed appropriate for this reason. The developed scale consists of two main dimensions: 'Factors Arising from Education - Teaching' and 'Factors Arising from Management.' These two dimensions aim to comprehensively evaluate instructors' views on professional motivation. A comprehensive content validation approach was undertaken to guarantee that the scale accurately represented the constructs delineated in our theoretical framework (SDT and JD-R). The scale's development entailed a multi-phase process: (1) an initial pool of 94 items was created via literature review; (2) items were refined following evaluation by a language expert; (3) three experts in computer science education assessed the items for clarity, representativeness, and content validity; (4) a pilot version comprising 31 items was administered; (5) Exploratory Factor Analysis (EFA) condensed the instrument to 26 items across two factors; and (6) Confirmatory Factor Analysis (CFA) further refined the scale to its final structure of 18 items. This systematic procedure offers substantial evidence for the construct validity and reliability of the scale. An initial item pool of 94 statements was developed based on this method. The pool underwent an initial evaluation by a linguistic expert for clarity, followed by requisite changes. The modified pool was subsequently appraised by three specialists in computer science education, who evaluated the items for relevance, comprehensiveness, and intelligibility. In response to their suggestions, the items were improved and condensed to 31, which were included in the pilot study. Experts evaluated each item based on clarity, relevance, and representativeness, and their written comment was integrated into the modifications. The expert review process demonstrated the content validity of the scale. The construction of items and the structure of the scale were guided by Self-Determination Theory (SDT) and the Job Demands-Resources (JD-R) model. Items included under the ‘Education/Teaching-Based’ factor predominantly represent educators’ perception of professional significance, acknowledgment from parents and the community, and support at the classroom level—elements consistent with Self-Determination Theory’s autonomous motivation and the needs for relatedness and competence. The components of the 'Management-

Based' factor encompass administrative support, equipment, and workload concerns, so aligning with JD-R job resources. Table 2 presents a comprehensive mapping of each item to its respective theoretical concept.

Table 2. Mapping of Scale Items to Theoretical Constructs

Item Code	Item (short description)	Factor	Theoretical Construct (SDT / JD-R)
m27	Lack of peer praise reduces my motivation	Management	JD-R: Social support as job resource
m30	Adequacy of software affects motivation	Management	JD-R: Material/technical resources
m21	Admin not sensitive to my work reduces motivation	Management	JD-R: Organizational support
m24	Gaining trust of school administration increases motivation	Management	JD-R: Organizational trust/support
m29	Adequacy of equipment affects motivation	Management	JD-R: Material resources
m28	Knowing I will retire in this profession affects motivation	Management	JD-R: Perceived Job security / long-term prospects
m22	Admin praise increases motivation	Management	JD-R: Recognition as job resource
m31	Supervisors' fair treatment affects motivation	Management	JD-R: Fairness / justice as job resource
m26	Peer praise increases motivation	Management	JD-R: Collegial support
m23	Admin not praising my work reduces motivation	Management	JD-R: Recognition deficit / lack of resources
m19	Transportation between home–school affects motivation	Management	JD-R: Physical/structural resources (workload strain)
m16	Living in an urban area affects motivation	Management	JD-R: Contextual resources/constraints
m25	Lack of admin trust reduces motivation	Management	JD-R: Organizational trust deficit
m20	Admin sensitivity affects motivation	Management	JD-R: Organizational support
m17	Distance home–school affects motivation	Management	JD-R: Physical strain / resource constraint
m5	Social media's negative attitude affects motivation	Education/Teaching	SDT: Controlled motivation (external pressures, social image)
m7	Parents' trust increases my motivation	Education/Teaching	SDT: Relatedness / autonomous motivation

m3	Praise from society increases my motivation	Education/Teaching	SDT: External recognition integrated as autonomous motivation

The items on the scale are assessed utilizing a five-point Likert-type rating system. Participants received five response options for each item: "Strongly Agree," "Agree," "Neutral," "Somewhat Agree," and "Disagree," so obtaining quantitative data on teachers' motivation levels. This grading method allows participants to express their ideas with increased flexibility and breadth (Tezbaşaran, 1997). This scale was created to evaluate the professional motivations of computer science educators and has become a crucial data source in achieving the study's primary goal by analysing various factors that affect instructors' motivations.

2.3 Data Analysis

The data obtained during the scale development process was subjected to exploratory and confirmatory factor analysis. The exploratory factor analysis was conducted using SPSS version 25.0 software. Subsequently, confirmatory factor analysis was conducted using AMOS 24.0 software. Following these methods, the data obtained from the developed scale were analysed using SPSS 25.0 software. Exploratory factor analysis (EFA) was performed utilizing Principal Axis Factoring on the Pearson correlation matrix with an oblique rotation (Direct Oblimin, $\delta = 0$), based on the anticipation of correlated factors. Items were maintained if their principal pattern loading was $\geq .40$ and the difference between the primary and secondary loading was $\geq .10$; items with communalities $< .30$ were deemed for removal. Factor retention was determined by eigenvalues exceeding 1 and the examination of the scree plot.

3. **Results**

The study methodology is structured into three phases: Phase 1 (item generation and content validation), Phase 2 (exploratory and confirmatory factor analyses for scale construction), and Phase 3 (final application of the validated scale). The findings are delineated into two primary sections: the initial phase (integrating Phases 1 and 2 for scale development) and the final phase (Phase 3 application)

3.1Results of Exploratory Factor Analysis (Initial Phase)

The research technique has three phases: Phase 1 (item generation and content validation), Phase 2 (exploratory and confirmatory factor analyses for scale development), and Phase 3 (final implementation of the approved scale). The results are categorized into two main sections: the starting phase (combining Phases 1 and 2 for scale development) and the concluding phase (Phase 3 application).

Table 2. Results of the Kaiser-Meyer-Olkin (KMO) Measure and Bartlett's Test for the EFA Data Set

KMO Coefficient		0,968
Bartlett Test	X^2	6251,976
	sd	325
	p	0,000

Tabachnick and Fidell (2013) assert that the KMO test value should be at least 0.6. The KMO test value for the dataset (KMO= 0.968), beyond the threshold, indicates a highly significant and normal distribution (Tavşancıl, 2018). After verifying that the KMO test values satisfied the necessary criteria, exploratory factor analysis (EFA) was conducted. Subsequent to the exploratory factor analysis, the factor loadings table indicated that specific scale components demonstrated cross-loading. Akgün et al. (2017) defines items that load on multiple factors as cross-loading items, stating that the difference in values between the factors must surpass 0.10. He emphasizes the importance of removing items that do not meet this criterion from the scale. Subsequent to the EFA, some alterations were executed on the "Computer Science Education Professional Motivation Perception Scale" to enhance its effectiveness. The overlapping entries m12, m14, m15, and m18 were subsequently removed from the scale. Items were maintained if their primary loading was ≥ .40 and cross-loadings on other factors were < .30, with a minimum difference of .10 between primary and secondary loadings to ensure discriminant validity (Tabachnick & Fidell, 2013). The tables proposed for examination in the factor analysis study were methodically reviewed, indicating that m1 demonstrated no substantial correlation with three items in the correlation matrix. As a result, the m1 item was omitted from the scale. Items were preserved according to established EFA criteria: (a) primary factor loading of at least .40, (b) cross-loading of less than .30 on any non-primary factor, and (c) conceptual alignment with the factor theme. Items that did not meet these criteria were eliminated progressively. According to these regulations, five items (m1, m12, m14, m15, m18) were discarded. Table 4 illustrates that items

distinctly loaded onto two connected variables. A Heywood case for item m27 (loading = 1.022) was observed but is allowable under oblique rotation (see to Table 4 comment). Item m11 was cross loaded but kept under Factor 2 due to its superior loading of .50. Factor 1 was designated as Management-Based Motivation, whereas Factor 2 was designated as Education/Teaching-Based Motivation, according to the conceptual consistency of the items. Following these processes, it was determined that the scale comprises 26 items and exhibits a two-factor structure. The table of total variance produced by these operations is displayed below. Factor retention was determined by the Kaiser criterion (eigenvalues > 1.0) and corroborated by visual analysis of the scree plot, both suggesting a two-factor solution.

Table 3. Aggregate Variance Table Subsequent to EFA

	Initial Core Values			**Sum of Squares of Loadings**		
Factor	**Total**	**Variance %**	**Cumulative %**	**Total**	**Variance %**	**Cumulative %**
1	16,421	63,158	63,158	16,421	63,158	63,158
2	1,181	4,541	67,700	1,181	4,541	67,700
3	0,930	3,576	71,275			

The cumulative total variance must be at least 60% for social sciences. Thus, the overall variance of the dataset following the EFA (67.700) surpasses this ratio. Subsequent to the exploratory factor analysis, it is important to examine the factor loading table.

Table 4. Pattern Matrix of Factor Loadings (Principal Axis Factoring, Direct Oblimin Rotation)

Item No	**Item**	**Factor 1**	**2**
m27	The fact that other teachers at school do not praise my work affects my teaching motivation.	1,022	
m30	The adequacy of the software I will use in the course affects my course motivation.	0,883	
m21	The fact that the school administration is not sensitive to my work affects my course motivation.	0,878	
m24	Gaining the trust of the school administration affects my motivation.	0,877	

m29	The adequacy of the equipment I will use in the lesson affects my motivation.	0,844	
m28	Knowing that I will retire in this profession affects my teaching motivation.	0,824	
m22	The fact that the school administration praises my work affects my course motivation.	0,778	
m31	Administrative supervisors' fair treatment of the staff affects my course motivation.	0,772	
m26	The fact that other teachers at school praise my work affects my teaching motivation.	0,768	
m23	The school administration not praising my work	0,735	
m19	The mode of transportation between school and home affects my motivation for studying.	0,710	
m16	Living in an urban area affects my motivation to study.	0,656	
m25	Not having earned the trust of the school administration affects my motivation to study.	0,656	
m20	The school administration's sensitivity to my work affects my motivation in class.	0,641	
m17	The distance between school and home affects my motivation to study.	0,576	
m10	Negative parent-teacher cooperation affects my motivation in class.	0,341	
m5	Social media's negative attitude and stance towards the teaching profession affects my motivation in class.		0,955
m7	Parents' trust in me affects my motivation to teach.		0,928
m3	People in society praising me for my profession affects my motivation to teach.		0,845
m6	In situations involving students, parental support affects my motivation to teach.		0,771
m4	Social media's positive attitude and stance towards the teaching profession affects my motivation in class.		0,758
m9	Good parent-teacher cooperation affects my motivation in class.		0,754

m8	Parents' belief that I perform my job well affects my motivation to teach.		0,679
m2	The negative perception of the teaching profession among people in society affects my motivation to study.		0,626
m13	The fact that the place where I live meets my personal needs affects my motivation to study.		0,560
m11	The small size of the place where I live affects my motivation to study.	0,330	0,501

Extraction is equivalent to Principal Axis Factoring. Rotation equals Direct Oblimin (δ = 0). The values reported are coefficients of the pattern. Standardized regression weights, as opposed to correlations, can yield values marginally exceeding 1.00 (e.g., m27 = 1.022) in oblique solutions (Tabachnick & Fidell, 2013). Item m11 exhibited cross-loadings of .33 on Factor 1 and .50 on Factor 2. The item was retained under Factor 2 due to its larger loading, which surpassed the .40 criterion. The Cronbach Alpha coefficient for the initial factor of the developed scale was 0.965, whilst the coefficient for the succeeding factor was 0.946.

Table 5. Cronbach's Alpha Values for the Factors Derived from Exploratory Factor Analysis

Factor	**Article Number**	**Cronbach's Alpha Value**
1	16	0.965
2	10	0.946

The results indicate that the scale possesses high dependability (Alpar, 2016). A study of the top group, which was 27% lower, was subsequently conducted on the dataset to evaluate the scale's discriminative capabilities.

Table 6. Reliability of Sub-Group and Super-Group Following Exploratory Factor Analysis

Factors	**N**	**Average**	**Average difference**	**p**
Top group	66	67,7576	40,37879	0,000
Subgroup	66	27,3788	40,37879	0,000

The t-test results revealed that the mean for the upper group was 67.7576, while the mean for the lower group was 27.3788. The difference between the means indicates that the developed scale is significantly discriminative ($p<0.05$). The initial factor is classified as Educational – Instructional Factors, whereas the subsequent element is referred to as Management-Related Factors, according to the items identified in the investigation. The two extracted factors exhibited a moderate correlation, suggesting that although they are conceptually separate, they share shared variance in line with theoretical assumptions.

3.2 Results for Confirmatory Factor Analysis (Initial Phase)

Following the adjustments, the Perception of Professional Motivation Scale in Computer Science Education was redistributed online and completed by 366 computer educators. Confirmatory Factor Analysis (CFA) is essential in the research of scale development (Akgün et al., 2017). Therefore, to assess the feasibility of doing a Confirmatory Factor Analysis (CFA) on the obtained dataset, the correlation between the individual items of the scale and the total scale score was examined. The analysis utilized Pearson correlation coefficients.

Table 7. Confirmatory Factor Analysis Table 7. Impact on Mean, Variance, and Item-Total Correlation Table Upon the Deletion of an Item from the CFA Data Set

Article No	Effect on the Average when the item is deleted	Effect on Variance When Item Deleted	Item-Total Score Correlation
m2	43,20	115,966	0,319
m3	42,98	114,268	0,402
m4	42,81	112,058	0,448
m5	42,86	112,310	0,454
m6	42,84	111,230	0,488
m7	42,63	111,117	0,426
m8	42,80	112,171	0,457
m9	42,81	112,418	0,439
m10	42,71	111,242	0,446
m11	42,67	109,903	0,517
m13	42,63	111,824	0,337

m16	42,61	113,406	0,284
m17	42,80	112,963	0,434
m19	42,75	110,185	0,512
m20	42,81	110,511	0,533
m21	42,79	111,702	0,440
m22	43,06	115,325	0,347
m23	42,92	113,445	0,405
m24	42,91	113,389	0,433
m25	42,80	112,105	0,440
m26	42,80	112,533	0,433
m27	42,76	111,201	0,481
m28	42,87	110,314	0,512
m29	42,73	111,476	0,417

The item-total score correlation values of the scale ranged from 0.284 to 0.533, and these correlations were statistically significant. Subsequently, the KMO (Kaiser-Meyer-Olkin) test was re-administered to assess the adequacy of the dataset for confirmatory factor analysis.

Table 8. Results of the Kaiser-Meyer-Olkin (KMO) and Bartlett's Test for the CFA Data Set

KMO Coefficient		0,886
Bartlett Test	X^2	2228,704
	sd	325
	p	0,000

The Bartlett test findings indicate a significant chi-square value (x^2=2228.704; sd=325; $p<0.05$), suggesting that the dataset demonstrates a multivariate normal distribution. The acceptable fit indices obtained from the confirmatory factor analysis significantly influence the scale's acceptability.

Table 9. Fit Index Benchmarks Adopted (Schreiber et al., 2006)

Index	Value
GFI	>0,90
AGFI	>0,90
NFI	>0,90
CFI	>0,90
RMSEA	<0,08

Furthermore, Çokluk et al. (2018) assert that an RMSEA value below 0.08 signifies acceptable fit, a CFI value above 0.90 denotes acceptable fit, and an NFI value surpassing 0.90 suggests good fit. Alongside these values, other critical factors must be considered during confirmatory factor analysis, particularly prior to item removal or model enhancement. The Standardized Regression Coefficient is paramount among these. The standardized regression coefficient indicates the capacity of observed variables to forecast latent variables, with a preference for these values to exceed 0.60 (Karagöz, 2021). Consequently, to enhance the scale, the Standardized Regression Coefficient is considered while eliminating items, and the item with the lowest value is discarded from the scale. Another element to contemplate in enhancing the Confirmatory Factor Analysis (CFA) model is the modification indices. The modification index signifies the anticipated decrease in the Chi-Square value when a parameter is altered or a new parameter is incorporated into the model (Sümer, 2000). Consequently, Confirmatory Factor Analysis (CFA) was utilized on the dataset. The fit index values derived from the CFA without any alterations are presented below. As shown in Table 10, the initial CFA model did not reach acceptable fit indices, indicating that the proposed two-factor structure required further refinement.

Table 10. Preliminary Fit Indices and Values for the Confirmatory Factor Analysis Model

Index	Value
GFI	,874
AGFI	,852
NFI	,720
CFI	,825
RMSEA	,056

As shown in Table 10, the initial CFA model did not reach acceptable fit indices. Consequently, some modifications were implemented to improve model fit. Based on low standardized loadings and modification indices, items m16, m13, m22, m2, m29, m10, m7, and m21 were sequentially removed from the scale, beginning with the lowest loading values. At each step, fit indices were re-assessed. In addition, three error covariances suggested by the modification indices were incorporated into the model. After these revisions, the fit indices reached acceptable levels, as reported in Table 11.

Table 11. Fit Indices and Values of the Final CFA Model

Index	Value
GFI	,945
AGFI	,929
NFI	,870
CFI	,954
RMSEA	,036

As shown in Table 11, the final CFA model demonstrated good fit. The RMSEA value (.036) indicates excellent fit, while the AGFI (.929), NFI (.870), and CFI (.954) reflect acceptable to good levels of fit (Schreiber et al., 2006). The GFI value (.945) is also considered satisfactory (Hooper et al., 2008). These results confirm that the revised two-factor model provided a valid representation of the data. Note. GFI = Goodness of Fit Index; AGFI = Adjusted Goodness of Fit Index; NFI = Normed Fit Index; CFI = Comparative Fit Index; RMSEA = Root Mean Square Error of Approximation.

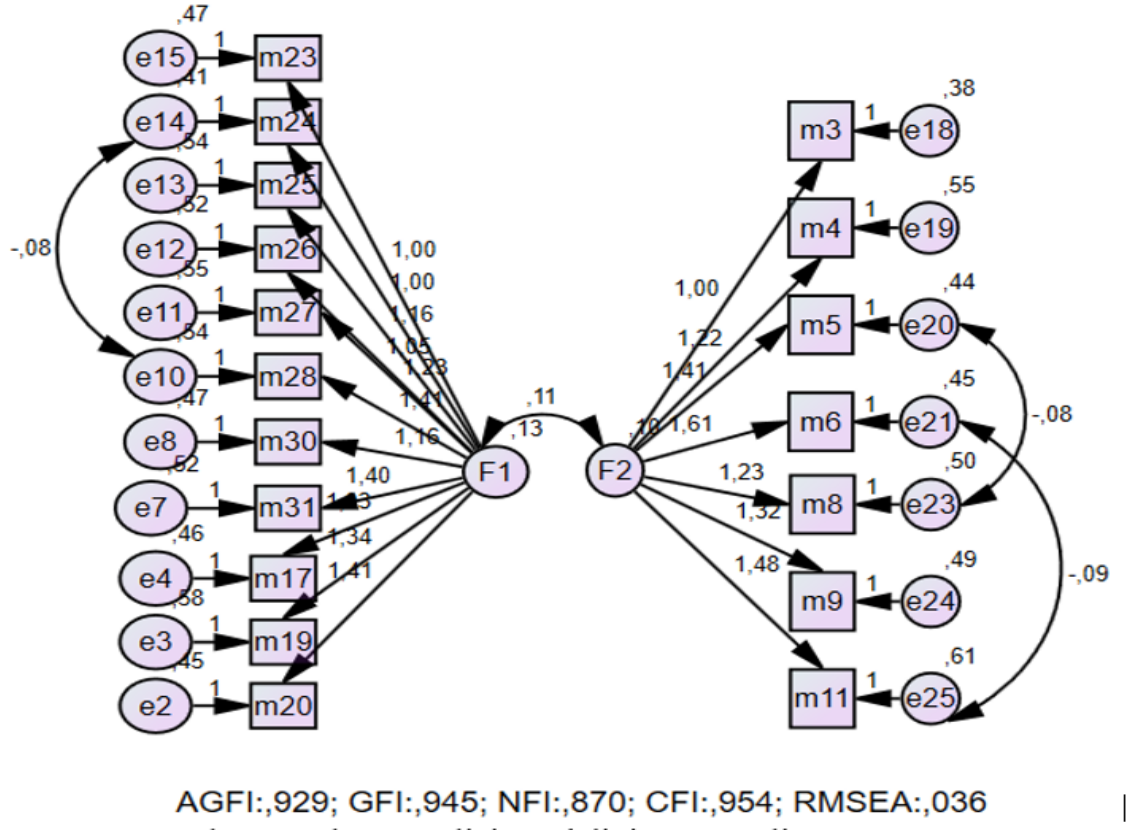


Figure 2. Conclusive Version of the Confirmatory Factor Analysis Model

Following the Confirmatory Factor Analysis (CFA), a 27% lower group - upper group analysis was performed on the dataset to assess the scale's discriminative power, and Cronbach's Alpha was utilized to evaluate its reliability.

