Development and Validation of Adaptable Rubrics for Programming Assessments: Measuring Computational Competencies

Priyanka Gupta[1], Deepti Mehrotra[2], Sunil Vadera[3]
[1]Amity University Uttar Pradesh, Noida, India
[2]Jaypee Institute of Information Technology, Noida, India
[3]University of Salford, Manchester, England, UK
[1]pjain2004@gmail.com [2]mehdeepti@gmail.com [3]S.Vadera@salford.ac.uk

***Abstract*— Advanced technological innovations and problem-solving across various domains demand proficient programming skills. Thus, developing and measuring computational competencies are critical in this modern digital era. Despite their significance, assessing these competencies is challenging, especially when catering to personalized learning needs. Several assessment methods are available, yet there is a lack of fast, adaptable and validated evaluation tools. This study addresses this gap by developing and validating adaptable rubrics for assessing computational competencies and fostering improved learning outcomes in the computer programming domain. The preliminary rubric was developed per the JAVA programming curricula, educational frameworks, and experts' opinions. A sample of 300 students studying the JAVA programming language were assessed via a multiple-response objective assessment based on the initial rubric. The rubric-based evaluation results were subjected to rigorous statistical analyses, including exploratory and confirmatory factor analyses, correlation, and regression analyses. The findings yielded a four-factor validated adapted rubric structure emphasizing the required computational competencies in the programming domain. Further, moderate associations between rubric scores and Java academic evaluations were revealed, highlighting the predictive validity of rubric-based objective assessments. This research work contributes to the refinement of rubric and assessment practices in programming education, offering insights into the role of adaptable rubrics in promoting student learning and suggesting pedagogical approaches. The findings of this study hold practical implications for educators, curriculum designers, and policymakers. The targeted interventions, adapting rubric-based assessment criteria, improving pedagogical strategies, and enriching curriculum design ensure that the learning experiences of programming learners are tailored to their individual needs and aspirations.**

***Keywords*— Adaptable Rubrics, Computational Competencies, Programming Assessments, Factor Analysis**

***JEET Category*—Research**

I. INTRODUCTION

With the increasing demand for digital literacy and rapid technological advancements, computational skills have emerged as a keystone in modern educational establishments (Tang et al., 2020). Computational skills are incorporated as a set of substantial learning goals to prepare the learners to improve their competencies in various domains (Grover & Pea, 2013; Forero-García et al., 2022; Tan et al., 2023). Therefore, it is essential to identify and assess students' computational skills.

The transformation of teaching-centric to learning-centric environments necessitates adaptable assessment tools to measure the required learner competencies and enhance their proficiencies effectively (Ingaleshwar & Jamadar, 2019), specifically in online learning scenarios (Kukreja et al., 2021). Rubrics have emerged as valuable instruments in educational assessments, providing structured guidelines for evaluating learners' performance across various domains (Jagtap & Powar, 2016; McTighe & Ferrara, 2021). The rubric approach helps both the learner and assessor clearly understand what learning outcomes the learner has achieved and where the gaps persist (Monbec et al., 2020).

However, the dynamic nature of computer programming education demands rubrics that can adapt to the evolving skills of learners and technological advancements (Jeyamala & Abirami, 2020). To thrive in this digital age, worldwide educational institutions employ various rubrics to teach computer programming languages to equip learners with the necessary computational competencies (Ana et al., 2020). Adaptable rubrics represent a cornerstone in contemporary educational assessment practices, specifically in the dynamic field of programming education. They offer a comprehensive approach to assessment, assisting educators with the required flexibility to evaluate diverse student learning dimensions

Deepti Mehrotra
Jaypee Institute of Information Technology, Noida, India
mehdeepti@gmail.com

based on well-defined criteria and learning objectives and tailor feedback to individual learner needs (Fletcher et al., 2012; DeLuca & Klinger, 2010; Molins-Ruano et al., 2014). The adaptability or flexibility in rubrics emerges from learners' diverse needs and skills as per the demands of the technological industry (Pedro et al., 2017; Peña, 2018).