Table 12. Reliability of Sub-Group and Upper-Group Post-CFA

Factors	N	Average	Average difference	p
Top group	99	41,2121	18,45455	0,000
Subgroup	99	22,7576	18,45455	0,000

The established Computer Science Education Professional Motivation Perception Scale significantly differentiates between the lower and upper groups ($p<0.05$) due to the enhancements implemented.

Table 13. Cronbach's Alpha Values for Factors Established Post-CFA

Factor	Article Number	Cronbach's Alpha Value
1	11	0.803
2	7	0,713

The Cronbach Alpha values indicate that the scale is at an appropriate level. Upon completion of all phases, the scale comprising 2 factors and 18 elements is prepared for final implementation.

3.3 Conclusive Findings of Application Results (Phase Two)

In the conclusive application of the Computer Science Education Professional Motivation Perception Scale, 186 computer science educators participated, with the distribution completed digitally. To choose the analytical methods for evaluating the final scale data, it is crucial to first examine the normal distribution of the dataset. Literature evaluations show that parametric tests are used for data with a normal distribution, while non-parametric tests are applied to data that diverges from normality. Subsequently, after analysing the descriptive statistics of the final scale, normality tests were conducted by computing the mean of the items that constitute the scale.

Table 14. Statistical Data for the Normality Test of the Final Scale

	Statistics	Standard Error
Average	1,7572	
Median	1,6667	
Variance	,229	
Standard Deviation	,47898	
Skewness Coefficient	,731	,178
Kurtosis Coefficient	,028	,355

According to Tabachnick & Fidell (2013), a scale demonstrates a normal distribution if the skewness and kurtosis values range from -1.5 to +1.5. Upon analysing the skewness and kurtosis values of the final scale, it was concluded that the scale exhibits a normal distribution. Consequently, parametric tests, including the t-Test and One-Way ANOVA, were performed on the final scale data. The histogram and Q-Q Plot of the final scale's data set, indicating a normal distribution, are presented below.

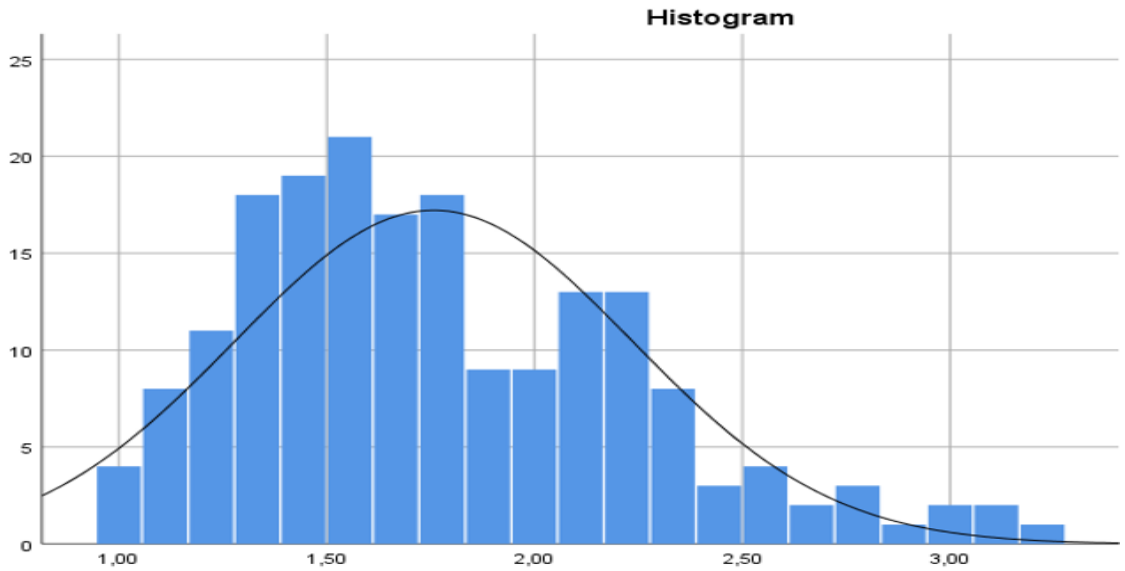


Figure 3. Histogram of the Normality Assessment for the Final Measurement

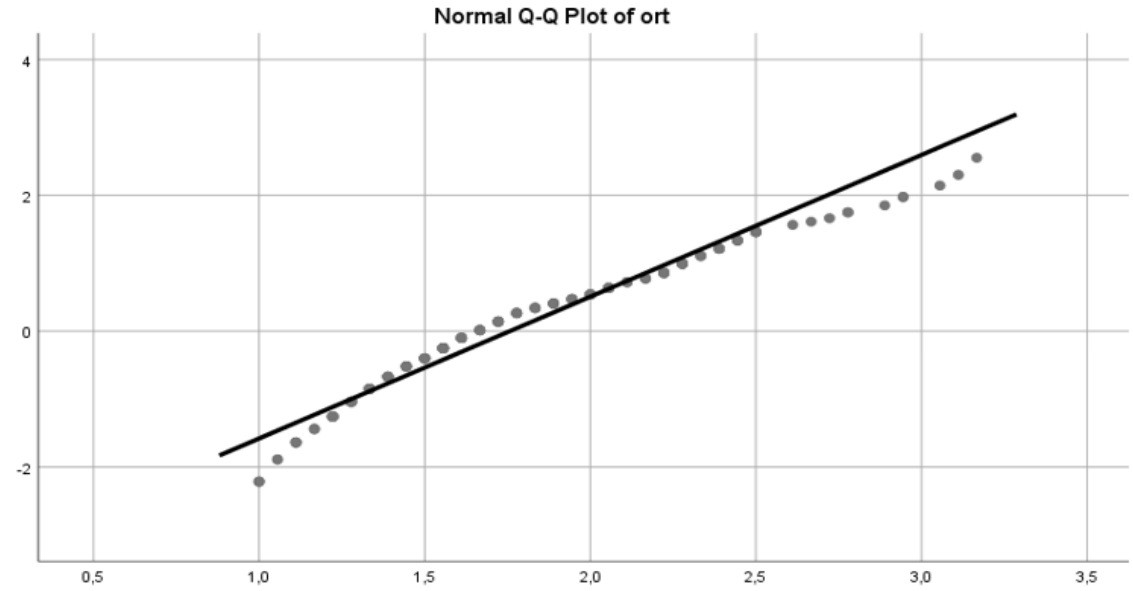


Figure 4. Q-Q Plot illustrating the normality assessment for the final measurement

Subsequent to these phases, the data acquired about the research's sub-problems were examined.

In the development process of the Computer Science Education Professional Motivation Perception Scale, reliability analyses were conducted, including EFA and CFA, followed by Cronbach's alpha coefficient and 27% lower and upper group comparisons. Based on the findings obtained from these analyses, it can be stated that the scale in question is a reliable measurement tool. The evaluation of construct validity, guided by EFA and CFA

findings, demonstrates that the Computer Science Education Professional Motivation Perception Scale is a legitimate tool. The t-test results for the Computer Science Education Professional Motivation Perception Scale scores, classified by gender, are displayed in the table below.

Table 15. T-Test Outcomes of the Perception of Professional Motivation Scale in Computer Science Education by Gender Variable

Factor	Groups	N	X	ss	t	sd	p
Factors related to Education and Training	Male	115	1,7715	,53436	,102	184	,919
	Female	71	1,7631	,56092			
Management Factors	Male	115	1,7143	,53251	,407	184	,684

The T-Test results indicate no significant difference between male and female computer science instructors for Education – Teaching Source variables (t[184]=0.102; p>0.05). Nonetheless, it has been established that there is no substantial difference between male and female computer science educators about Management-Related Factors (t[184]=,407;p>0,05). Consequently, it has been determined that the factors influencing the motivation of computer science educators remain consistent across genders for each sub-dimension. Prior to conducting the analysis of the Computer Science Education Professional Motivation Perception Scale scores based on the university attended, the assumption of homogeneity of variances was assessed using the Levene Test to ascertain the uniform distribution of the groups.

Table 16. Levene's Test Results for the University Graduated from Variable of the Perception of Professional Motivation Scale in Computer Science Education

Factor	Type of Statistics	Levene's Statistic	p
Factors related to Education and Training	Based on the average	1,409	,108
Management Factors	Based on the	1,604	,105

The findings of the Levene Test indicate that the factors are homogeneously distributed (p>0.05). A One-Way ANOVA test was conducted based on this result.

Table 17. Results of the One-Way ANOVA Test on the Perception of Professional Motivation Scale in Computer Science Education by University Graduates

Factor		Sum of Squares	sd	Squares Mean.	F	p
Factors related to Education and Training	Between Groups	9,170	35	,262	,865	, 684
	In Group	45,410	150	,303		
	Total	54,579	185			
Management Factors	Between Groups	8,593	35	,246	,877	, 667
	In Group	41,991	150	,280		
	Total	50,584	185			

The findings of the ANOVA test indicated no significant difference between Education – Teaching-Related factors and Management-Related factors concerning the university attended (p<0.05). Prior to analysing the scores of the Computer Science Education Professional Motivation Perception Scale based on educational levels through the One-Way ANOVA test, the assumption of homogeneity of variances was assessed using the Levene Test to ascertain the uniform distribution of the groups.

Table 18. Levene's Test Results for the Educational Level Variable in the Perception of Professional Motivation Scale for Computer Science Education

Factor	Type of Statistics	Levene's Statistic	p
Factors related to Education and Training	Based on the average	,244	,865
Management Factors	Based on the	1,423	,238

The Levene Test results indicate that the distributions of Education-Teaching Related Factors and Management Related Factors are homogeneous ($p > 0.05$). Following the confirmation of homogeneity of variances, the One-Way ANOVA test analysis was performed.

Table 19. Outcomes of the One-Way ANOVA Test for the Variable of Educational Level in the Perception of the Professional Motivation Scale in Computer Science Education

Factor		Sum of Squares	sd	Squares Mean.	F	p
Factors related to Education and Training	Between Groups	2,192	3	,731	2,539	,058
	In Group	52,387	182	,288		
	Total	54,579	185			
Management Factors	Between Groups	2,927	3	,976	3,725	,012
	In Group	47,658	182	,262		
	Total	50,584	185			

The analysis of the One-Way ANOVA test revealed no significant difference between the education levels examined and the Education-Teaching Source components ($p=0.058$; $p<0.05$). Furthermore, a statistically significant difference has been identified between the sub-dimension of Management-Related Factors and the educational level examined ($p=0.012$; $p<0.05$). An LSD test was conducted to identify the subgroups exhibiting significant differences. The exam results indicate a substantial difference ($p<0.05$) in Management-Related Factors between middle school and vocational high school levels, although no such difference exists among the other levels. Prior to conducting the analysis of the Computer Science Education Professional Motivation Perception Scale scores based on years of professional experience through a One-Way ANOVA test, the assumption of homogeneity of variances was assessed by examining the groups for a uniform distribution via the Levene Test.

Table 20. Levene's Test Results for the Variable of Years of Professional Experience in the Perception of Professional Motivation Scale for Computer Science Education

Factor	Type of Statistics	Levene's Statistic	p
Factors related to Education and Training	Based on the average	1,309	,273
Management Factors	Based on the	,581	,628

Upon analysing the outcomes of the Levene Test, it was concluded that the variances of Education-Teaching Related Factors and Management Related Factors were homogeneously distributed (p>0.05). Upon establishing the homogeneity of variances, the One-Way ANOVA test analysis was performed.

Table 21. Outcomes of the One-Way ANOVA Test for the Variable of Years of Professional Experience in Relation to the Perception of the Professional Motivation Scale for Computer Science Education

Factor		Sum of Squares	sd	Squares Mean.	F	p
Factors related to Education and Training	Between Groups	2,103	4	,526	1,813	,128
	In Group	52,476	181	,290		
	Total	54,579	185			
Management Factors	Between Groups	2,369	4	,592	2,223	,068
	In Group	48,216	181	,266		
	Total	50,584	185			

The research revealed no significant difference between Education-Training-Related variables and years of professional experience (p=0.128; p<0.05). It has been established that there is no substantial difference between Management-Related variables and years of professional experience (p=0.068; p<0.05). Prior to conducting the study of the Computer Science Education Professional Motivation Perception Scale scores by city through a One-Way ANOVA test, the assumption of homogeneity of variance was assessed using the Levene Test to ascertain whether the groups exhibit a homogeneous distribution.

Table 22. Levene's Test Results for the City Variable in the Perception of Professional Motivation Scale within Computer Science Education

Factor	Type of Statistics	Levene's Statistic	p
Factors related to Education and Training	Based on the average	1,219	,221
Management Factors	Based on the	1,233	,210

The findings of the Levene Test indicate that the Education-Training Related Factors and Management Related Factors have a homogenous distribution ($p>0.05$). Following the confirmation of homogeneity of variances, the One-Way ANOVA test analysis was performed.

Table 23. Results of One-Way ANOVA for the City of Employment Variable in the Perception of Professional Motivation Scale for Computer Science Education

Factor		Sum of Squares	sd	Squares Mean.	F	p
Factors related to Education and Training	Between Groups	17,902	50	,358	1,318	,108
	In Group	36,677	135	,272		
	Total	54,579	185			
Management Factors	Between Groups	15,622	50	,312	1,206	,199
	In Group	34,962	135	,259		
	Total	50,584	185			

The ANOVA test findings indicated no significant difference between Education-Training Related variables and the city of duty performance ($p=0.108$; $p<0.05$). No substantial difference was seen between the city of duty and Management-Related Factors ($p=0.199$; $p<0.05$).

4. Discussion And Conclusion

The Computer Science Education Professional Motivation Perception Scale aims to discover the factors affecting the motivation of computer science educators. The scale's content validity was first evaluated, subsequently leading to the development of a critical item pool through an extensive review of national and international literature. A total of 94 items were assessed by a language expert and subsequently submitted to three field specialists. Following the integration of expert feedback, revisions were made to the questionnaire, resulting in a 31-item tool named "Computer Science Education Professional Motivation Perception Scale." Exploratory Factor Analysis (EFA) and Confirmatory Factor Analysis (CFA) were employed to evaluate the construct validity of the instrument. Exploratory Factor Analysis (EFA) was conducted on data from 246 participants, resulting in the elimination of items m1, m12, m14, m15, and m18 from the scale, so yielding a two-factor structure of 26 items. These elements are classified as Education – Teaching-Oriented and Management-Oriented. To assess the scale's reliability, Cronbach's Alpha values and 27% upper-lower group comparisons were conducted, indicating that the scale exhibits robust dependability. Confirmatory Factor Analysis (CFA) was utilized to assess the structure obtained from Exploratory Factor Analysis (EFA) based on data gathered from 366 individuals. Subsequent to the CFA, some items (m2, m7, m10, m13, m16, m21, m22, m29) were discarded until the fit indices reached an acceptable threshold, culminating in a modified scale comprising 18 items. The model fit was assessed using many indices: GFI, AGFI, NFI, CFI, and RMSEA. The final 18-item, two-factor model demonstrated a satisfactory fit (GFI= 0.945, AGFI= 0.929, NFI= 0.870, CFI= 0.954, RMSEA= 0.036), conforming to established thresholds (Hu and Bentler, 1999). These indices validate that the two-factor model sufficiently encapsulates the data. The dependability was re-evaluated by Cronbach's Alpha values, determining that the scale is adequately reliable. Cronbach’s Alpha values were .96 for the Management-Based factor and .94 for the Education/Teaching-Based component. The total scale produced an Alpha of .97, above the .70 benchmark suggested for internal consistency (Kalyar, Ahmad, & Kalyar, 2018). The composite reliability values surpassed .70, hence reinforcing reliability. This scale, consisting of two components and eighteen elements, is considered a reliable, valid, and useful measurement tool. The two-factor structure is consistent with the theoretical frameworks underpinning this investigation. Pragmatic elucidation of the two-factor model. The Education–Teaching-Based aspect underscores the significance of autonomy-supportive classroom practices, such as providing meaningful choices and

prioritizing mastery feedback, while also enhancing connections with families and the community to reinforce teachers' autonomous motivation. The Management-Based element underscores employment resources identified by the JD-R model—transparent and equitable administrative procedures, prompt acknowledgment, and sufficient equipment/software—which can mitigate demands and maintain motivation. Explicitly framing interventions using Self-Determination Theory (needs for autonomy, competence, relatedness) and the Job Demands-Resources model (organizational resources) offers a theoretically informed framework for schools aiming to augment the professional motivation of computer science teachers. The Education–Teaching-Based factor encapsulates teachers' intrinsic motivation within Self-Determination Theory (SDT), encompassing aspects such as professional significance, acknowledgment from parents and the community, and classroom-level resources that fulfil the demands for relatedness and competence. The Management-Based factor relates to job resources in the JD-R model, including organizational trust, administrative assistance, and the sufficiency of equipment and infrastructure. This theoretical congruence offers additional evidence for the construct validity of the scale and contextualizes the findings within the wider field of motivation research.

Although motivation studies for educators in other disciplines exist at the national level, the lack of study focused on the motivation of computer science teachers is due to the absence of a measurement scale for evaluating their motivation. This situation underscores the imperative of creating a framework in the field. The development of the Computer Science Education Professional Motivation Perception Scale would significantly enrich the current literature. This scale will serve as the first national tool to evaluate the professional motivations of computer science educators, providing valuable data for scholars and educational institutions. The application of the scale may augment the breadth of quantitative and qualitative study into the factors influencing computer science teachers' motivation. Research examining the relationships between teachers' personality variations, work conditions, and motivations in educational settings can yield new insights on educational sciences. Globally, there is a dearth of research concerning the motivations of computer science educators, and the limited studies utilize scales that exhibit recognized validity and reliability. The recent international study presented the 18-item Motivation to Teach Computer Science (MTCS) scale, encompassing four dimensions: external pressures, external advantages, student benefits, and personal enjoyment (Martin, Baker, Haynes, &

Warner, 2023). Positioning with relation to MTCS. While MTCS categorizes motives into four teaching-centric domains (external pressures/benefits, student benefits, personal enjoyment), our two-factor framework encompasses a wider ecological perspective of professional motivation, incorporating organizational factors (management, resources, recognition) in addition to teaching-related influences. This differentiation is beneficial for governance and leadership: MTCS may provide greater diagnostic insights for pedagogical support, whereas our scale also highlights systemic levers—such as administrative trust, equity, and infrastructure—that school leaders may influence. While MTCS emphasizes instructors' motivation to instruct in computer science, our instrument concentrates on the professional incentive drivers within the Turkish educational setting, resulting in two factors: Education–Teaching-Based and Management-Based. Consequently, our research enhances MTCS by offering a verified, nationally normed instrument and empirical results pertinent to Turkey, encompassing subgroup analyses applicable to local administrative frameworks. However, the development of the Computer Science Education Professional Motivation Perception Scale will provide a foundation for international comparative study. An examination of the motivation levels of computer science instructors in different countries and the factors affecting this motivation could guide the formulation of global educational strategies. Moreover, examining the influence of cultural differences on motivation should assist in developing both universal and specific measures for enhancing teacher motivation. The study concluded with an examination of data from 186 computer science educators who completed the Computer Science Education Professional Motivation Perception Scale. The results demonstrated that the factors affecting the motivations of computer science professors were uniform across genders. This study indicates that gender does not substantially affect teachers' motivations. This outcome suggests that the motivating factors in our framework—autonomous motivation (SDT) and job resources (JD-R)—are seen similarly by male and female instructors. This indicates that motivational resources and requirements, as defined by SDT and JD-R, are independent of institutional background. Therefore, efforts to enhance motivation should use a gender-neutral and inclusive approach. International literature indicates some studies identifying gender disparities (Duursma, 2016), while others claim no substantial differences (Kippers et al., 2018). These findings indicate that educational institutions should cultivate cultures that enhance teacher motivation free from gender bias. No differences in motivating factors have been seen based on the university attended. This scenario indicates that the university does not affect the

motivation of computer science instructors. This discovery raises questions about the impact of graduation on motivation and suggests that the quality of education is predominantly consistent among universities. Selvitopu & Taş (2020) found that motivation levels significantly varied according to undergraduate degree status. The motivational factors were consistent regardless of teachers' educational qualifications, focusing instead on managerial elements. No notable disparities were detected according to the city of assignment. A notable disparity was observed in the Management factor based on the educational level of the teachers (primary, secondary, or high school). This study utilized convenience sampling, potentially constraining the sample's representativeness and, therefore, the generalizability of the scale features and Phase 3 subgroup comparisons. The imbalance in group sizes for some demographics diminished the statistical power to identify minor effects and heightened the likelihood of Type I and Type II errors. All measurements were self-reported and cross-sectional, hence precluding causal inference. Subsequent research ought to replicate these findings utilizing probability samples (e.g., stratified sampling across areas and school types), gather longitudinal data to assess stability over time, and evaluate measurement invariance across significant subgroups. Educational caveat. The survey link was disseminated to both public and private schools; however, the questionnaire lacked a question to identify the type of school, preventing us from assessing any potential variations between the two types of institutions. Future applications must specifically document school type and utilize a stratified sample by sector to facilitate comparisons and improve external validity. Considerations for measurement. Prior to comparing subgroup means in subsequent investigations, multi-group confirmatory factor analysis (CFA) should be employed to ascertain configural, metric, and scalar invariance of the two-factor model across significant groups (e.g., gender, educational attainment). Furthermore, to alleviate common-method variance associated with single-source, self-report methodologies, subsequent research could integrate survey responses with observational or administrative metrics or implement temporal separation of measures. The study utilized convenience sampling; hence, the results from demographic subgroup analyses (e.g., gender, years in the profession, school type [public vs. private]) should be evaluated cautiously, since they may not be representative of the wider community of computer science educators. Nonetheless, a weakness of this study is the employment of convenience sampling, which constrains the generalizability of the results. By analysing the components via Self-Determination Theory (SDT) and the Job Demands-Resources (JD-R) model, our research broadens the

relevance of these frameworks to computer science education, emphasizing the dual significance of personal meaning and institutional resources. Future research should utilize more representative sample techniques to improve external validity and offer a comprehensive understanding of the motivation of computer science educators.

Consequences for implementation and regulation. Based on our findings, school leaders and policymakers should prioritize (a) transparent and equitable administrative processes that acknowledge the contributions of computer science teachers, (b) dependable provision and upkeep of computer science-specific equipment and software, (c) organized parent-school engagement to enhance community recognition, and (d) professional development aligned with autonomy-supportive pedagogy. Resource allocation models at the system level must align with the unique management requirements of various educational stages, especially in vocational high schools where management-related motivation is notably elevated, thereby ensuring that organizational support is customized to the specific needs of stage-specific computer science curricula and infrastructure.

REFERENCES

Akgün, E., Karadeniz, Ş., Demirel, F., Kılıç, E., & Büyüköztürk, Ş. (2017). *Bilimsel araştırma yöntemleri*. Pegem Akademi Yayıncılık.

Alpar, C. R. (2016). *Spor Sağlık ve Eğitim Bilimlerinden Örneklerle Uygulamalı İstatistik ve Geçerlik Güvenirlik*. Detay Yayıncılık.

Bakker, A. B., & Demerouti, E. (2007). The Job Demands-Resources model: state of the art. *Journal of Managerial Psychology*, *22*(3), 309–328. https://doi.org/10.1108/02683940710733115

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

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, *109*, 162–175. https://doi.org/10.1016/j.compedu.2017.03.001

Çokluk, Ö., Şekercioğlu, G., & Büyüköztürk, Ş. (2018). *Sosyal Bilimler İçin Çok Değişkenli İstatistik SPSS ve Lisrel Uygulamaları*. Pegem Akademi.

Coşkun, M. (2009). *İlköğretim Okullarında Motivasyon Araçları Hakkında Öğretmen Görüşleri Ve Doyum Düzeyleri Üzerine Bir Alan Araştırması* [Yüksek Lisans Tezi, Beykent Üniversitesi]. https://tez.yok.gov.tr

Duman, N. (2014). *Okul Yöneticilerinin Kullandıkları Motivasyon Faktörleri Ve Öğretmen Görüşleri* [Yüksek Lisans Tezi, Yeditepe Üniversitesi]. https://tez.yok.gov.tr/

Duursma, E. (2016). Who does the reading, who the talking? Low-income fathers and mothers in the US interacting with their young children around a picture book. *First Language*, *36*(5), 465–484. https://doi.org/10.1177/0142723716648849

Elibol, S. (2013). *Ortaöğretim Okullarında Motivasyon Araçları Hakkında Öğretmen Görüşleri Ve Doyum Düzeyleri Üzerine Bir Alan Araştırması* [Yüksek Lisansn Tezi, Yeditepe Üniversitesi]. https://tez.yok.gov.tr/

Ertuğrul, S. (2021). *Öğretmen Algılarına Göre Okul Müdürlerinin Toksik Liderlik Davranışları İle Öğretmenlerin Motivasyon Ve İş Tatmin Düzeyleri Arasındaki İlişki* [Yüksek Lisans Tezi, İstanbul Sabahattin Zaim Üniversitesi]. https://tez.yok.gov.tr/

Grout, V., & Houlden, N. (2014). Taking Computer Science and Programming into Schools: The Glyndŵr/BCS Turing Project. *Procedia - Social and Behavioral Sciences*, *141*, 680–685. https://doi.org/10.1016/j.sbspro.2014.05.119

Hooper, D., Coughlan, J., & Mullen, M. (2008). Evaluating model fit: a synthesis of the structural equation modelling literature. *In 7th European Conference on Research Methodology for Business and Management Studies*, 195–200.

Hu, L., & 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. https://doi.org/10.1080/10705519909540118

İşgörür, N. (2020). *Okul Yöneticilerinin Öğretmen Motivasyonunu Artırmaya Yönelik Yararlandığı Motivasyon Araçları* [Yüksek Lisans Tezi, Bahçeşehir Üniversitesi]. https://tez.yok.gov.tr/

Kalyar, M., Ahmad, B., & Kalyar, H. (2018). Does Teacher Motivation Lead to Student Motivation? The Mediating Role of Teaching Behavior. *Voprosy Obrazovaniya / Educational Studies Moscow*, *3*, 91–119. https://doi.org/10.17323/1814-9545-2018-3-91-119

Karagöz, Y. (2021). *Bilimsel Araştırma Yöntemleri*. Nobel Akademik Yayıncılık.

Karasar, N. (1994). *Bilimsel Araştırma Yöntemi*. Nobel Akademik Yayıncılık.

Karasar, N. (2003). *Bilimsel Araştırma Yöntemi*. Nobel Akademik Yayıncılık.

Kippers, W. B., Wolterinck, C. H. D., Schildkamp, K., Poortman, C. L., & Visscher, A. J. (2018). Teachers' views on the use of assessment for learning and data-based decision making in classroom practice. *Teaching and Teacher Education*, *75*, 199–213. https://doi.org/10.1016/j.tate.2018.06.015

Kulpcu, O. (2008). *İlköğretim Okullarında Görev Yapan Öğretmen Ve Yöneticileri Motive Etmede Kullanılabilecek Motivasyon Araçları Üzerine Bir İnceleme* [Yüksek Lisans Tezi, Gaziantep Üniversitesi]. https://tez.yok.gov.tr/

Mabula, J. S. (2013). Impact of motivation on commitment of teachers for public secondary schools in dar es salaam: a case of kinondoni district. *Master's Thesis, In Human Resource Management (Mhrm) of The Open Unıversity of Tanzania.*

Martin, N. D., Baker, S. N., Haynes, M., & Warner, J. R. (2023). The Motivation to Teach Computer Science (MTCS) scale: development, validation, and implications for use. *Computer Science Education*, 1–18. https://doi.org/10.1080/08993408.2023.2182561

Ni, L., Bausch, G., & Benjamin, R. (2023). Computer science teacher professional development and professional learning communities: a review of the research literature. *Computer Science Education*, *33*(1), 29–60. https://doi.org/10.1080/08993408.2021.1993666

Popovich, P. M., Gullekson, N., Morris, S., & Morse, B. (2008). Comparing attitudes towards computer usage by undergraduates from 1986 to 2005. *Computers in Human Behavior*, *24*(3), 986–992. https://doi.org/10.1016/j.chb.2007.03.002

Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, *55*(1), 68–78. https://doi.org/10.1037/0003-066X.55.1.68

Ryan, R. M., & Deci, E. L. (2013). *Intrinsic motivation and self-determination in human behavior*. Springer Science & Business Media.

Schreiber, J. B., Nora, A., Stage, F. K., Barlow, E. A., & King, J. (2006). Reporting Structural Equation Modeling and Confirmatory Factor Analysis Results: A Review. *The Journal of Educational Research*, *99*(6), 323–338. https://doi.org/10.3200/JOER.99.6.323-338

Selvitopu, A., & Taş, A. (2020). Lise Öğretmenlerinin İş Doyumu ve Mesleki Motivasyon Düzeylerinin İncelenmesi. *Bayburt Eğitim Fakültesi Dergisi*, *15*(29), 23–42. https://doi.org/10.35675/befdergi.426989

Sümer, N. (2000). *Yapisal Eşitlik Modelleri: Temel Kavramlar ve Örnek Uygulamalar*. Türk Psikoloji Yazılari.

Tabachnick, B. G., & Fidell, L. S. (2013). *Using multivariate statistics (6th edn)*. Pearson Education.

Tavşancıl, E. (2018). *Tutumların ölçülmesi ve spss ile veri analizi (6. Baskı)*. Nobel Yayın Dağıtım.

Tezbaşaran, A. (1997). Validity issues of a likert type scale (a case study). *Hacettepe Üniversitesi Eğitim Fakültesi Dergisi*, *13*, 41–45.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, *49*(3), 33–35.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, *60*(4), 55–62. https://doi.org/10.1145/2994591

Yavuz, C., & Karadeniz, C. B. (2009). Sınıf öğretmenlerinin motivasyonunun iş tatmini üzerine etkisi. *Uluslararası Sosyal Araştırmalar Dergisi*, *2*(9), 507–519.

Yıldırım, A., & Şimşek, H. (2003). *Sosyal Bilimlerde Nitel Araştırma Yöntemleri*. Seçkin Yayıncılık.

International Journal of Computer Science Education in Schools, February 2026, Vol. 7, No. 3
ISSN 2513-8359

Rubric for the qualitative assessment of student-designed Snap! Projects

Nicole Marmé[1], Jens-Peter Knemeyer[1], Alexandra Švedkijs[1]

[1]University of Education Heidelberg

DOI: 10.21585/ijcses.v7i3.226

Abstract

An objective evaluation and assessment of individual student-designed projects are challenging. Appropriate tools are currently lagging and have to be developed. Block-based programming languages, such as Snap! are often used for teaching programming basics and the subsequent development of student-designed programming projects. The current research qualitatively developed a rating rubric for Snap! projects to investigate how novices' programming skills can be evaluated and assessed in a criterion-guided manner. For this purpose, an evaluation was conducted on a baseline dataset of 36 student projects created over three school years after a programming course for novices. Based on this database we designed an assessment rubric. A team of experts reviewed and evaluated the assessment rubric. Following expert evaluation, the rubric was improved and expanded. Finally, prospective teachers conducted a comparative evaluation of a test data set consisting of ten Snap! projects of varying complexity, with and without the resulting rubric. The results show that the rating rubric significantly improves the comparability of assessments. In addition, a clear differentiation of the projects by level is achieved for the test data set. Furthermore, the assessment rubric enables a more precise achieved result evaluation in particular rubric category.

Keywords: Computer science, rubric, qualitative assessment, learning outcomes, teaching materials, coding, programming languages, Snap!

1. Introduction

Global challenges and technological progress have brought about a heightened emphasis on information technology skills over the last two decades. The demand for e-learning offers is constantly growing, especially for IT skills (mmb Institut, 2021). Digitization at all levels and global crises, such as the Covid-19 pandemic, are intensifying discussions across Europe about which skills and abilities will be needed in the future to be able to participate in social life (European Commission. Directorate General for Communication., 2020). Regarding digital competences in particular, competence requirements and necessary action steps for the next decades are being formulated nationally and internationally at various political levels (European Commission. Directorate General for Education, Youth, Sport and Culture., 2023). The Council of the European Union highlights digital literacy as one of the eight key competences for lifelong learning in the 21st century (Publications Office of the European Union, 2019). The current framework on European Union digital citizenship competence DigComp 2.2 lists programming competence as one of the key competences (European Commission. Joint Research Centre., 2022). The current approach to facing the challenges in Germany, for example, is to expand computer science lessons across all grades from fifth grade onwards. For the required strengthening of programming skills, the current educational plan for computer science recommends block-based programming for the acquisition of basic knowledge and skills in programming, especially for beginners (Ministerium für Kultus, Jugend und Sport Baden-Württemberg, 2016a, 2016b). The use of block-based programming languages often goes hand in hand with the development of individual creative projects (Krugel & Ruf, 2020; Resnick, Silverman, et al., 2009; Resnick, 2014). To successfully implement block-based programming languages in the classroom, a systematic approach is needed to evaluate such creative student projects. There are already some approaches to evaluating block-based programmes as will be discussed in section 2.2. However, most approaches deal with automated evaluation of the generated code. This involves solving pre-designed test tasks and evaluating them automatically. Such systems do not allow for the evaluation of individual projects on open topics. This paper therefore investigates whether a competency grid can be used to evaluate open-ended Snap! projects and how such a grid must be structured to ensure valid and consistent evaluation. To address this question, a multi-phase research design was applied, including the development of the rubric, expert validation, and empirical testing

with student projects. The results demonstrate that the rubric improves the comparability of evaluations and provides a practical, criterion-based tool for assessing creative, block-based programming projects.

2. Background

2.1 Block-based programming for novices

Block-based programming languages are visual programming languages that use blocks to represent code, rather than traditional text-based code. This allows users to create programs by dragging and dropping these blocks together, without having to write lines of code. A program code is put together like a puzzle by assembling the already available instruction blocks. These environments operationalize Papert's constructionist principles by providing concrete, manipulable elements that support self-directed creation, experimentation, and reflection (Papert, 1993). Learners actively construct knowledge, explore multiple solution paths, and iteratively refine their projects, fostering discovery-based learning and reducing the abstraction barriers typical of traditional coding (Brennan & Resnick, 2012; Resnick et al., 2009). Platforms such as Snap! enable students to design interactive projects-games, stories, or animations - promoting cognitive engagement, problem-solving, and creativity. The visual, block-based interface simplifies syntax, while project sharing, remixing, and collaborative exploration enhance social learning and knowledge co-construction, key aspects of constructivist and constructionist pedagogy (Papavlasopoulou et al., 2019). Compared to common programming languages that use textual syntax, block-based languages allow easier interaction with the programming environment and learners can focus more on programming logic instead of dealing with syntactical errors (Balouktsis, 2016). Block-based languages provide a low barrier to entry and a flexible, expressive environment. This allows learners to focus on creative and meaningful projects, fostering computational thinking, systematic reasoning and digital literacy (Resnick, Maloney, et al., 2009).

Block-based programming languages are characterised by their ability to eliminate syntax errors, reduce cognitive load and shift the focus from memory recall to visual recognition through structured, visual program construction. They are particularly valuable in lowering the entry barrier for novices and enabling intuitive, interactive learning that fosters engagement and a deeper understanding of core programming concepts (Bau et

al., 2017). In particular, beginners are able to concentrate more on understanding programming concepts rather than memorising text syntax due to the reduction in cognitive load (Weintrop & Wilensky, 2018)
Block-based programming languages, such as Snap! show significant advantages for introducing programming to novices. These languages are considered "easier" than text-based programming languages (Weintrop & Wilensky, 2015) and enable an introduction to programming for learners without any prior knowledge (Maloney et al., 2010). For example, the use of block-based programming languages can provide a better understanding of basic programming concepts, like loops (Mladenović et al., 2020). In addition, block-based programming languages offer a more visual interface that can make programming concepts more accessible. Features such as execution visibility, language extensibility and liveness in block-based languages create a positive attitude towards learning and using them (Perera et al., 2021). The use of block-based languages also increases student motivation in introductory programming courses by promoting positive emotions about performance, which in turn improves learning performance and engagement (Tsai, 2019; Wen et al., 2023). With block-based programming languages, learners grasp the task more quickly and achieve significantly more learning goals in the same amount of time compared to those using text-based languages (Price & Barnes, 2015). Interest in further programming activities is also rated higher after a learning sequence with a block-based programming language (Weintrop & Wilensky, 2017). The integration of block-based programming activities significantly improves pupils' computational thinking skills and their self-efficacy in problem solving. Such activities actively engage learners, promote their independence and strengthen their confidence in applying programming concepts (Koray & Bilgin, 2023).

Snap! is a further development of the Scratch programming environment, already established in many schools. Snap! offers some advantages and additional functions compared to Scratch; for example, Snap! enables comprehensive prototype-based programming by creating objects (Modrow, 2018). In addition, new blocks can be created as subroutines with control structures, also called the Build Your Own Block principle. The programming toolbox for object-oriented programming is comprehensive, so that Snap!, in contrast to Scratch, is a "fully developed programming language" (Modrow, 2018) and is thus in principle also suitable for advanced computer science teaching. This is also reflected in the fact that Snap! is now sometimes offered as an

introductory programming language for first semesters of computer science (Garcia et al., 2012). In summary, learning programming using block-based programming languages such as Snap! offers an accessible and visual approach to learning basic concepts, enabling students to develop essential programming skills while fostering their creativity, problem-solving abilities, and logical reasoning. Block-based programming languages are moreover based on the vision of enabling programming beginners to implement learning-by-doing or learning-by-making, where they are free to experiment with their own ideas, such as creating, sharing, playing, and learning with computers (Harel et al., 1993). Therefore, to promote programming skills for beginners in a school context, the use of block-based programming languages can be beneficial, especially for creation of student-designed projects.

2.2 Assessment of block-based programmed student projects

When working in the context of student-designed projects, it is crucial to establish suitable evaluation concepts that offer clear and transparent assessment measures for both teachers and students. By doing so, educators can review the quality of learning materials and provide valuable feedback to support student learning and growth. Assessment of student performance and feedback is an essential part of the learning process (Hattie, 2009). Nevertheless, the assessment process is one of the most complex activities in a teacher's job (Jürgens & Lissmann, 2015). Effective feedback should focus on the task and process, provide clear guidance on how to improve, link specifically to goals and performance (Shute, 2008). Additionally, research suggests that feedback should be specific and focused on the most important aspects of student work (Wiliam, 2011). The challenge of assessing student-designed projects lies in their open-ended nature, as they are characterized by diverse approaches, ideas, and implementations, making direct comparisons difficult.