Unlike traditional rubrics focused solely on predetermined criteria, adaptable rubrics allow for customization, ensuring that the assessments accurately reflect the diverse competencies required in the current dynamic job market of the computer programming domain. Adapting the rubric framework involves refining and modifying the existing structure (Lee et al., 2020). However, it transcends mere refinement; it is a strategic endeavor to improve the rubric for better alignment with its intended purpose. This is achieved by revising the rubric's criteria, descriptors, and scoring strategies to improve accuracy and meaning. Also, the adaptation process highlights the specific relevant parts of the rubric. It emphasizes certain skills or competencies substantially impacting desired educational goals (Chen et al., 2019).

This paper embarks on a comprehensive exploration of developing and validating adaptable rubrics tailored specifically for programming assessments. With a keen focus on measuring computational skills and nurturing student learning in this domain, this study aims to establish the validity and reliability of these adaptable rubrics in assessing programming skills and fostering student improvement.

Although rubrics are extensively used for subjective evaluation purposes, their potential in objective assessments is yet to be explored. A simple scoring rubric is typically referred to as a mere key for multiple-choice examinations (Kayarkaya & ÜNALDI, 2020; Dhanya et al., 2022), but if a rubric is developed and utilized for partial grading in multiple response-based assessments (Denny et al., 2019; Das et al., 2021), its potential is extended from a simple key to effectively identifying learner competencies as well (Simkin & Kuechler, 2005). Further, the concept of adaptability in such rubrics provides a flexible approach to assessing and evaluating the diverse skills of learners (Pang et al., 2022). The advent of adaptable rubrics also aligns with the personalized learning scenario, allowing learners to tailor their educational learning paths to their specific learning goals and styles (Qushem et al., 2021). This paper promotes and validates this aspect of rubrics to be utilized in the objective assessment of programming languages that facilitates the mass assessment process and identifies programming aptitudes and valuable feedback mechanisms.

Our research objectives include initially developing a programming rubric considering experts insights and JAVA syllabi, then validating this rubric through rigorous statistical analysis and factor identification via EFA and CFA, leading to the four-factor adapted rubric, and finally exploring the associations between rubric-based assessments and JAVA academic performances via correlation and regression analysis. This paper aims to provide a comprehensive understanding of the role of adaptable rubrics in identifying the desired computational competencies of learners in computer programming and promoting student learning.

This study contributes to the advancements in current practices in programming education by introducing an adaptable and validated rubric-building practice that aligns with the evolving industry demands and provides personalized learning pathways.

It is important to note that the focus of this research work is not to develop any new Java programming assessment or instrument, as numerous standard programming instruments exist already (Relkin et al., 2020; Relkin et al., 2021; Fraillon et al., 2020; Steinhorst et al., 2020). Instead, the study emphasizes the significance of adaptable rubrics in the objective assessment process and highlights their potential to enhance learning outcomes in programming education. The rubric developed in this study serves as a versatile tool that can be utilized to assess any programming language to evaluate diverse computational proficiencies and promote a holistic approach to programming education. The refined rubric may specifically evaluate key computational competencies such as problem-solving, algorithm design, code optimization, debugging, and object-oriented paradigms, ensuring the assessment of learners' programming knowledge and skills on a deeper level.

In the subsequent sections, this research paper delves into the methodology utilized in rubric development and validation, presents the analysis findings, and discusses the research questions and implications of this work for assessment practices in programming education.

II. LITERATURE REVIEW

Pursuing computational competencies among programming learners is essential to enhance their proficiency and readiness for the continuously evolving technological landscape. This section investigates the prior research that guides the foundation of this study, focusing on identifying competencies through rubric-based assessment and their subsequent association with academic performance.

Computational competencies have been widely acknowledged in the literature. Scholars and researchers explore various intellectual processes, such as logical thinking, algorithmic reasoning, and problem-solving skills (Wing, 2006; Tsai et al., 2019). The integration of computational thinking skills in the curriculum of almost every educational scenario has been internationally acknowledged and is increasing rapidly (Nouri et al., 2019; Kjällander et al., 2021; Fagerlund et al., 2021; Dimos et al., 2023). Computational Thinking is a common way of solving problems utilizing programming perspectives (Nouri et al., 2019; Knuth, 2014; Amoako et al., 2013). Researchers used various methodologies to measure computational practices and competencies by assessing coding exercises and projects (Relkin et al., 2021; So et al., 2020; Durak et al., 2019; Papadakis, 2020).