To address this challenge, various concepts and tools for assessing block-based programming projects have been proposed. In most assessment concepts, however, there is a lack of consensus regarding the concrete establishment and weighting of assessment criteria (Da Cruz Alves et al., 2019). This is probably because there is currently no standardized competence framework derived from an empirically validated model (Gesellschaft für Informatik, 2016). Moreover, most existing systems were not designed for the evaluation of authentic, open-ended projects, but rather for standardized, task-based learning contexts**.** Most authors concerned with

assessment, either through the development of tools or the investigation of evaluation processes, regard their approaches as supplementary to teaching and as a means of supporting learning (Boe et al., 2013; Denner et al., 2012; Funke & Geldreich, 2017; Koh et al., 2014; Moreno-León et al., 2017; Seiter & Foreman, 2013; Werner et al., 2012; Zhang & Biswas, 2019).

Table 1 provides an overview of prominent approaches and tools for assessing block-based programming projects, highlighting their aims, methods, strengths, and limitations.

Study (Author, Year)	Aim / Context	Assessment Method	Strengths	Limitations
Boe et al., 2013 – Hairball	Evaluate Scratch projects to identify problematic or missing constructs	Static analysis with customizable plugins (e.g., initialization, synchronization, loops)	Objective, scalable detection of code patterns; high accuracy (≈99%)	Limited to predefined patterns; cannot assess creativity or design; manual review still needed
Denner et al., 2012	Middle school girls' game projects (Stagecast Creator)	Research study analysing 108 games using coding categories (complexity, usability, documentation)	Authentic insight into conceptual understanding; large dataset	Not a standardized tool; rule-based, not block-based; limited transferability
Koh et al., 2014 – REACT	Middle school STEM / Scalable Game Design classes	Real-time formative assessment of computational thinking patterns	Timely feedback for teachers; identifies misconceptions during coding	Limited to predefined CT patterns; misses qualitative and creative aspects
Werner et al., 2012 – Fairy Performance Assessment	Game programming elective using Alice	Performance-based tasks measuring CT (abstraction, modelling, problem-solving)	Authentic, multi-dimensional CT measurement; supports collaboration studies	Specific to Alice; high implementation effort; limited creativity assessment
Ball & Garcia,	University Snap!	Automated grading	Scalable grading;	Limited to closed-

Study (Author, Year)	Aim / Context	Assessment Method	Strengths	Limitations
2016 – Autograder λ	courses	and feedback integrated into Snap!	immediate feedback; simple setup	ended tasks; no assessment of creativity or design
Wang et al., 2021 – SnapCheck	Snap! courses with interactive projects	Automated testing using predefined templates and simulated user actions	High accuracy (≈98%); scalable; integrated into Snap!	Only for testable behaviours; setup time-intensive; cannot assess open-ended creativity
Moreno-León et al., 2017 – Dr. Scratch	Scratch programming contest projects	Automated static analysis compared to human expert ratings	Strong correlation with experts; consistent and scalable	Ignores creativity and design; focused on technical aspects only

As the table (Table 1) shows, automated tools such as Hairball (Boe et al., 2013), Dr. Scratch (Moreno-León et al., 2017), or SnapCheck (Wang et al., 2021) offer highly scalable solutions and produce consistent results but are primarily limited to predefined technical patterns and cannot capture creativity or the quality of open-ended designs. In addition, some systems face technical barriers such as installation issues and a constant need for updates to remain functional, which affects their acceptance among teachers (e.g., Ball & Garcia, 2016). Even when functioning well, these systems often provide only structural feedback about the code and lack the ability to evaluate whether a problem was solved in a meaningful way (Moreno-León et al., 2017; Wang et al., 2021). These limitations explain why most authors explicitly recommend using automated systems as a complement to traditional, teacher-driven assessments rather than as a replacement.

For example, Hairball and Dr. Scratch are powerful tools for detecting certain constructs, but they do not assess design aspects, while SnapCheck provides highly accurate testing of interactive behaviours yet requires significant preparation of templates and is unsuitable for authentic, free-topic projects.
Thus, there is still a clear need for research and development to create evaluation instruments that can provide rich, individualized feedback for authentic student work.

One effective approach to evaluating student-designed projects is using rubrics. Andrade H. defines a rubric as a one– or two–page document that describes varying levels of quality, from excellent to poor, for a specific assignment (Andrade, 2000). Rubrics provide a clear and consistent framework for evaluating authentic student-designed projects. By making expectations explicit and providing qualitative, criterion-based feedback, rubrics help students understand how to improve their work and promote deeper learning (Wolf & Stevens, 2007). The rubric presented in this study was developed specifically for Snap! projects and aims to qualitatively capture and objectively assess the outcomes of open-ended, autonomous student projects. It was developed as part of the evaluation of an interdisciplinary self-learning course, "Smart City" (Svedkijs et al., 2022) for learning the basics of programming with Snap! to be able to qualitatively record and objectively assess the student projects created.

3. Method

3.1 Development procedure

We opted for a qualitative and exploratory approach to developing the assessment rubric because the research question is open and the aim is to generate a practical, field-tested assessment instrument (Döring & Bortz, 2016; Gummels, 2020).

3.2 Teaching sequence

To this purpose, 183 students (the majority with no prior knowledge) in grades 9-11 were taught the basics of programming with the block-based language Snap! in an approx. 20-hour teaching sequence in the school years 2018/19, 2019/20 and 2020/21. No one had any previous knowledge of block-based programming. Following the lesson sequence, the pupils created their own projects in small groups on a free topic. Forty pupil projects resulted from this and after data cleansing, 36 projects were available. The rubric was developed using anonymized student project data, collected with informed consent and without any personal identifiers, complemented by published projects from the Snap! platform.

3.3 Analysis and Drafting

Available projects could be used as a baseline data set for the development of the rubric (Fig.1). The development of the rubric involved a comprehensive process, starting with the analysis of baseline data from student projects and expert evaluation.

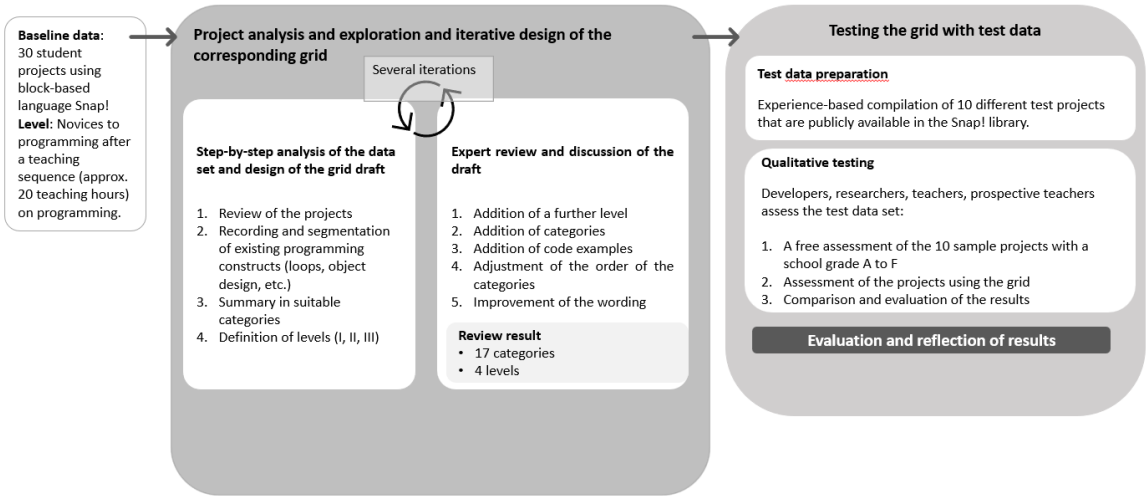


Figure 1:Development process of the assessment rubric

To begin, a thorough analysis of the given dataset was conducted, examining each project's structure and content to gain a deep understanding of its programming constructs, such as loops and object designs. This allowed for systematization and categorization of used programming constructs. The resulting summaries enabled definition of three different levels (I, II, III) within the dataset. Based on these findings, a draft of the rubric was created with twelve thematic categories and three levels.

3.4 Expert Review and Iterative Exchange

We reviewed the initial rubric version together with four educational experts (2 female, 2 male) in the field of programming for qualitative assessment. We defined experts as individuals with several years of experience using block-based languages, particularly Snap!, in teaching contexts or those who had published academically on block-based programming languages. Experts' review led to refinement through an iterative exchange process. The final version featured seventeen categories and four levels. In addition, according to the expert advice the rubric was supplemented with source code examples, and categories were edited and put in a different order. Beyond that, a general "project characteristics" category with a keyword-like description of the project characteristics in the respective level was added as an orientation framework. Furthermore, a "creativity impression" category was introduced. Here, a subjective estimate of project creativity in the sense of technical originality and inventiveness is to be given.

4. Evaluation Process and Testing

To test the developed rubric, we prepared a dataset by selecting ten publicly available Snap! projects that reflect typical student work after their first exposure to programming. These projects varied in complexity, subject matter, and interactivity, ensuring a representative range of examples. This dataset illustrates the possible range of projects and serves as a reference for evaluation.

Finally, nine prospective teachers (male: 4, female: 5) majoring in computer science, technology, mathematics or natural sciences participated in the evaluation process. They had prior knowledge of Snap! or other block-based programming languages and possessed existing teaching experience. Initially, they rated the randomly sorted projects without any predetermined criteria using a school mark scale (1 = very good, A; 6 = insufficient, F). Afterward, they received the developed rubric and evaluated the same projects again based on the specified criteria. The evaluation of the competence grid was performed in German language.

4.1 Current version of the rubric

The current version of the rubric[i] comprises seventeen categories and four levels (0, 1, 2, and 3). For each category, a level can be awarded in one of the four levels. The overall level is determined as the sum of all points awarded within all categories.

The respective categories cover aspects of object-oriented development (e.g. objects or object communication), algorithmic design (use of loops, branches, reporters), handling of data (variables, lists), Snap! specific design options (graphic effects), handling of multithreading (header blocks and multithreading), and code outsourcing (BYOB). In addition, the "project characteristics" category describes a general implementation in relation to the corresponding level. The "creativity impression" category attempts to capture a subjective impression of the project that cannot be measured by the other categories. All categories and levels are listed below in descriptive statements translated into the English language.

1. Category “objects”

Level 0: Create an unstructured instruction sequence in a sprite.

Level 1: Create instruction sequences in an existing sprite to implement a specific function, e.g. object draws, object moves.

Level 2: Create and name another object(s) using a parallel statement sequence.

Level 3: Independently create several other objects with a communication or interaction for modelling a complex system.

2. Category “stage as an object”

Level 0: Cannot recognize stage as an object. No stage backgrounds/functions.

Level 1: Embed the stage in the system: set one or more backgrounds for the stage.

Level 2: Perceive the stage as an object: create a program for designing the stage, for example, by automatically changing the background images, using the graphic effects, time lapses.

Level 3: Perceive the stage as an object: create a program for the stage with object interaction.

3. Category “communication with a user AND/OR with other objects”

Level 0: Cannot use communication instructions.

Level 1: Use condition block to evaluate keyboard or mouse input or colour coding.

Level 2: Create simple communication between objects or with the environment.

Level 3: Create advanced communication between objects/with the user, for example via variables.

4. Category “Use of reporter blocks or predicates”

Level 0: Cannot demonstrate implementation of the reporter and predictor blocks.

Level 1: Use simple reporter blocks, such as random number or x-position.

Level 2: Use reporter/predicator blocks as parameter AND/OR in conditions.

Level 3: Use complex/composite reporter/predicator blocks.

5. Category “Graphical effects, sound effects, draw effects”

Level 0: Cannot demonstrate implementation of effects, etc.

Level 1: Use simple sound/speech/drawing instructions/graphical effects.

Level 2: Control the graphic effects AND/OR use combinations of different properties and sounds.

Level 3: Use graphical effects (effect combinations) meaningfully, for example to visualize a complex plot or to design the program interface.

6. Category “Hat blocks and multithreading”

Level 0: Always start instruction sequence without a hat block.

Level 1: Use a hat block to start the script, the script runs linearly.

Level 2: Create several scripts within a project, but without targeted use of the multithreading concept: scripts work independently of each other.

Level 3: Use several different hat blocks for a multithreading processing of the programs AND/OR use a hat block for sending the messages AND/OR "When I start as a clone”.

7. Category “Object actions”

Level 0: Present a loose collection of instructions, no meaningful structure of a program.

Level 1: Create a sequence of instructions with fixed numerical values, e.g. with concrete size specifications AND/OR create a sequence of instructions for a sprite movement or figure geometry with waypoints.

Level 2: Use control flows with fixed values.

Level 3: Parameterize the statement sequence AND/OR use variables in control flows.

8. Category “Creating variables”

Level 0: Treat data as fixed values, with no variables present.

Level 1: Create and name a variable.

Level 2: Create several variables.

Level 3: Create (a) variable(s) for data exchange between objects (global variables) or within an object (local variables). Demonstrate meaningful use of local and global variables.

9. Category “Using variables”

Level 0: Use only numbers or words as constants.

Level 1: Change variables as numbers or strings in the course of the program.

Level 2: Change the value of a variable depending on a condition, for example, set false to true.

Level 3: Use variables as data containers for various data such as lists, objects.

10. Category “Using operators”

Level 0: Cannot show use of operators.

Level 1: Use simple mathematical operations, such as plus, minus, etc. in the function as a reporter.

Level 2: Use nested operators with variables AND/OR simple operators within a one-way branch/loop.

Level 3: Demonstrate meaningful use of complex operators, e.g. in conditions.

11. Category “Use of predicates in control flows”

Level 0: Cannot show existing termination condition (except for endless loop) AND/OR incorrect termination condition.

Level 1: Formulate a non-parameterized termination condition for a control flow.

Level 2: Formulate a parameterized termination condition for a control flow.

Level 3: Use operators (e.g. and, or, not) for a termination condition in a condition/loop AND/OR complex conditions (referring to other objects).

12. Category “Use of conditions”

Level 0: Cannot demonstrate implementation of conditions.

Level 1: Use an if-condition or an if-else condition.

Level 2: Use a nested branch AND/OR use a one-way branch for multiple cases.

Level 3: Show sensible use of complex nesting (but no unnecessary nesting, clear source code).

13. Category “Use of loops”

Level 0: Cannot demonstrate loops implementation.

Level 1: Use a loop.

Level 2: Use a combination of two loops (e.g. nesting them).

Level 3: Use multiple loops and complex loop structures, e.g. For loop.

14. Category “Use of lists”

Level 0: Cannot demonstrate list implementation.

Level 1: Create a simple list AND/OR output the list AND/OR prompt input for a list.

Level 2: Use list elements according to the respective index.

Level 3: Create lists with objects AND/OR further lists AND/OR use complex structures and commands.

15. Category “Build Your Own Block”

Level 0: Cannot demonstrate own block implementation.

Level 1: Combine several commands in their own blocks (outsource code).

Level 2: Create a block with a return value or with (a) parameter(s). Create reporter.

Level 3: Create a block with complex parameters AND/OR return values, such as lists and objects.

16. Category “Project characteristics”

(Selected examples; full description in the online version)

Level 0: A simple project with partly correct approaches but overall is inadequate or contains errors. No concept/no idea available. Loose collection of objects and functions.

Level 1: A project is manageable. 1-2 stage backgrounds are used. The plot is implemented with 2 to 3 objects. Simple control flows, instructions, operators are used.

Level 2: The project has a comprehensive structure. Several stage sets with effects are used. Control structures, instructions, links are used. Code is outsourced.

Level 3: The project has a complex structure. The plot is complex, exciting. Complex control structures, instructions, links, lists are used. Custom blocks are used with parameters and return values.

17. Category “Creativity impression”

Level 0: "The task has not been solved."

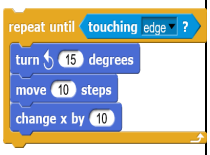
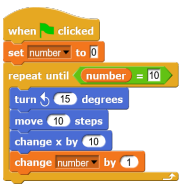
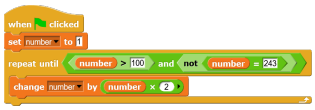
Level 1: "The task is solved, but not very creatively".

Level 2: "I understand the concept, it's exciting!"

Level 3: "Wow, that's a cool idea; a successful concept!"

Each level is described with a statement and, if useful, supplemented with a source code example (see example Tab. 1 "Using reporters and predicates"):

Table 1: Excerpt of a category from the rubric with four descriptions at each level and corresponding source code examples.

Category	Level 0	Level 1	Level 2	Level 3
Using predicates in control flows	Cannot show existing termination condition (except for endless loop) AND/OR use incorrect termination condition	Formulate a non-parameterized termination condition for a control flow	Formulate a non-parameterized termination condition for a control flow	Use operators (e.g. and, or, not) for a termination condition in a condition/loop AND/OR complex conditions (referring to other objects).
Code example				

4.2 Description of the test data set

The dataset used for the testing of the rubric consists of ten exemplary projects taken from the virtual Snap! library. All of these projects can be found in the Snap! collection" at: The link has been hidden for the review process for anonymity reasons.

The following criteria were formulated for the dataset:

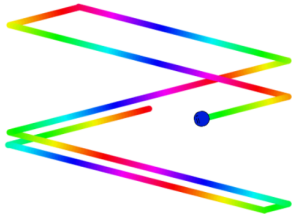
- The projects should, as far as possible, have different levels of complexity in source code, presentation, and plot.
- The projects can be interpreted as an average student performance after a teaching sequence in programming for novices.

The following projects were selected. For reasons of clarity, the projects are presented in ascending order of complexity - Note: the test persons, however, received the projects in random order.

Project 1. Row row - Movement of an object along predefined waypoints.

The source code exhibits a linear, redundant structure. The object moves from one coordinate point to another, and the route is not automated. The outputs are implemented using a concurrently running script. The functionality and presentation are rudimentary. Overall, it is a simple project with some recognizable multi-threading usage.

Project 2: Rainbow Ball – Movement of an object along a random route with colour change.



The source code includes a loop and instructions from various categories such as movement, appearance, etc. The action is limited to visualization on the stage, and the representation is animated. Overall, it is an interesting project with an idea that was not further developed or implemented in a context.

Project 3: Exclusive Complexity – Calculating the average of 10 number inputs.

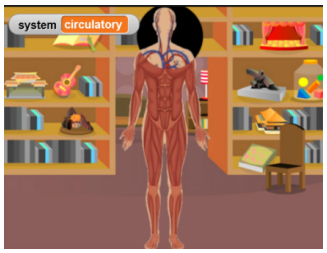
The source code has a linear structure, not parameterized, and lacks code modularization.

The program flow is linear, with a single thread of execution. The presentation includes a background image and an object. Overall, it is a simple project involving mathematical calculations.

Project 4: Avocado gif – An animated postcard featuring an avocado mother plant and its seed.

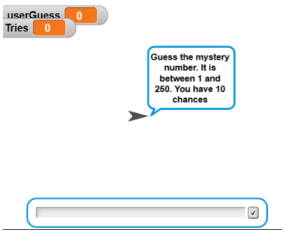
The source code implements multiple objects. Various control structures are used, and the code is modularized. The action runs concurrently. The action involves visualization without user interaction, and the representation is animated. Overall, it is a small but visually appealing project with a concept. The narrative flow could be further developed.

Project 5: Human body scanner – With a lens, various systems of the body can be observed.



The source code is concise. Instructions from different categories are utilized. There is no code modularization or user communication. Overall, it is a small, visually appealing project with potential for further enhancements.

Project 6: Guessing Game – The user is required to guess a number within a specific range.



The source code includes control structures and instructions for user communication. Code is modularized. There are no stage animations, only one object. Overall, it is a simple project related to a classic task.

Project 7: eCard Challenge – A game and an animated Halloween postcard combined into one. The user is required to answer quiz questions.

The source code incorporates various types of instructions. Object interaction and communication are present. The action and presentation are cohesive. However, the source code lacks modularization, resulting in redundant code segments. Overall, it is a good project with room for improvement.

Project 8: Fashion game – The user can dress and style a model.

An extensive project utilizing instructions from various categories, with code modularization. It has a complex structure, and the action and visualization complement each other. Overall, it is a comprehensive project with a clear concept.

Project 9: Dogder – A reaction game where the square object needs to avoid black dots.

The source code utilizes a comprehensive range of instructions and control structures. It involves complex interactions, a well-defined narrative flow, and efficient visualization. However, the code lacks modularization, resulting in some code redundancy and reduced readability. Overall, it is a complex project that showcases a wide range of functionalities. Project 10: Shooter Arcade Game – A shooting game with different levels of complexity.

An extensive project utilizing instructions from various categories, with efficient visualization. It has a complex structure, and the action and visualization complement each other. Overall, it is a comprehensive project with a clear concept, but the source code may be somewhat challenging to navigate due to its complexity

5. Results

Each project was first assessed with a school grade using German grading system from 1(A = very good) to 6 (F = insufficient). In the second part of the evaluation the test dataset was assessed with the described rubric.

The rubric consists of **17 categories**, each of which can be rated on four performance levels (0–3 points). This results in a **maximum attainable score of 51 points** (*17 categories × 3 points*). To ensure comparability between the two evaluation phases, the total rubric score was converted into the German grading system using a linear transformation formula:

$$Grade = 6 - \frac{5 * RS}{MS}$$

RS: reached score; MS: maximum score

The following graph (Fig.2) illustrates the comparisons of the mean values of the ten assessments. The blue cross represents the average ratings of the projects using grades without the rubric, while the orange cross represents the mean rubric scores converted using the formula mentioned above.

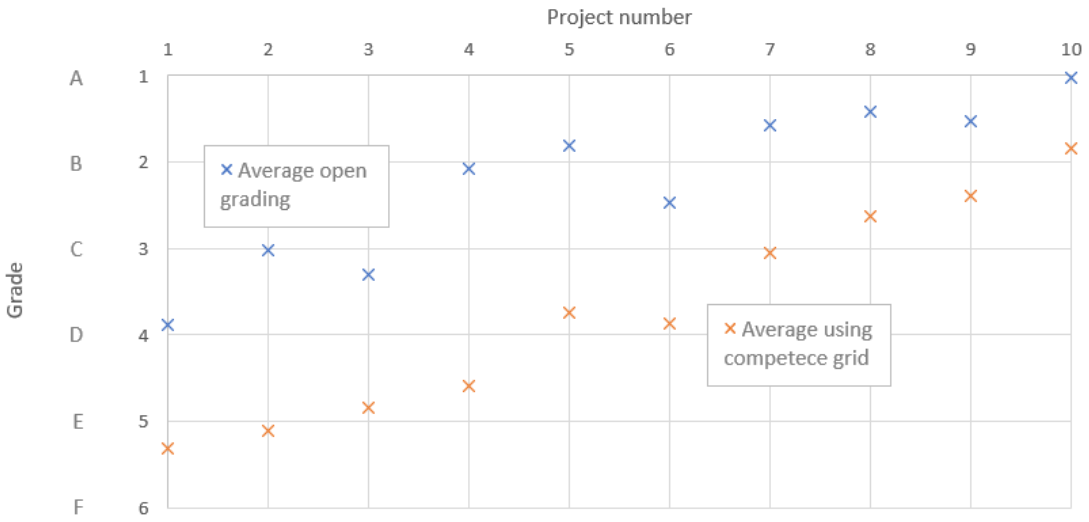


Figure 2: The average grades for test dataset. Blue crosses mark average results for grading without a rubric. Orange crosses mark the average results for grading using the rubric. For better comparison the score was converted in German school grades from 1(A) to 6(F)

It is noticeable that the projects assessed with the rubric receive significantly lower ratings. Even the best project, on average, achieves only a good grade (B) compared to grading without the rubric. Four projects do not meet the minimum requirements (grade D). In the open evaluation without a rubric, most projects achieve good to excellent grades. Both grading systems show in general the tendency from weak projects to good projects. Nevertheless, there are some outliers in the evaluation without a rubric, for example regarding projects 2 and 3 or projects 5 and 6. The difference in evaluation is indeed significant, with certain projects showing an average difference of two grades (e.g., Project 4).

The following diagram (Fig. 3) shows the individual results of the assessment without a rubric.

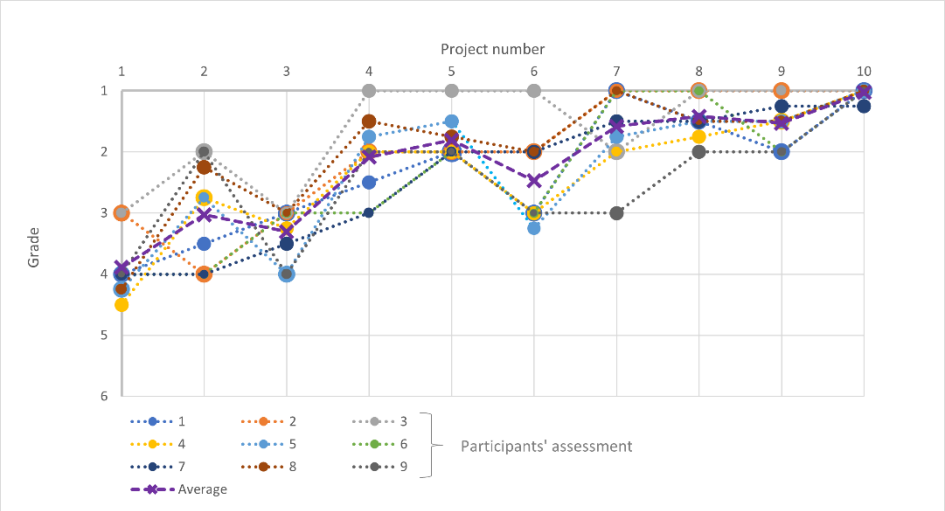


Figure 3: Dispersion of the results after assessing with a school grade with no predetermined criteria. Average is marked with a cross.

In the first part of subjective grade assessment of the projects, the spread of the grades for the respective projects is particularly striking. Project 2, for example, is assessed by the experts in a range of grades between 2 (good) and 4 (sufficient). The dispersion is strikingly high for all projects. Only project 10 is rated as a very good by all evaluators. Furthermore, it is striking that most projects tend to be assessed in the upper third of the rating range. The assessment of projects shows wide variation within each individual evaluation. For example, one participant rates project 2 as well as projects 8 and 9 with a good grade, although in the direct comparison it remains questionable whether these three projects achieve the same level. At the same time, this participant rates projects 6 and 7 as significantly worse, with a satisfactory grade. Another participant also evaluates project 2 as good, but again evaluates projects 6,7,8 and 9 with an almost very good grade. In this evaluation model, it is thus not obvious according to which criteria the evaluations are made and a comparison of the evaluations among each other becomes almost impossible. Thus, this evaluation method does not appear to be transparent and cannot be used for the evaluation of the student projects.

When using the rubric, a higher consistency of the distribution of points can be observed (Fig. 4) and the results of the assessment show usually much lower dispersion.

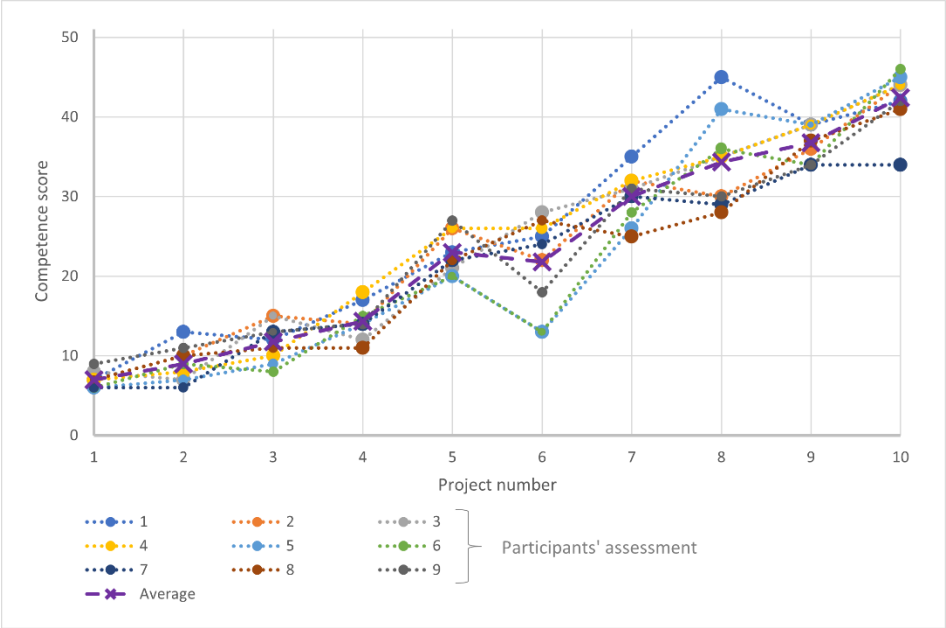


Figure 4: Dispersion of the results after assessing with a rubric. Maximum score result is 51 points for the best grade. Round dots mark the evaluation of German experts. Average is marked with a cross.

Clear outliers can be observed in projects 6 and 8. In project 6, code outsourcing (Category 15: Build Your Own Block) in own blocks was sometimes overlooked during the assessment, resulting in an incorrect assessment. Project 8 was partially classified as having a too high level of mastery. Presumably, differences between the second and third levels of the rubric can be recognized less easily by inexperienced raters. For example, for this project, the third level is awarded in the categories on loops and branches, even though the source code has a level of only two. Despite the observed inconsistencies, the most projects can be assessed more homogeneously in each case. All ratings are mostly within one grade. This means that projects can be better assigned to the different levels. Thus, the projects are evenly distributed among the lower, middle, and high score ranges. The following graph (Fig. 5) shows the average deviation in grade points from the mean assessment grades for the respective projects with and without rubric. The overall mean deviation is 0.41 without the rubric. Without the ceiling effect, the deviation would probably be significantly higher, especially for good projects (8-10 comparable with projects 1-7). The overall mean deviation is 0.24 with the rubric, demonstrating a substantial decrease in rating variability and improved assessment consistency.

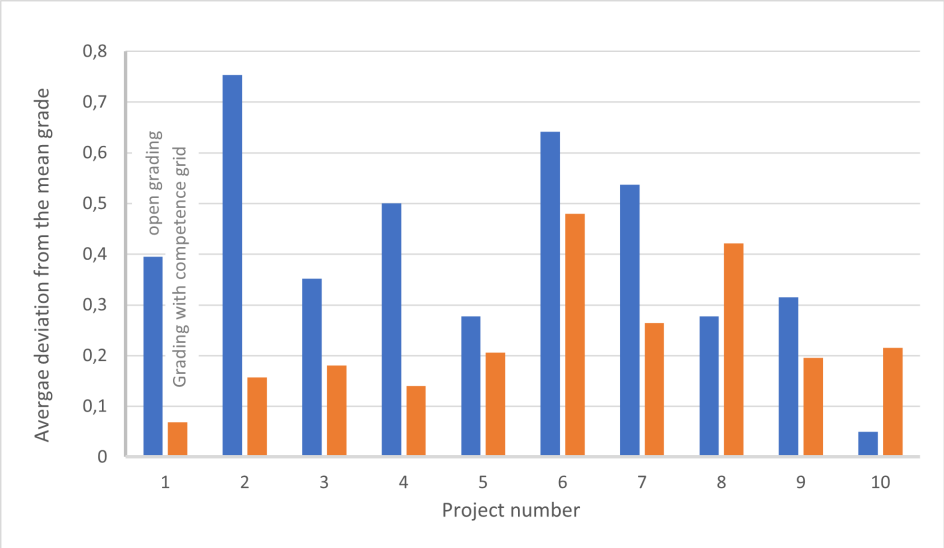


Figure 5: Average deviation from mean grade, blue colour for grading without rubric and orange colour for grading with the rubric.

The dispersion of values around the mean is significantly smaller by using the rubric. The highest average deviation is 0.48 grade points. In contrast, without the rubric, the maximum average deviation from the mean is 0.75 grade points. However, particularly as project complexity increases, the dispersion in evaluations with the rubric also tends to increase. There could be two reasons for this. First, evaluating complex structures requires more knowledge in assessment, making it more challenging for inexperienced evaluators. It may indicate inexperienced assessors´ inability to differentially assess complexity, as well as their tendency to uniformly assign a good grade. Second, a smaller evaluation dispersion without the rubric does not necessarily indicate a better quality of assessment. Rather, as the note scale stops, a ceiling effect occurs. The assessment results show a ceiling effect, where many projects are concentrated at the higher end of the score range, making it difficult to distinguish between them. It seems, project 10 is assessed as attaining the highest level by almost all evaluators without a rubric. Obviously, this project works as a standard in comparison to other projects because it is the most complicated example. That is the probable explanation for the highest score on the open grading. But if the projects are mapped to a rubric standard, project 10 does not achieve the best possible grade because it does not fulfil all the requirements. This project, like project 8, has a complicated structure, so evaluators probably have difficulties scoring it, even with a rubric. Therefore, this could explain higher dispersion in the evaluation of

complicated projects. The second reason could be that at higher levels; the rubric allows for more room for interpretation and evaluative freedom. Overall, it can be still said that in most projects (except projects 8 and 10), the average deviation from the group mean is significantly smaller when using the framework, as Fig. 5 clearly shows.

An individual comparison of the ratings by example person (light grey dot) is shown in the following diagram (Fig. 6). This is a participant who awards a very different rating, both with and without the rubric. The participant's ratings without the rubric revealed a ceiling effect, as most projects were scored highly, often receiving A grades. In contrast, when using the rubric, their evaluations became more nuanced and differentiated, indicating a more refined assessment of the projects.

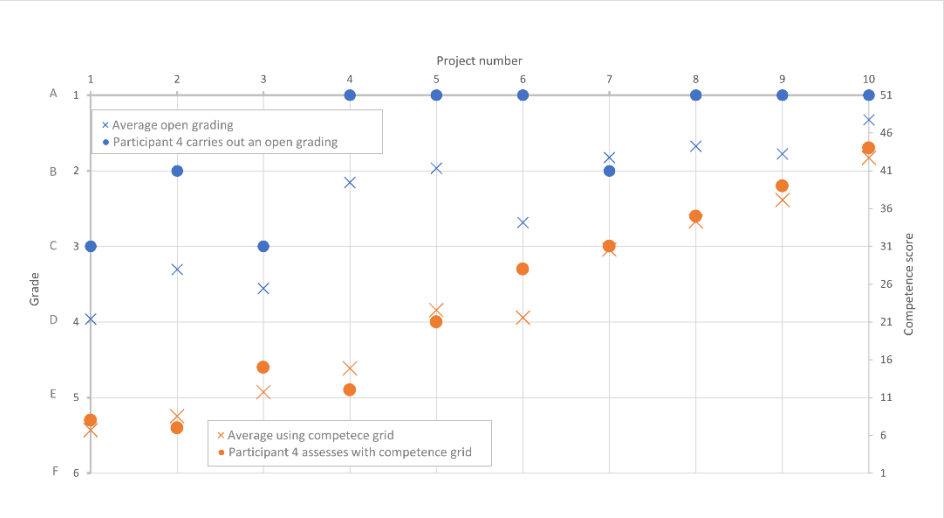


Figure 6: The results for a single example participant. Orange dots mark the evaluation with a rubric; blue dots mark the evaluation without a rubric. Corresponding group average is marked with crosses.

The example data set shows the effects of the rubric. Originally, this participant rated projects significantly higher than the average. For example, project 4 deteriorates from a very good (1, A) grade to a "fail" (5, E). Projects 1, 3, 5, 6, 8 also experience a significant deterioration. This is an interesting phenomenon that could not be clarified within the framework of this evaluation. There was a tendency for all evaluators to be significantly higher at open grading. Furthermore, when the results of example person are compared with the mean values, it is noticeable that, with the rubric, the assessment is closer to the general mean values. The individual mean

deviation is 0.04 grading points (2.12 of 51 points). Except for one project (7), the results of open grading by example person are far from the average grading. The mean deviation with open grading is 0.86 grading points. The rubric allows each rater to evaluate using the same scale as all other raters.

To examine the consistency of the evaluations and to determine whether the rubric improved the objectivity of grading, the interrater reliability (IRR) was calculated for both evaluation phases. For the free grading phase without predetermined criteria, Kendall's coefficient of concordance (W) was applied, as this method is appropriate for ordinal data such as school grades (Gibbons, 1993; Olson et al., 2003; Venugopal et al., 2024). For the rubric-based evaluation, the raw scores were first converted into the German grading system with increments of 0.25 (e.g., 5.81 → 5.75) to ensure a direct comparison with the free grading phase. A higher Kendall's W indicates greater agreement among raters, with values ranging from 0 (no agreement) to 1 (perfect agreement) (Olson et al., 2003).

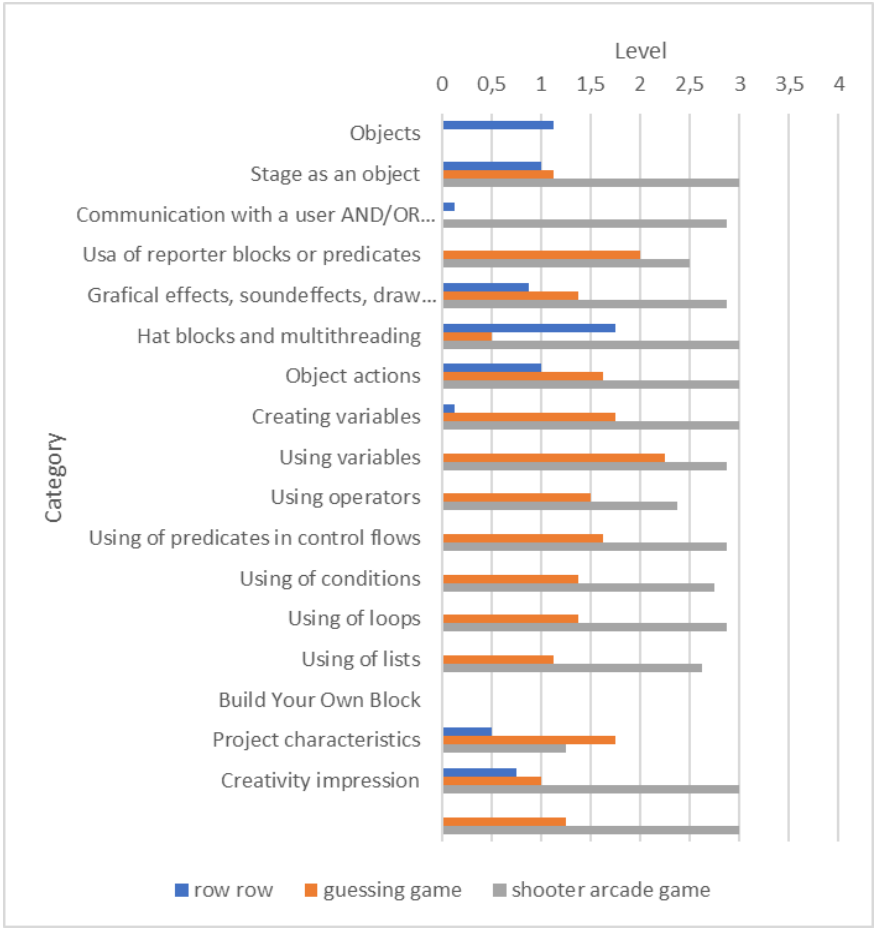


Figure 6: Comparison of results for individual projects by category. Selection of three projects. The picture shows an average of all graders.