Rubrics, a dynamic assessment tool, assist in measuring computational skills in various domains if properly developed and validated in the proper context. For instance, Ardito et al. (2020) utilized rubric scores to assess students' performances and understand the collaborative development of computational

thinking skills via a six-week robotics program. Similarly, Ung et al. (2021) developed an assessment rubric to evaluate the teaching and learning outcomes of integrated Computational Skills among primary school students. Alves et al. (2020) designed a rubric for a performance-based assessment of programming concepts using an automated static analysis tool. Adler et al. (2022) also created and implemented a computational thinking rubric incorporating key computational components using Bloom's Taxonomy. Ridlo et al. (2022) explored computational skills in implementing a learning model of teaching integrated with computer programming using rubrics and questionnaires. Some researchers also emphasize building smart and intelligent learning environments to support student learning in programming education utilizing computational Thinking approaches and rubrics (Agbo et al., 2019; Sonsilphong et al., 2022).

Further, adapting the rubric framework serves as an educational compass directing focus on the key competencies that significantly affect learning outcomes and teaching pedagogies (Chen et al., 2019; Lee et al., 2020). This research emphasizes an adaptable rubric based on the latent factors obtained from the factor analysis in the programming domain. Then, this refined rubric was validated against the final academic performances of programming learners. The preliminary rubric development and its adaptation based on factor analysis and regression analysis in the programming education scenario should be explored more.

In summary, the research exploring and measuring computational skills in computer programming focuses on utilizing rubrics and their analysis, code analysis methods, survey questionnaires, and computational thinking framework. These research threads build a foundation for this study. However, the gap lies in the amalgamation of rubric-based assessment, factor analysis, and regression analysis to identify the specific computational proficiencies of programming learners, adapting the rubric accordingly, and their implications on academic performance. There is a need to apply statistical analysis techniques to rubric-generated assessment results, leading to more efficient adaptable rubrics. This study will contribute to the ongoing implementation of measuring computational skills and adaptable rubrics, shaping pedagogical strategies and curriculum design to empower learners in this digital era.

III. RESEARCH QUESTIONS

This study investigates the structure of a multiple-response-based JAVA questionnaire prepared per the preliminary developed rubric to extract the significant factors and associations between questionnaire scores and JAVA final exam marks. After data analysis, the following research questions (RQs) will be addressed in the Discussion section.

1. What computational competencies are represented as a result of a thorough analysis of rubric results and objective assessment?

2. What kind of factor structure does the statistical analysis of rubric-based JAVA objective assessment items produce?

3. What associations can be found between the rubric scores and academic programming grades?

IV. METHODS

A. Participants

This research study involved 300 students studying JAVA programming in their undergraduate courses.

B. Rubric Creation

The whole process of rubric development demands careful selection of different criteria across which the competencies have to be assessed, scoring levels/scale across which a particular criterion is measured, and descriptors describing the expectations of an evaluator related to each criterion per scoring level (Gulikers et al., 2021; Jones et al., 2019). The first step of this study was to develop a JAVA-based rubric that can quantify the learner's abilities in diverse programming domains.

To create an initial rubric, learning/course outcomes of various JAVA syllabi of computer programming courses from different Indian Universities were thoroughly studied and analyzed. In addition, the authors extensively reviewed knowledge areas related to computer programming in the ACM/IEEE Computer Science curricula (Mehran et al., 2013). They identified common CS concepts in the course descriptions (Parker et al., 2021). This analysis included a review of the key concepts, skills, and knowledge areas emphasized in these curricula. We identified common themes and patterns across the learning outcomes through this process, which guided the selection of the rubric criteria. Conclusively, almost all learning outcomes focus on strengthening Object-Oriented Concepts, syntax, multithreading, exception handling, event handling, optimization, and application building.

After careful consideration, we categorized all programming learning proficiencies into seven criteria to develop a rubric, ensuring the scope of this rubric is not limited to only the JAVA programming language. However, it can be utilized for any language as any computer programming language demands the learners to master these global skills. These criteria are as follows:

1. Theory and Concepts: This criterion assesses students' comprehension and familiarity with fundamental JAVA programming concepts and ideas.

2. Syntax Knowledge: This criterion evaluates students' familiarity with and accuracy with the syntax and structure of the Java programming language.