This quantitative analysis provides an objective measure of the reliability of the evaluation process and demonstrates whether the rubric successfully reduced subjective variation in grading. The following section first presents **the Kendall's W values** for both evaluation phases and compares the level of agreement among raters. The interrater reliability analysis revealed a clear difference between the two evaluation phases. Without the rubric, the agreement among the nine raters was **moderate to good**, with a Kendall's W of **0.634** ($p < .001$). When using the developed rubric, the level of agreement increased substantially to **W = 0.940** ($p < .001$), indicating **very high to almost perfect concordance** between raters.

This result demonstrates that the rubric not only provides a structured framework for evaluation but also significantly reduces subjective variation in grading. The substantial increase in Kendall's W suggests that the competence grid helped the raters to apply more consistent and comparable evaluation criteria, thus improving the reliability of the assessment process.

Moreover, the designed rubric enables a more refined analysis of project quality at the individual level, providing a detailed breakdown of each project's strengths and weaknesses. Fig. 7 illustrates this capability, presenting a comparative analysis of three exemplary projects, highlighting their distinct characteristics and achievements. This nuanced evaluation allows educators to provide targeted feedback, fostering growth and improvement in each student's programming skills.

This representation method can help to break down each assessment individually into strengths and weaknesses as needed. Using the evaluation results, it is possible to explicate single components and compare them. Specifically in this example, the selected sample dataset could be reviewed in terms of existing concepts and the level of proficiency achieved on average. For example, in the category Use of conditions, the "shooter arcade game"- project achieves a high level of mastery, "guessing game" project shows moderate expertise, and the "row row" project lacks understanding in this area. On the base of this analysis method, it is possible to evaluate learning goals and correlated results and to give more detailed feedback on each project.

6. Discussion and Conclusion

The results of this study demonstrate that the rubric, with its 17 categories, is a comprehensive tool for evaluating programming projects. The primary aim of creating a structured description for creative block-based programming projects was successfully addressed with this rubric, providing a clear and systematic framework for assessment. Statistical analysis confirmed the reliability of the rubric. Kendall's W showed a high degree of agreement between assessors, demonstrating that the rubric supports consistent assessments by different assessors. At the same time, the distribution of scores indicated a possible ceiling effect, as particularly good projects achieved the highest possible score in several categories. This result shows that future iterations of the rubric could benefit from the addition of more advanced descriptions in order to better distinguish particularly high-performing projects. The lack of correlation between categories suggests that each category provides unique insights into the project's quality. As a result, the number of categories cannot be reduced without compromising the effectiveness of the evaluation when the rubric is used to derive a grade. However, for purely qualitative evaluations, certain categories may be excluded, particularly when specific aspects have not been covered during instruction.

The comparison of the two assessment forms clearly demonstrated that using a rubric led to criterion-led assessment, significantly reducing the average deviation of grades and thereby improving comparability between evaluators. Qualitative feedback from the raters confirmed their satisfaction with the tool, highlighting its clarity, perceived objectivity, and the sense of “clear conscience” when grading. The included source code examples were particularly valued, especially by less experienced assessors.

In terms of feasibility, evaluating programming projects with the rubric proved manageable. Assessing a single project required about nine minutes, totalling roughly 270 minutes for a class of 30 students. While no empirical data exist for grading times in computer science, this workload is comparable to grading a standard mathematics test, which typically takes around 360 minutes (Frank et al., 2023) . Thus, the rubric is not only reliable and comprehensive but also practical for classroom use, even in larger cohorts.

Nevertheless, the rubric has limitations. Its construction is based on student projects and qualitative expert assessment. As a result, not all possible Snap! categories are currently covered in the rubric, indicating a need for ongoing research and refinement. There is room for further differentiation of category descriptions, and higher proficiency levels would benefit from additional examples to support the application of the rubric.

Implementing the rubric in diverse educational contexts may pose challenges, particularly when teachers have limited experience with programming and assessment. The results of this study indicate that the rubric can be especially useful in such cases, as it helps to harmonize evaluations and align them with the mean value, as illustrated in Figure 6. These findings highlight the potential of the rubric to support less experienced teachers and suggest that future research should explore strategies to further facilitate its effective use.

Currently, there is no standardized, empirically validated framework for the evaluation of block-based programming projects. Existing approaches vary widely and are often designed for standardized, task-based contexts rather than authentic, open-ended projects. This study contributes to filling this gap by providing a structured, qualitative instrument for assessing Snap! projects, while also laying the groundwork for future comparative studies and broader validation efforts.

The developed rubric may also be applicable to other block-based programming languages such as Scratch. However, this potential transferability was not examined within the scope of the present study. Future research should therefore investigate its suitability across different programming environments to validate and possibly extend its applicability. From the students' perspective, the rubric can also serve as a reference framework to understand expectations and support self-assessment. Finally, by providing clear, criterion-based guidance, the rubric helps to overcome common challenges in evaluating problem-solving skills within the computational thinking process, particularly in the areas of algorithmic design, parallelization, iteration, and automation.

References

Andrade, H. G. (2000). *Using Rubrics to Promote Thinking and Learning*.

Balouktsis, I. (2016). Learning Renewable Energy by Scratch Programming. *Επιστημονική Επετηρίδα Παιδαγωγικού Τμήματος Νηπιαγωγών Πανεπιστημίου Ιωαννίνων*, *9*(1), 129. https://doi.org/10.12681/jret.8916

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the ACM*, *60*(6), 72–80. https://doi.org/10.1145/3015455

Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013). Hairball: Lint-inspired static analysis of scratch projects. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 215–220. https://doi.org/10.1145/2445196.2445265

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*.

Da Cruz Alves, N., Gresse von Wangenheim, C., & Hauck, J. C. R. (2019). Approaches to Assess Computational Thinking Competences Based on Code Analysis in K-12 Education: A Systematic Mapping Study. *Informatics in Education*, *18*, 17–39. https://doi.org/10.15388/infedu.2019.02

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, *58*(1), 240–249. https://doi.org/10.1016/j.compedu.2011.08.006

Döring, N., & Bortz, J. (2016). *Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften*. Springer Berlin, Heidelberg. https://doi.org/10.1007%2F978-3-642-41089-5

Frank, M., Thomas, H., & Martin, R. (2023). *Arbeitszeit und Arbeitsbelastung von Lehrkräften an Schulen in Sachsen 2022: Ergebnisbericht*. https://doi.org/10.47952/gro-publ-172

Funke, A., & Geldreich, K. (2017). Measurement and Visualization of Programming Processes of Primary School Students in Scratch. *Proceedings of the 12th Workshop on Primary and Secondary Computing Education - WiPSCE '17*, 101–102. https://doi.org/10.1145/3137065.3137086

Garcia, D. D., Harvey, B., & Segars, L. (2012). CS principles pilot at University of California, Berkeley. *ACM Inroads*, *3*(2), 58. https://doi.org/10.1145/2189835.2189853

Gesellschaft für Informatik (Hrsg.). (2016). Bildungsstandards Informatik—Sekundarstufe II. *Empfehlungen der Gesellschaft für Informatik e. V. erarbeitet vom Arbeitskreis »Bildungsstandards SII«*, *183/184*, 88.

Gibbons, J. (1993). *Nonparametric Measures of Association*. SAGE Publications, Inc. https://doi.org/10.4135/9781412985291

Gummels, I. (2020). *Wie kooperatives Lernen im inklusiven Unterricht gelingt*. Springer Spektrum Wiesbaden. https://doi.org/10.1007/978-3-658-29114-3

Harel, I., Massachusetts Institute of Technology, & Media Laboratory (Hrsg.). (1993). *Constructionism: Research reports and essays, 1985-1990* (2. print). Ablex Publ. Corp.

Hattie, J. (2009). *Visible learning: A synthesis of over 800 meta-analyses relating to achievement*. Routledge.

Jürgens, E., & Lissmann, U. (2015). *Pädagogische Diagnostik*. Beltz Verlag.

Koh, K. H., Basawapatna, A., Nickerson, H., & Repenning, A. (2014). Real Time Assessment of Computational Thinking. *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 49–52. https://doi.org/10.1109/VLHCC.2014.6883021

Koray, A., & Bilgin, E. (2023). The Effect of Block Coding (Scratch) Activities Integrated into the 5E Learning Model in Science Teaching on Students' Computational Thinking Skills and Programming Self-Efficacy. *Science Insights Education Frontiers*, *18*(1), 2825–2845. https://doi.org/10.15354/sief.23.or410

Krugel, J., & Ruf, A. (2020). *Learners' perspectives on block-based programming environments: Code.org vs. Scratch*. http://doi.acm.org/10.1145/3421590.3421615

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education*, *10*(4), 1–15. https://doi.org/10.1145/1868358.1868363

Mladenović, M., Mladenović, S., & Žanko, Ž. (2020). Impact of used programming language for K-12 students' understanding of the loop concept. *International Journal of Technology Enhanced Learning*, *12*(1), 79. https://doi.org/10.1504/IJTEL.2020.103817

Modrow, E. (2018). *Informatik mit Snap!, Snap! In Beispielen*. http://ddi-mod.uni-goettingen.de/InformatikMitSnap.pdf

Moreno-León, J., Román-González, M., Harteveld, C., & Robles, G. (2017). On the Automatic Assessment of Computational Thinking Skills: A Comparison with Human Experts. *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2788–2795. https://doi.org/10.1145/3027063.3053216

Olson, L., Schieve, A. D., Ruit, K. G., & Vari, R. C. (2003). Measuring Inter-rater Reliability of the Sequenced Performance Inventory and Reflective Assessment of Learning (SPIRAL): *Academic Medicine*, *78*(8), 844–850. https://doi.org/10.1097/00001888-200308000-00021

Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2019). Exploring children's learning experience in constructionism-based coding activities through design-based research. *Computers in Human Behavior*, *99*, 415–427. https://doi.org/10.1016/j.chb.2019.01.008

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. BasicBooks.

Perera, P., Tennakoon, G., Ahangama, S., Panditharathna, R., & Chathuranga, B. (2021). A Systematic Mapping of Introductory Programming Languages for Novice Learners. *IEEE Access*, *9*, 88121–88136. https://doi.org/10.1109/ACCESS.2021.3089560

Price, T. W., & Barnes, T. (2015). Comparing Textual and Block Interfaces in a Novice Programming Environment. *Comparing Textual and Block Interfaces in a Novice Programming Environment*. https://doi.org/10.1145/2787622.2787712

Resnick, M. (2014). *Give P's a chance: Projects, peers, passion, play*.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Resnick, M., Silverman, B., Kafai, Y., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., & Silver, J. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60. https://doi.org/10.1145/1592761.1592779

Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, 59–66. https://doi.org/10.1145/2493394.2493403

Shute, V. J. (2008). Focus on Formative Feedback. *Review of Educational Research*, *78*(1), 153–189. https://doi.org/10.3102/0034654307313795

Svedkijs, A., Knemeyer, J.-P., & Marmé, N. (2022). Förderung von Computational Thinking durch ein digitales Leitprogramm zur blockbasierten Programmiersprache Snap! In B. Stadl (Hrsg.), *Digitale Lehre nachhaltig gestalten*. Waxmann Verlag. https://doi.org/10.31244/9783830996330

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. https://doi.org/10.1016/j.chb.2018.11.038

Venugopal, V., Dongre, A., & Kagne, R. N. (2024). Development of an analytical rubric and estimation of its validity and inter-rater reliability for assessing reflective narrations. *The National Medical Journal of India*, *36*, 323–326. https://doi.org/10.25259/NMJI_732_21

Weintrop, D., & Wilensky, U. (2015). To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-Based Programming. *Proceedings of the 14th International Conference on Interaction Design and Children*, 199–208. https://doi.org/10.1145/2771839.2771860

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)*, *18*(1), 3.

Weintrop, D., & Wilensky, U. (2018). Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. *ACM Transactions on Computing Education*, *18*(1), 1–25. https://doi.org/10.1145/3089799

Wen, F.-H., Wu, T., & Hsu, W.-C. (2023). Toward improving student motivation and performance in introductory programming learning by Scratch: The role of achievement emotions. *Science Progress*, *106*(4), 00368504231205985. https://doi.org/10.1177/00368504231205985

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. *ACM Transactions on Computing Education*, 215–220. https://dl.acm.org/doi/10.1145/2157136.2157200

Wiliam, D. (2011). *Embedded formative assessment*. Solution Tree Press.

Wolf, K., & Stevens, E. (2007). The Role of Rubrics in Advancing and Assessing Student Learning. *The Journal of Effective Teaching*, *7*(1), 3–14.

Zhang, N., & Biswas, G. (2019). Defining and Assessing Students' Computational Thinking in a Learning by Modeling Environment. In S.-C. Kong & H. Abelson (Hrsg.), *Computational Thinking Education* (S. 203–221). Springer Singapore. https://doi.org/10.1007/978-981-13-6528-7_12

International Journal of Computer Science Education in Schools, February 2026, Vol. 7, No. 3
ISSN 2513-8359

Towards a consensus on program elements of specialized computer science / information technology (CS/IT) programs in high schools: A Delphi study

Jonathan D. Becker, J.D., Ph.D.
Virginia Commonwealth University

Amy D. Corning, Ph.D.
Virginia Commonwealth University

Jon S. Graham, M.A., M.S.
Virginia Commonwealth University

James T. Carrigan, L.C.S.W., M.S.W.
Virginia Commonwealth University

DOI: 10.21585/ijcses.v7i3.236

Abstract

In our increasingly technological and advanced times, demand for K-12 education in computer science and information technology (CS/IT) is growing. Current data offer insight into student access to computer science education and course-taking. In addition to the expansion of individual course offerings, there is also a growing number of specialized CS/IT programs in high schools. However, there has been no systematic attempt to document the landscape of those programs. This study is part of a larger landscape study of secondary CS/IT programs in Virginia and uses a consensus-based approach to identify the common elements that expert and practitioner panelists believe should be included in such a program. The results reveal strong consensus on a wide range of program goals, activities, and curricular elements, suggesting that there are many opportunities to create purposeful and coherent CS/IT programs in high schools.

Keywords: computer science, information technology, high school, programs

1. Introduction

1.1 Understanding the landscape of computer science/information technology (CS/IT) programs in high schools

Educators and other stakeholders are keenly aware of the need for high-quality computing education at the secondary level – on the one hand, to enhance the diversity and thus the vibrancy and sustainability of the computing workforce, and on the other, to prepare citizens for a world increasingly reliant on computing. The need for computing education exists at multiple levels of schooling, but high school may be a critical juncture, when contexts and experiences influence students' engagement, self-efficacy, and belonging in ways that affect their interest and post-secondary persistence, with respect to both STEM (e.g., Bottia et al., 2015, 2018; Legewie & DiPrete, 2014) and computing specifically (Eisenhart & Allen, 2020; Master et al., 2016; NASEM, 2021).

On a state level, Virginia's status as a technology hub lends particular urgency to issues of STEM education generally and computing education in particular. Over the past two decades, the Commonwealth of Virginia has invested considerable resources into STEM education programs at the high school level – in part through the establishment of schools and programs focusing on computing, computer science, and information technology. Virginia was also one of the first states to develop Computer Science Standards of Learning (Virginia Department of Education [VDOE], 2022a), and the first state to adopt a K-12 computer science framework (Crowder et al., 2020). A number of Virginia's STEM-focused high schools and programs offer computing and information technology-oriented education, and many of these schools and programs appear to share common elements, including emphasis on advanced mathematics and computer science coursework, authentic and hands-on learning, projects and internships, career exposure, development of workplace skills, and opportunities to earn college credits.

This article reports on one part of a more comprehensive environmental scan of specialized high school computer science/information technology (CS/IT) programs in Virginia. In partnership with the Virginia Department of Education (VDOE), we have conducted a detailed census of schools and programs designed to support secondary students in pursuing computing education. One part of the project involves web- and survey-based research to gather information about the programs in terms of characteristics such as

selectivity/inclusivity, program length, cohesiveness of program community, location, and student demographics. That is, the goal of that part of the project is to be able to describe what is offered by programs. The part of the research project reported herein is an attempt to understand what experts believe should be offered by specialized high school CS/IT programs.

1.2 Research Question and Significance

To understand beliefs about what specialized high school CS/IT programs should offer students, we used the Delphi Method among a panel of CS education experts whose professional backgrounds ranged from classroom teacher to university professor. Through three rounds of questions posed to our panel, areas of consensus and dissensus emerged that allowed us to surface understanding of what specialized CS/IT programs should offer.

Specifically, the study was guided by the following research question: what common educational and experiential elements (e.g., advanced courses, degrees/college credits, credentials, hands-on/authentic experiences, internships, workplace skill development) do educators believe are important for specialized secondary CS/IT programs/schools to provide?

This study is significant because while a number of Virginia's STEM-focused high schools and programs offer CS/IT-oriented education, and many of them appear to share common elements, at present we have no systematic understanding of the prevalence of these elements across schools/programs. Nor do we have a conceptual map of the outcomes to which they are intended to lead. This study involves critical first steps that lay the groundwork for understanding similarities and differences among CS/IT-focused programs and schools and will help us develop appropriate measures for evaluating their effectiveness.

2. Literature Review

2.1 Defining CS Education

Computer science as a discipline has long struggled to define itself as distinct from other disciplines, including mathematics and engineering. Today, with the growth of fields like artificial intelligence, data science,

cybersecurity, and human-computer interaction, new questions of the disciplinary boundaries of computer science as a discipline have emerged.

These disciplinary and definitional struggles are evident in computer science education. Those charged with teaching within the discipline, from elementary school to postsecondary education, have had to figure out what their students need to know and what skills they need within a rapidly changing society. In K-12 education, many states, including Virginia, have adopted a definition of computer science drawn from Tucker et al. (2006). According to this definition, computer science is "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society" (p. 2). Virginia describes computer literacy (the general use of computers and programs), educational technology (applying computer literacy to school subjects), and digital citizenship (the appropriate and responsible use of technology) as the building blocks of computer science (Commonwealth of Virginia Board of Education, 2017). Furthermore, information technology shares key principles with computer science but is largely focused on applications of computer science, such as software installation as opposed to software development.

These definitional and disciplinary overlaps present challenges to educators and policymakers in charge of developing courses and programs of study. They also present challenges to policymakers and researchers attempting to understand and report on computer science offerings in schools. As an example, for the purposes of our study, we settled on the "CS/IT" nomenclature to ensure that the research addressed the full range of programs that might include CS-related education.

2.2. The Demand for CS Education

Definitional challenges notwithstanding, there is clear demand for and growth in computer science education in K-12 education. According to a report from Code.org (Code.org et al., 2023), in 2023, 57.5% of U.S. public high schools offered at least one foundational computer science course. This percentage is up from 53% in 2022 and represents the largest year-to-year growth documented by Code.org. Furthermore, across the 35 states that provided relevant data, 5.8% of high school students were enrolled in one of those foundational computer

science courses. Finally, states allocated more than $120 million for computer science education, the most ever allocated in one year.

This growth in CS education comes from a few different directions. From a federal policy perspective, while CS education had been discussed for decades, CS education was specifically mentioned in federal education policy for the first time in the 2015 Every Student Succeeds Act (ESSA) which reauthorized the Elementary and Secondary Education Act (ESEA). Specifically, computer science was included in the definition of a "well-rounded" education in section 8102 of ESEA of 1965. However, CS education got a major boost from the federal government when, in 2016, the Computer Science for All initiative was launched. That initiative offered $4 billion for states and $100 million for school districts that agreed to expand computer science education over ten years (Marshall & Grooms, 2022).

Undoubtedly, this demand is also fueled by societal changes and the changing workforce. That is, to the extent that K-12 education is aimed at preparing students for the workforce, schools must help students explore information technology and computer science (Muraski & Iversen, 2022). And, as Marshall & Grooms (2022) document, industry and private sector actors have been significantly involved in advocacy of CS education, though such influence networks are often focused on private interests and not on broader policy goals including equity or equality of opportunity.