3. Conceptual Thinking & Skills: This criterion assesses the student's ability to think conceptually and apply abstract reasoning to programming problems.

4. Critical Thinking: This criterion evaluates students' ability to analyze, evaluate, and make logical decisions regarding programming scenarios and code segments.

5. Logic Building & Thinking: This criterion assesses students' logical reasoning skills and ability to identify structured and efficient algorithms.

6. Optimization Skills & Complexity: This criterion evaluates students' ability to optimize code efficiency and complexity measures.

7. Applications Design: This criterion assesses students' capability and skills to design and develop JAVA applications.

Further, Bloom's taxonomy aligned each criterion with the appropriate cognitive skill. The categorization of all rubric criteria with the revised Bloom's taxonomy levels provides a framework for assessing students' cognitive engagement and depth of understanding in each criterion, enabling a comprehensive evaluation of their JAVA programming skills. The categories assigned to the criteria are as follows:

1. Remembering: Theory and Concepts, Syntax Knowledge
2. Understanding: Conceptual Thinking & Skills
3. Analyzing: Critical Thinking
4. Applying: Logic Building & Thinking
5. Evaluating: Optimization Skills & Complexity
6. Creating: Applications Design

The finalized rubric serves as a valuable instrument for instructors to evaluate students' performance and proficiency in each programming domain aligned with the key cognitive skills. The development process ensures that the essential aspects of JAVA programming are captured and provides a reliable assessment mechanism for measuring and improving student learning outcomes.

C. Measures

In this study, the participants were administered a JAVA quiz of 35 multiple-choice (multiple correct answers) questions. This quiz format of multiple-response objective assessment was chosen for this study because it aligns with evaluating the foundational and specific computational skills in programming education which is the objective of this study. Further, current programming courses generally utilize evaluation methods such as coding assignments, project-based assessments, and automated objective tests to measure the practical competencies of learners, specifically in virtual learning environments. The objective assessments with multiple-response format are able to measure conceptual understanding and basic knowledge effectively and rapidly. In addition, while traditional multiple-choice questions (MCQs) assess learners' knowledge in a binary, black-and-white manner (correct or incorrect), multiple-response quizzes evaluate learners' partial knowledge and understanding. The quiz in this study was prepared according to the rubric developed for assessing programming skills.

Five questions with four possible options were formulated for each rubric criterion in the quiz. These types of MCQs demand competency-based and careful framing of questions with equivalent and perplexing distractors. This is critical for assessing analytical and logical reasoning abilities in programming. Consequently, the proposed method necessitates analyzing and categorizing learners based on their learning aspects and requirements.

Each question was worth four points, with one point assigned to each correctly chosen option. Therefore, each criterion accounted for a total of 20 points. A particular partial marking scheme was employed to determine the correctness of each option. For example, if a specific question had valid answers (a) and (d), the marking scheme was as follows:

If option (a) was selected, 1 point was awarded; otherwise, 0 points were awarded.

If option (b) was selected, it was awarded 0 points; otherwise, it received 1 point.

If option (c) was selected, it was awarded 0 points; otherwise, it received 1 point.

If option (d) was selected, 1 point was awarded; otherwise, 0 points were awarded.

For instance, if a student selected options (b) and (d) for that question, their score would be 2 (0 + 0 + 1 + 1), indicating that they answered the question with 50% correctness.

The partial grades obtained for each rubric criterion reflect the learner's knowledge and analytical abilities relevant to that criterion. The final scores were calculated by adding up all the partial grades. This comprehensive calculation gave learners detailed feedback on their programming abilities, with the scoring scale aligned with the rubric. Since the questions were designed to represent each criterion, the cumulative grade for each criterion allowed students to receive specific feedback on their performance in various programming domains.

D. Validation

According to Messick (1995), construct validity is validating the internal structure of the questionnaire. The internal structure refers to the relationships between test items and the components employed to validate the constructs measured by the instrument (Hogenboom et al., 2021). Item Response Theory (IRT)-Rasch method was used to examine the internal structure of the instrument.

E. Quantitative Analysis

To analyze the questionnaire's factor structure based on the preliminary rubric, Exploratory Factor Analysis (EFA) and Confirmatory Factor Analysis (CFA) were conducted. Correlations and linear regression techniques were used to study the relationships between the latent factors and JAVA final exam marks to examine the convergent validity. The latent factors were finally labeled based on the desired computational competencies for the computer programming outcomes.