2.3. The Effects of CS Education

Alongside the growing demand for CS education, there has been no shortage of research on pedagogical techniques within CS education. However, there is less research on its overall impacts. It may be a bit early to assess the effect of this new policy emphasis, but there has been some research on the relationship between CS education and the development of skills such as computational thinking as well as the relationship between CS course-taking in K-12 education and the selection of STEM majors in college.

Outcomes that have been examined include interest in CS (Clarke-Midura et al., 2020; Sabin et al., 2017; Starrett et al., 2015; Webb & Rosson, 2011) and CS self-efficacy (Aivaloglou & Hermans, 2019; Aritajati et al., 2015;

Elizabeth Casey et al., 2017). The research on the development of computational thinking skills via CS education is quite robust. Lee et al. (2022) conducted a systematic review of the research on CS education and K-12 students' computational thinking (CT) skills and found "strong evidence that CS education promotes the development of students' CT in the K-12 setting while improving students' creative and critical thinking skills" (p. 10). Considering longer term outcomes, computer science course-taking in high school has been associated with the selection of STEM majors in both two-year and four-year institutions (Lee, 2015; Giani, 2022; Armoni & Gal-Ezar, 2023).

2.4. The Challenges for CS Education

Nearly a decade into the "CS for All" era, one of the most significant challenges CS education faces is that CS education has not been for all. The most recent State of Computer Science Education report from Code.org shows that schools in rural and urban areas, as well as smaller schools, are less likely to offer a foundational CS course. Also, "Black/African American students, Hispanic/Latino/Latina/Latinx students, and Native American/Alaskan students are less likely to attend a school that offers foundational computer science" (p. 5). CS-related outcomes are inequitable as well. Based on data from the International Computer and Information Literacy Study (ICILS) in 2018, Karpiński et al. (2021) found that "...regardless of what proxy for socioeconomic status is employed, and in line with expectations, students from more advantaged backgrounds perform better in both [Computer and Information Literacy] CIL and [Computational Thinking] CT tests, compared with their peers from less advantaged backgrounds" (Karpiński et al., 2021, p. 3).

The availability of well-qualified CS education teachers is an additional equity challenge for the field. "In order to fully realize the promise of computing education, we need to ensure that students have highly qualified teachers with knowledge of computing, and that teachers are implementing pedagogical approaches that center students' lived experiences" (Shah and Yadav, 2023. P. 469). For both CTE and general education CS courses in K-12 education, the challenge is finding teachers who have both the pedagogical and content knowledge needed to best facilitate learning in computer science.

Finally, a real challenge in the K-12 CS education space is figuring out what students need to know and be able to do as a result of taking CS courses and enrolling in CS/IT programs in order to….??. In an increasingly technological society, those choices are important but difficult. Therefore, this study aims to inform those curricular conversations.

3. Research Design

3.1 Delphi method study

Our research employed the Delphi method to elicit beliefs from a panel of experts on computer science education to see where there is consensus (or dissensus) on the goals and characteristics of specialized secondary CS/IT programs. In the Delphi method, "[T]he aim is to reach agreement or a convergence of opinion, and the structured process allows for the effective amalgamation of information" (Drumm et al., 2022, p. 3). There are variations across studies in how the Delphi Method is carried out, but, generally, a panel of experts is asked to complete multiple rounds of questionnaires. The first-round questionnaire includes mostly open-ended questions; data from that questionnaire are used to generate a second questionnaire consisting of five-point, agree-disagree scale questions. In most cases, a third questionnaire is used to seek consensus and/or prioritization in areas where there was consensus.

3.2 Study Sample

Our study included three rounds of questionnaires (described below) administered to a panel of individuals with experience in CS education and likely to be informed about specialized computer science and information technology programs for Virginia high school students. We cast a wide net to identify potential participants, drawing on sources including known CS/IT program directors, professional contacts, individuals recommended by our partner, VDOE, etc. We chose not to include representatives from business or industry. At this point in our research, we were primarily concerned with understanding the needs of students and families and the capacities of teachers, schools, and programs – not with workforce or employer demand or pipelines, though of course the different realms are interconnected. In addition, some schools and programs actively work with

industry partners to better understand their needs through curriculum, credentials offered, etc., so business interests are to some extent already reflected.

With these goals in mind, we developed a list of 56 potential participants. We then selected 39 to contact, with a view to including individuals in a range of roles at different types of institutions and organizations, in different regions of Virginia, who might represent a spectrum of perspectives on specialized CS/IT programs. We contacted potential participants by email, inviting them to participate, explaining the purpose of the study and providing details, and offering them a $25 gift card incentive for completing all three rounds. Of those we contacted, 23 agreed to participate, and 17, or 44%, completed all three question rounds. Five were not willing to participate (13%), and 11 never responded (28%). Three referred us to other individuals within their institution who were better positioned to respond to our request; one of the three referrals agreed to participate, and we replaced the three original invited participants with the three referrals for the purpose of these calculations.

3.3 Study Questionnaires

The Round 1 questionnaire consisted of nine open-ended questions (some in multiple parts). Panelists were asked about their perceptions of specialized high school CS/IT programs, including their intended goals, skills they should foster, what sorts of experiences they should offer; panelists were also asked about their views on admissions approaches and program recruitment. Round 2 was designed to ascertain the degree of consensus about program goals and other elements. The themes and language used by participants in Round 1 formed the basis for seven sets of closed-ended questions and three further open-ended questions in Round 2. Questions again asked about program goals, skills taught/learned, experiences offered, and approaches to admissions – all in closed-ended format. Foth et al. (2016) reviewed studies using the Delphi method in nursing and found that the studies that predefined consensus described it as a percentage of agreement for an item, "...usually 60% agreement or higher (median = 75%)" (p. 118). Diamond et al. (2014) found a similar median consensus level in their review of Delphi studies. We defined "consensus" as 70% agreement or higher.

The purpose of Round 3 was twofold. First, we wanted to encourage participants to consider their own perspectives in light of others' responses before responding to a final set of questions, so we provided

participants with a summary report of the Round 2 findings. Second, since there was so much consensus generated in Round 2, in the final round we chose to ask participants which of the consensus elements they would prioritize in specialized high school CS/IT programs. For example, there was strong consensus on a large number of goals for such programs, so we asked the panelists to tell us which three goals should be emphasized. Participants received both the Round 2 summary and the Round 3 questionnaire at the same time.

4. Findings

The findings about common elements of specialized HS CS/IT programs are described below, and are organized into three categories: program goals, program activities, and program skills/competencies.

4.1 Program Goals

Using the data from Round 1, we identified nine possible goals for specialized high school CS/IT programs. The goals clustered around two broad themes: specialized CS/IT education as a means to pursue learning about CS/IT (e.g., the item "allowing students explore in interest in CS/IT"), and specialized CS/IT education as preparation for post-secondary activities (e.g., "providing students with a foundation for post-secondary education in CS/IT."

On eight of the nine goals, Round 2 consensus was nearly complete, with over 90% of the panelists agreeing or strongly agreeing with each, as shown by the orange bars in Figure 1. Even the goal endorsed by the smallest percentage – helping students obtain a job in CS/IT after HS graduation – was supported by nearly 80%. More than 70% of panelists strongly agreed with six of the nine listed goals. The three that did not reach consensus based on strong agreement were promoting access to CS/IT for students historically marginalized in CS/IT education (67% strongly agreed), providing students with a strong foundation for post-secondary education in CS/IT (61% strongly agreed), and helping students obtain a job in CS/IT after HS graduation (17% strongly agreed).

In Round 3, panelists were asked to choose the three of the nine goals from Round 2 that they saw as most important for programs to emphasize. The blue bars in Figure 1 show the percentage of participants who selected

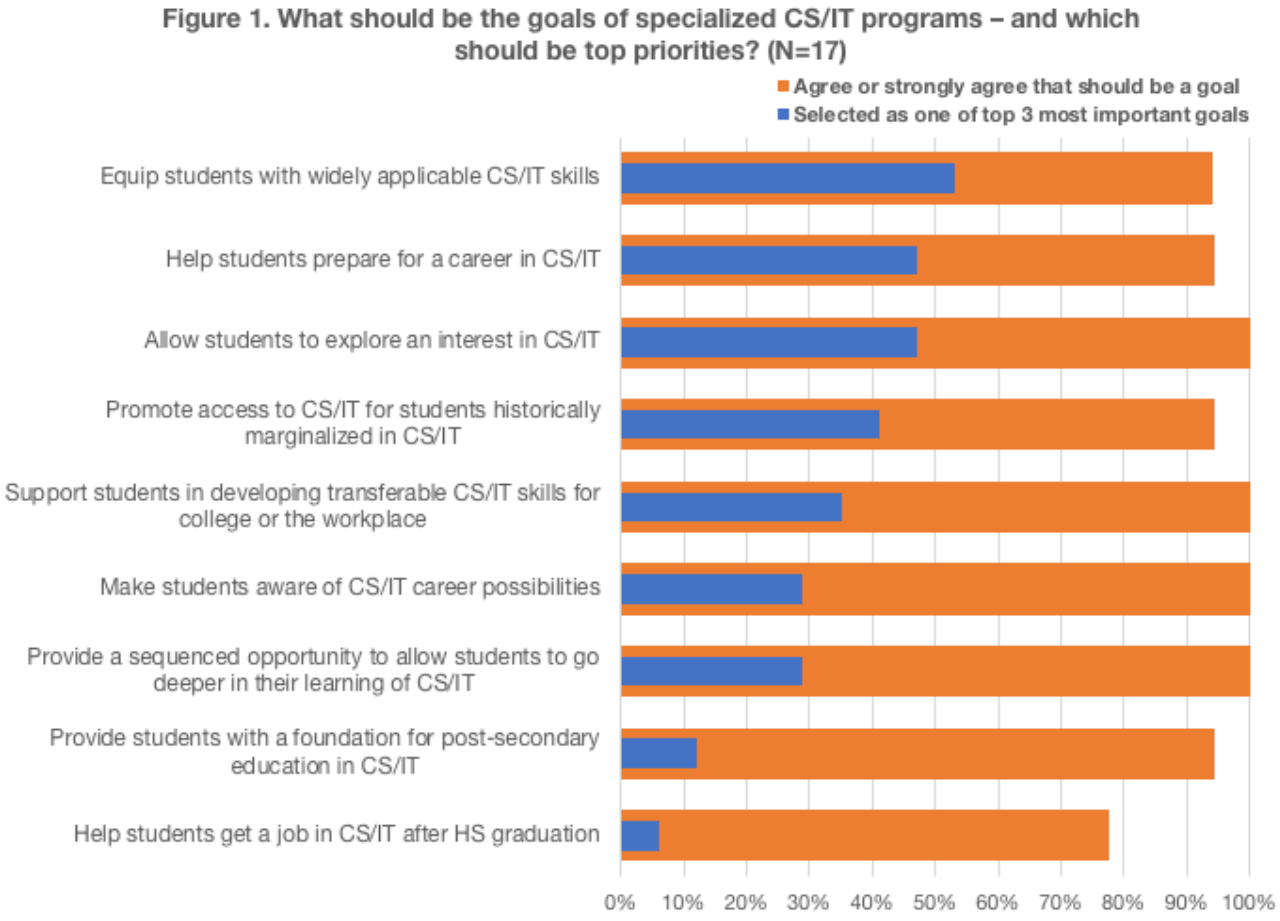
each goal as one of the three most important, with priority goals falling into three broad groups. The first group, selected by more than 40%, included equipping students with widely applicable CS/IT skills, helping students prepare for a career in CS/IT, allowing students to explore an interest in CS/IT, and promoting access to CS/IT for students historically marginalized in CS/IT fields. Only the first of these was selected by more than half the panelists. The second group of prioritized goals included those selected by one third of panelists or slightly fewer, including: supporting students in developing transferable CS/IT skills for college or the workplace, making students aware of CS/IT career possibilities, and providing a sequenced opportunity to allow students to go deeper in their learning of CS/IT. Finally, two goals were selected by the lowest percentages of respondents: providing students with a foundation for post-secondary education in CS/IT and helping students get a job in CS/IT after high school graduation. Even though some programs espouse these goals, our panelists may have assigned them lower priority because each addresses the needs of only a subset of high schoolers.

Panelists also responded to an open-ended question on the Round 2 questionnaire inviting them to comment on their agree–disagree ratings, providing further insight into their thinking about program goals. Three themes emerged from those data. The first theme reflected a belief that specialized CS/IT programs should provide a wide range of experiences to students. These experiences included specific elements such as “multiple offerings for juniors and seniors, depending on what they want to do post-high school,” and more general recommendations such as “Opportunity and exploration. Life skills and knowledge, not necessarily career specific.” A second theme was the importance of CS/IT programs offering work-based learning and career readiness elements that prepare students for the workplace. The third theme involved the need for CS/IT programs to teach the technical skills necessary to be knowledgeable about and successful in computer science and information technology. Specific skills included cybersecurity, software development, programming, Unix, Python, Microsoft, and Google.

4.2 Program Activities

Using the data from Round 1 as our starting point, we identified 10 possible activities that participants thought specialized high school CS/IT programs should offer, and asked about each in a closed-ended, agree–disagree format in Round 2. Figure 2 below shows the activities, along with the percentage of panelists who agreed or

strongly agreed that they should be offered (the yellow bars) in the Round 2 questionnaire. Figure 2 also shows the percentage of panelists who chose each activity as one of the three most important to emphasize (in blue) in Round 3.



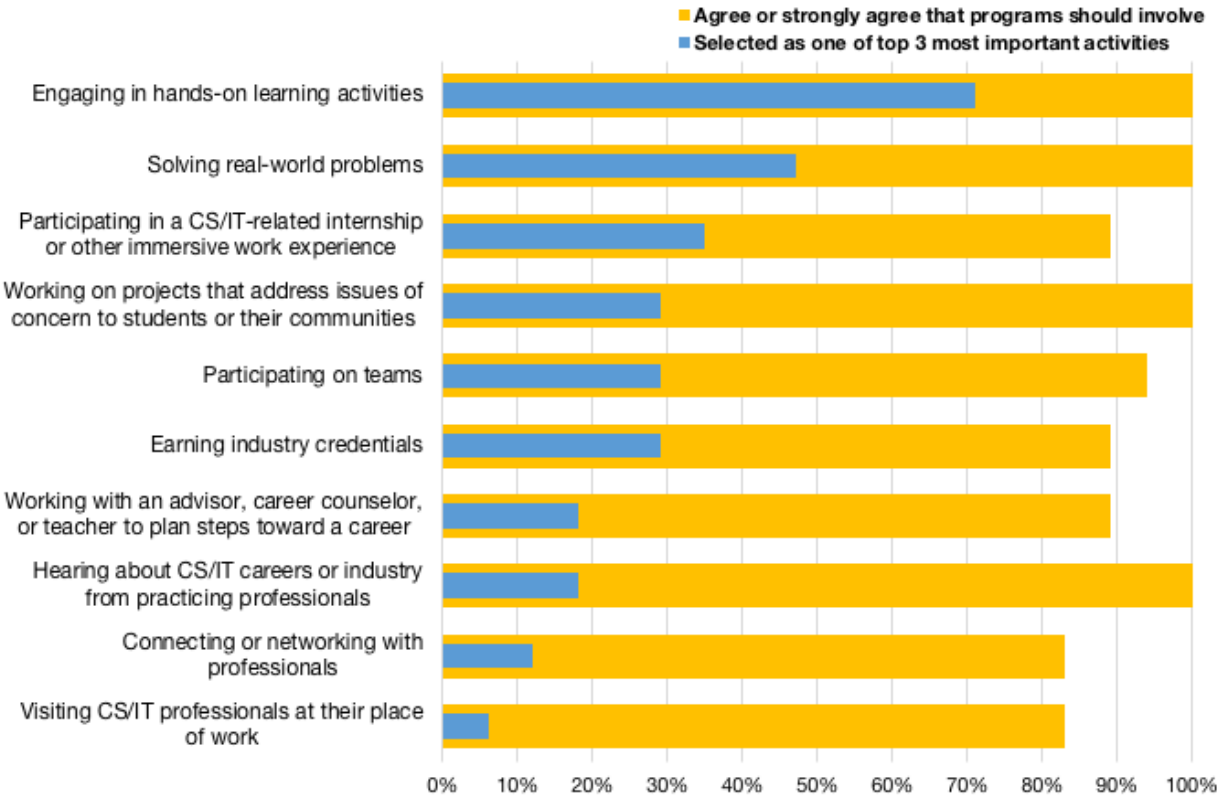
Consensus that programs should include the 10 activities was almost complete: approximately 90% of the panelists agreed or strongly agreed that each activity should be included in programs. The only activities that did not receive 90% support were “visiting CS/IT professionals at their place of work,” and “connecting or networking with professionals and others in the community.” Percentages strongly agreeing were the highest for hands-on learning activities (94%), solving real-world problems (89%), and participating on teams (78%).

Again, on the Round 3 questionnaire, respondents were asked to select the three activities that they thought were most important for specialized CS/IT programs to emphasize. Consensus was identified for only one of the ten

activities, "engaging in hands-on learning activities," with 71% of participants selecting the activity for emphasis. Considering the highest priority activities, there is an emerging consensus around constructivist-oriented pedagogy. That is, at least the first two highest priority activities reflect a possible consensus about the value of constructivist-oriented pedagogy. For example, one participant shared that they

> *...believe teachers need to be intentional about creating opportunities and projects where students have to work together. The curriculum for a lot of these programs really lends itself to students working independently and at their own pace. The more students can communicate and collaborate in the classroom, the better because that is reflective of how they will function, at least some of the time, in the workforce.*

Figure 2. Which activities should specialized CS/IT programs include – and which are top priorities? (N=17)



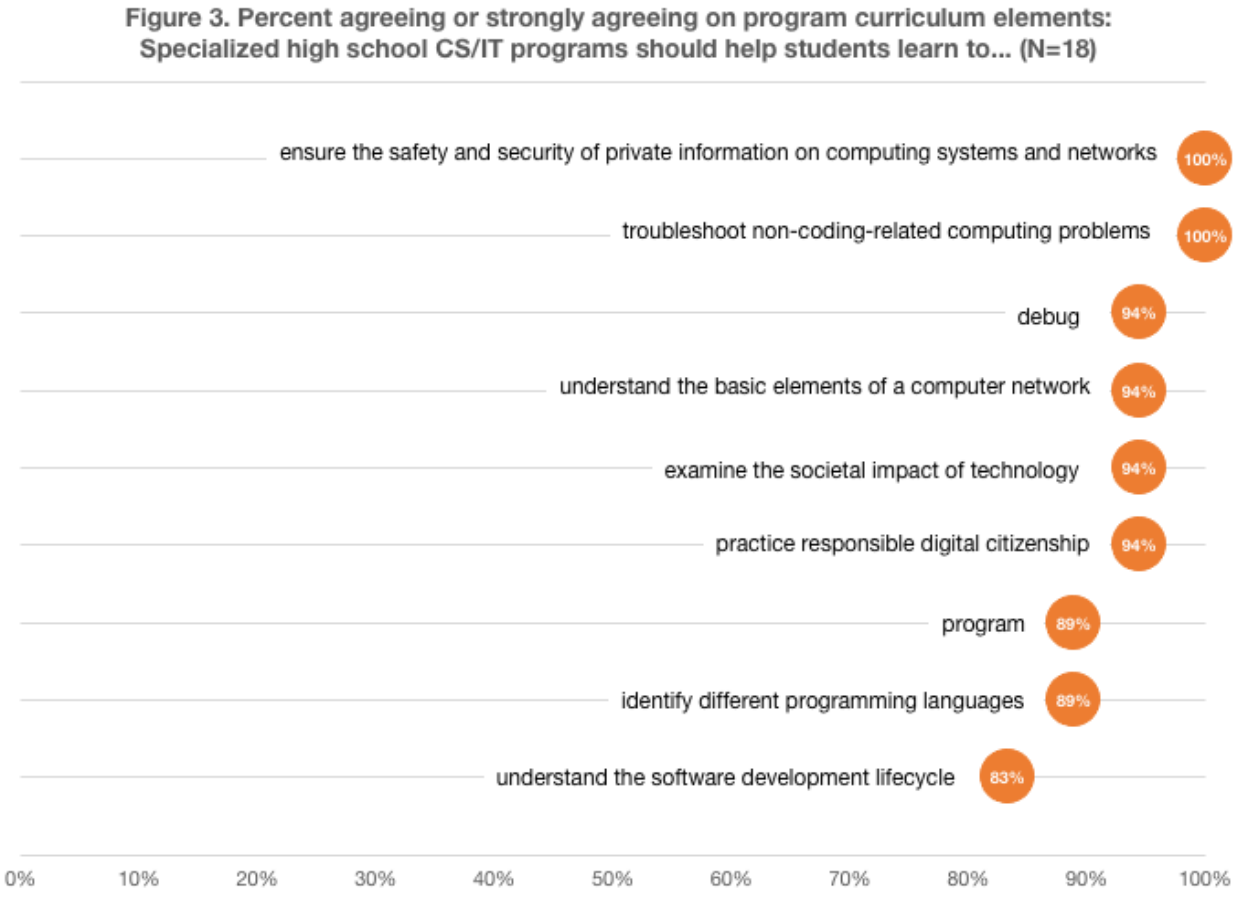
Despite the CTE focus of many CS/IT programs in Virginia, none of the six specific work-based learning elements was universally seen as important for programs to emphasize. Just over and just under one third felt that experiences such as participating in internships (35%) and earning industry credentials (29%) were among

the most important experiences to emphasize, while much smaller percentages prioritized working with an advisor, career counselor, or teacher to plan steps toward a career (18%), hearing about CS/IT careers or industry from practicing professionals (18%), connecting or networking with professionals (12%), or visiting CS/IT professionals at their place of work (6%). However, a total of 71% of panelists prioritized at least one of the six work- or career-oriented experiences, suggesting that panelists agreed on the value of work-based learning, just not the specific experiences.

4.3 Program Skills/Competencies

To identify potential skills or competencies that participants believed were important to include in specialized CS/IT programs, we analyzed responses to the Round 1 questionnaire. Our analysis generated nine potential CS/IT-related skills or competencies. In round 2, as Figure 3 shows, over 80% of panelists agreed or strongly agreed that program curricula should address each skill. Considering strong agreement only, 72% strongly agreed that specialized programs should help students learn to ensure the safety and security of private information on computing systems and networks, 67% strongly agreed that students should learn to troubleshoot non-coding-related problems, and 61% strongly agreed that students should learn to debug code.

In contrast to our approach for program goals and activities, we did not use the third-round questionnaire to ask the panelists to pick the three skills or competencies that programs should emphasize. Furthermore, skills and competencies are likely to vary across programs with different foci (e.g. a program within a programming pathway would have a different curriculum than a pathway focused on, say information security). In an open-ended question in Round 2, panelists had the opportunity to explain in their own words what they thought students should learn in a specialized high school CS/IT program. From these responses, one theme linked learning to the purpose of the program and the ostensible focus of the curriculum. As one participant put it, “I think it depends on what the purpose of the programs are: prep for job out of high school, prep for certification programs, prep for college courses, or prep for almost any career in STEM.”



This purpose-dependent theme was echoed by another participant, who stated "I think that some topics should be emphasized more or less by students, depending upon the student's focus more on CS or more on IT." Another noted the importance of program and curriculum responsiveness to diverse student interests and career plans: "we want to provide [sic] them with a variety of concepts so they can make informed choices about programs [sic] options after high school."

4.4 Summary of Common Program Elements

Table 1 lists the goals, activities, and skills/competencies that the panel of experts indicated should be the elements of a specialized high school CS/IT program. There was consensus (>70% agreed or strongly agreed) around each of the listed elements, which are sorted in order of level of consensus. The darkest shading reflects 100% consensus; the second darkest shade reflects a minimum of 89% consensus for activities and skills; the lightest gray represents 84% consensus; and white represents consensus between 70% and 83%. For goals and

activities, the three elements with highlighted borders were chosen by the highest number of panelists as key elements programs should emphasize.

Table 1. Common program elements as identified by the panel of experts

GOALS	ACTIVITIES	SKILLS/COMPETENCIES
Allow students to explore an interest in CS/IT	Engaging in hands-on learning activities	Ensuring the safety and security of private information on computing systems and networks
Support students in developing transferable skills for college or the workplace	Solving real-world problems	Troubleshooting non-coding kinds of computing problems
Make students aware of CS/IT career possibilities	Working on projects that address issues of concern to students or their communities	Debugging
Provide a sequenced opportunity to allow students to go deeper in their learning of CS/IT	Hearing about CS/IT careers or industry from practicing professionals	Understanding the basic elements of a computer network
Equip students with widely applicable CS/IT skills	Participating on teams	Examining the societal impact of technology
Help students prepare for a career in CS/IT	Participating in a CS/IT-related internship or other immersive work experience	Practicing responsible digital citizenship
Provide students with a foundation for post-secondary education in CS/IT	Earning industry credentials	Programming

Promote access to CS/IT for students historically marginalized in CS/IT	Working with an advisor, career counselor, or teacher to plan steps toward a career	Identifying different programming languages
Help students get a job in CS/IT after HS graduation	Visiting CS/IT professionals at their place of work	Understanding the software development lifecycle
	Connecting or networking with professionals and others in the community	

4. Discussion

At the end of the final questionnaire, we offered the following prompt to the panelists: Having now completed three rounds of questionnaires, how has the process made you think about specialized CS/IT programs or CS education in a new or different way? For example, one participant wrote, "I've thought about things I haven't considered before," and "it brought into focus that programs really are so limited and there are tough choices to be made. We cannot be everything to everyone." Similarly, one panelist offered a very good summary of the findings of this study by writing that the process had led them to have more respect for "building the program better. There are lots of moving parts."

The many moving parts are visible in the many different possible goals, activities, and curriculum elements identified in Round 1. Then, in Round 2, there was consensus that CS programs should reflect nearly all the nine possible goals and ten possible activities. Furthermore, when asked in Round 3 which three of the nine consensus goals they would emphasize, the goal that panelists picked most frequently was equipping students with widely applicable CS/IT skills. Panelists appeared to believe that specialized high school CS/IT programs should be broad enough such that students could either pursue postsecondary education in CS or be prepared for entry into the modern workforce. That belief informs as well as reflects the design of CS/IT programs. A significant proportion of the existing programs are under the CTE umbrella, and CTE tends to be organized around "career clusters" and pathways. Yet a number of those programs also include a dual enrollment option through which students can earn college credit, suggesting that students who study CS in high school have a range of future

options. As one participant noted, “I think we need to be conscious of not making HS programs only career oriented – it also needs to be a time for exploration and discovery.” Balancing the workforce development orientation of many specialized high school CS/IT programs with opportunities for exploration and discovery may be a real challenge.

From a curriculum and pedagogical perspective, the participants favored more constructivist-oriented programs. That is, the participants prioritized hands-on, project-based learning that addressed real-world problems of practice in computer science. There was wide consensus on the inclusion of high-quality work-based learning opportunities in specialized CS/IT programs, but those sorts of activities were prioritized at a lower level than other activities. These findings, too, present opportunities as well as challenges. That is, there are plenty of technologies and tools available for teachers to engage in CS activities that are oriented toward constructivism or, better, constructionism (CITE). That there are so many possibilities, though, is what presents a challenge. The curricular and pedagogical possibilities can be overwhelming.

Ultimately, we contend that the degree and breadth of consensus is reflective of the demand for CS education in the K-12 context. There is so much that we can be doing to help young people to be productive citizens in a society increasingly mediated by computing technologies. As of the end of 2024, 11 states have computer science graduation standards (Code.org, 2024). And, while requiring that students take at least one CS class improves access to CS education, this study suggests that a single course is unlikely to sufficiently educate students. Specialized CS/IT programs in secondary schools offer opportunities for students to go beyond what they can learn through a single course required for graduation.

The significance of the study reported herein lies in its unique focus on specialized CS education programs rather than on pedagogy or course enrollments. There is a growing number of landscape reports about CS education, and the Expanding Computing Education Pathways (ECEP) Alliance provides access to a number of those state-level reports (ECEP, 2022a) as well as state-level data dashboards (ECEP, 2022b). Those reports and dashboards are all focused on CS access in terms of course-taking; we are not aware of efforts to systematically document beliefs about common elements secondary CS/IT programs should provide. Our other investigation explores the elements secondary CS/IT programs are currently offering. Though not yet complete, that study has already

documented dozens of CS/IT programs for high school students just in Virginia. The findings from this common elements study provide us with a conceptual framework for understanding the educational opportunities provided by the programs identified in the other part of the larger study. Our intent is to use that framework to advance a research agenda aimed at understanding the outcomes of secondary CS/IT programs. The current study makes clear that the framework is necessarily comprehensive with lots of moving parts.

Acknowledgments

This research was funded by a Quest Fund grant through Virginia Commonwealth University.

No part of this manuscript was aided in any way by generative artificial intelligence.

References

Aivaloglou, E., & Hermans, F. (2019). Early Programming Education and Career Orientation: The Effects of Gender, Self-Efficacy, Motivation and Stereotypes. Proceedings of the 50th ACM Technical Symposium on Computer Science Education, 679–685. https://doi.org/10.1145/3287324.3287358

Aritajati, C., Rosson, M. B., Pena, J., Cinque, D., & Segura, A. (2015). A Socio-Cognitive Analysis of Summer Camp Outcomes and Experiences. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education., 581–586.

Armoni, M., & Gal-Ezer, J. (2023). High-school computer science – Its effect on the choice of higher education. *Informatics in Education*, 22(2), 183-206. https://doi.org/10.15388/infedu.2023.14

Bae, C.L., & Lai, M.H. (2020). Opportunities to participate in science learning and student engagement: A mixed methods approach to examining person and context factors. *Journal of Educational Psychology*, 112(6), 1128–1153. http://dx.doi.org/10.1037/edu0000410

Bottia, M.C., Stearns, E., Mickelson, R.A., Moller, S., & Parker, A.D. (2015). The relationships among high school STEM learning experiences and students' intent to declare and declaration of a STEM major in college. *Teachers College Record*, 117(3), 1-46.

Bottia, M.C., Mickelson, R.A., Jamil, C., Giersch, J., Stearns, E., & Moller, S. (2018). The role of high school racial composition and opportunities to learn in students' STEM college participation. *Review of Educational Research* 55, 446–476.

Calabrese Barton, A. & Tan, E. (2009). Funds of knowledge and discourses and hybrid space. *Journal of Research in Science Teaching*, 46(1), 50-73.

Clarke-Midura, J., Sun, C., & Pantic, K. (2020). Making apps: An approach to recruiting youth to computer science. ACM Transactions on Computing Education, 20(4), 1-23. https://doi.org/10.1145/3425710

Code.org, Computer Science Teachers Association, & Expanding Computing Education Pathways Alliance. (2023). 2023 State of computer science education. https://advocacy.code.org/2023_state_of_cs.pdf

Commonwealth of Virginia Board of Education. (2017). Computer science Standards of Learning for Virginia public schools. https://www.doe.virginia.gov/home/showpublisheddocument/9926/638026394162470000

Commonwealth of Virginia Board of Education. (2020). Digital learning integration Standards of Learning for Virginia public schools. https://www.doe.virginia.gov/home/showpublisheddocument/11286/638031727527100000

Corning, A., Becker, J., Broda, M., Hope, S., Lucas, B., Senechal, J., Sions, H. (2021). CodeRVA's Fourth Year Report. Metropolitan Educational Research Consortium.

Corning, A., Broda, M. D., Lucas, B. L., Becker, J. D., & Bae, C. L. (2023). An inclusive school for computer science: Evaluating early impact with propensity score matching. *Studies in Educational Evaluation*, 79, 1-13. https://doi.org/10.1016/j.stueduc.2023.101293

Crowder, A., Dovi, R., & Naff, D. (2020). Integrated STEM education in Virginia: CodeVA Elementary Coaches Academy. In J. Anderson & Y. Li (Eds) *Integrated approaches to STEM education: An international perspective* (pp. 447-467). Springer.

Diamond, I. R., Grant, R. C., Feldman, B. M., Pencharz, P. B., Ling, S. C., Moore, A. M., & Wales, P. W. (2014). Defining consensus: a systematic review recommends methodologic criteria for reporting of Delphi studies. *Journal of clinical epidemiology*, 67(4), 401-409.

Drumm, S., Bradley, C., & Moriarty, F. (2022). 'More of an art than a science?' The development, design, and mechanics of the Delphi Technique. *Research in Social and Administrative Pharmacy,* 18(1), 2230-2236. https://doi.org/10.1016/j.sapharm.2021.06.027

ECEP (Expanding Computing Education Pathways). (2022a). Landscape reports. https://ecepalliance.org/resources/toolkits-guides/landscape-reports/

ECEP (Expanding Computing Education Pathways). (2022b). State data dashboards. https://ecepalliance.org/cs-data/state-data-dashboards/

Eisenhart, M., & Allen, C.D. (2020). Addressing underrepresentation of young women of color in engineering and computing through the lens of sociocultural theory. *Cultural Studies of Science Education*, 15, 793–824.

Elizabeth Casey, J., Gill, P., Pennington, L., & Mireles, S. V. (2017). Lines, roamers, and squares: Oh my! using floor robots to enhance Hispanic students' understanding of programming. *Education and Information Technologies*, 23 (4), 1531–1546. https: //doi.org/10.1007/s10639-017-9677-z

Foth, T., Efstathiou, N., Vanderspank-Wright, B., Ufholz, L. A., Dütthorn, N., Zimansky, M., & Humphrey-Murto, S. (2016). The use of Delphi and Nominal Group Technique in nursing education: A review. *International journal of nursing studies*, 60, 112-120.

Gamoran, A. (2016). Will latest U.S. law lead to successful schools in STEM? *Science,* 353(6305), 1209-1211.

Giani, M. (2022). How attaining industry-recognized credentials in high school shapes education and employment outcomes. Thomas B. Fordham Institute. https://files.eric.ed.gov/fulltext/ED625598.pdf

Gray, D.L., Hope, E.C., & Matthews, J.S. (2018). Black and belonging at school: A case for interpersonal, instructional, and institutional opportunity structures. *Educational Psychologist*, 53(2), 97–113.

Karpiński, Z., Biagi, F., & Di Pietro, G. (2021). Computational thinking, socioeconomic gaps, and policy implications. IEA Compass: *Briefs in Education* No. 12. IEA.

Kwon, K., Ottenbreit-Leftwich, A. T., Brush, T. A., Jeon, M., & Yan, G. (2021). Integration of problem-based learning in elementary computer science education: effects on computational thinking and attitudes. *Educational Technology Research and Development*, 69, 2761-2787.

Lee, A. (2015). Determining the effects of computer science education at the secondary level on STEM major choices in postsecondary institutions in the United States. *Computers & Education*, 88. https://doi.org/10.1016/j.compedu.2015.04.019

Lee, S. J., Francom, G. M., & Nuatomue, J. (2022). Computer science education and K-12 students' computational thinking: A systematic review. *International Journal of Educational Research*, 114, 102008.

Legewie, J., & DiPrete, T.A. (2014). The high school environment and the gender gap in science and engineering. *Sociology of Education*, 87(4), 259–280.

Marshall, S. L., & Grooms, A. A. (2022). Industry's push for computer science education: Is computer science really for all?. *Policy Futures in Education*. https://doi.org/10.1177/14782103211045601

Master, A., Cheryan, S., & Meltzoff, A.N. (2016). Computing whether she belongs: Stereotypes undermine girls' interest and sense of belonging in computer science. *Journal of Educational Psychology*, 108(3), 424–437.

Means, B., Wang, H., Young, V., Peters, V.L., & Lynch, S.J. (2016). STEM-focused high schools as a strategy for enhancing readiness for postsecondary STEM programs. *Journal of Research in Science Teaching*, 53(5), 709–736.

Means, B., Wang, H., Wei, X., Lynch, S., Peters, V., Young, V., & Allen, C. (2017). Expanding STEM opportunities through inclusive STEM-focused high schools. *Science Education*, 101(5), 681–715.

Means, B., Wang, H., Wei, X., Iwatani, E., & Peters, V. (2018). Broadening participation in STEM college majors: Effects of attending a STEM-focused high school. *AERA Open*, 4(4), 1–17.

Muraski, J. M., & Iversen, J. (2022). Growing Computer Science and Information Technology Education in K-12: Industry Demand and Ecosystem Support. *Journal of the Midwest Association for Information Systems* (JMWAIS), 2022(2), 2.

NASEM (National Academies of Sciences, Engineering, and Medicine). (2021). Cultivating interest and competencies in computing: Authentic experiences and design factors. The National Academies Press. https://doi.org/10.17226/25912

National Research Council. (2011). Successful K-12 STEM education: Identifying effective approaches in science, technology, engineering, and mathematics. The National Academies Press. https://doi.org/10.17226/13158.

Perez, T., Cromley, J. G., & Kaplan, A. (2014). The role of identity development, values, and costs in college STEM retention. *Journal of Educational Psychology*, 106(1), 315–329. https://doi.org/10.1037/a0034027

Ryoo, J.J., Margolis, J., Lee, C.H., Sandoval, C.D.M., & Goode, J. (2013). Democratizing computer science knowledge: Transforming the face of computer science through public high school education. *Learning, Media, and Technology*, 38(2), 161–181.

Sabin, M. C., Deloge, R., Smith, A., & DuBow, W. (2017). Summer learning experience for girls in grades 7–9 boosts confidence and interest in computing careers. *Journal of Computing Sciences in Colleges*..

Sáez-López, JM., Sevillano-García, ML. & Vazquez-Cano, E. (2019). The effect of programming on primary school students' mathematical and scientific understanding: educational use of mBot. *Education Tech Research Dev* 67, 1405–1425. https://doi.org/10.1007/s11423-019-09648-5

Shah, N., & Yadav, A. (2023). Racial justice amidst the dangers of computing creep: A dialogue. *TechTrends*, 67, 467-474. https://link.springer.com/article/10.1007/s11528-023-00835-z

Starrett, C., Doman, M., & Garrison, C. (2015). Computational Bead Design: A Pilot Summer Camp in Computer Aided Design and 3D Printing for Middle School Girls. Proceedings of the 46th ACM Technical Symposium on Computer Science Education, 4. https://doi.org/http://dx.doi.org/10.1145/2676723.2677303

Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). A model curriculum for K-12 computer science: Report of the ACM K-12 task force curriculum committee (2nd ed.). Association for Computing Machinery.

Virginia Department of Education. (2022a). Computer science. https://www.doe.virginia.gov/teaching-learning-assessment/instruction/computer-science

Virginia Department of Education. (2022b). CTE Governor's STEM Academies. https://www.doe.virginia.gov/teaching-learning-assessment/k-12-standards-instruction/career-and-technical-education-cte/governor-s-stem-academies

Virginia Department of Education. (2022c). Governor's schools. https://www.doe.virginia.gov/teaching-learning-assessment/specialized-instruction/governor-s-schools

Virginia Department of Education. (2022d). Profile of a Virginia graduate. https://www.doe.virginia.gov/parents-students/for-students/graduation/policy-initiatives/profile-of-a-virginia-graduate

Webb, H. C., & Rosson, M. B. (2011, March). Exploring careers while learning Alice 3D: A summer camp for middle school girls. In Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 377-382).

www.ijcses.org