V. FINDINGS

A. Descriptive Statistics

A preliminary examination of the multiple-choice exam revealed a wide disparity in the applicants' scores. The total scores ranged from 59 to 120, with an average score of 86.75 (SD = 13.17). This observation indicates that the candidates' performance on the exam was significantly variable.

Fig. 1 depicts the average scores of all individual items of the questionnaire, confirming the observation of score dispersion. The average scores of personal items fluctuated between 0.51

and 3.32, with a mean score of 2.48 and an SD of 0.6. The relatively substantial standard deviation specifies the wide variation in the scores of individual items among the applicants.

The descriptive statistics offer significant insights into the scores' distribution and the applicants' performances on the multiple-choice examination. The broad range of aggregate scores and the disparity in scores for individual items emphasize the diversity in the abilities and knowledge of learners across different competencies assessed in the exam. These findings are the foundation for further analyses and interpretation of the rubric results.

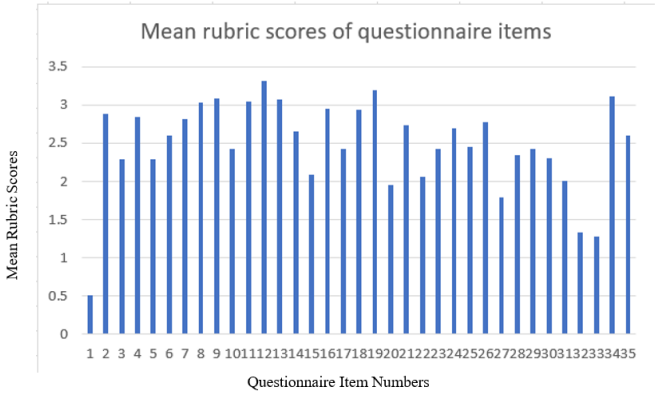

Fig. 1. Average scores distribution of individual items of the questionnaire

B. Quantitative content analysis based on an initial rubric

The initial rubric design consisted of seven factors. The JAVA questionnaire was categorized according to these seven factors. Five questions of each factor were included and assessed using a partial marking scheme. Each criterion's maximum score was 20 since each question had four marks. The means and SDs of the rubric scores of learners are represented in Table I. It can be observed that the criterion of Syntax Knowledge attained a maximum mean value of 16.59 (SD = 2.55), and the criterion of Applications Design possesses a minimum mean value of 10.06 (SD = 2.11). This indicates that learners score quickly in the Syntax Knowledge criterion while they score less in application design.

Further investigation into the results revealed that the students perform very well in syntax, conceptual, and critical thinking-related problems. At the same time, they face difficulties in code optimization and application designing criteria. The criterion Theory and Concepts also showed relatively less attention span due to its theoretical knowledge assessment property. In programming domains, learners are more interested in coding and logical thinking parts rather than memorizing the theory part. Further, not all learners can optimize the code and design parts since they may be in the intermediate stages of their learning. Instead, all learners efficiently understand the syntax and apply the concepts in all programming challenges.

TABLE I

MEANS AND STANDARD DEVIATIONS OF RUBRIC SCORES PER CRITERIA

Criteria	Mean	Standard Deviation
Theory and Concepts	11.77	3.09
Syntax Knowledge	16.59	2.55
Conceptual Thinking & Skills	15.64	2.52
Critical Thinking	14.34	2.51
Logic Building & Thinking	13.75	2.35
Optimization Skills & Complexity	11.95	2
Applications Design	10.06	2.11

C. The factor structure of multiple-choice exam: EFA and CFA

Initially, to assess the suitability of the data for factor analysis, the Kaiser-Meyer-Olkin (KMO) measure of sampling adequacy and Barlett's test of sphericity were performed in this research study. The KMO measure was found to be 0.79, suggesting that the data exhibits a reasonably high degree of shared variance among variables. Further, Barlett's test results indicate $\chi 2 = 1859.567$ with $p < 0.001$ and 595 degrees of freedom. This highly significant p-value provides additional evidence of the uncorrelatedness of variables. Both measures recommend the suitability of factor analysis for our data to explore the underlying latent factors to explain the further observable patterns in the data and whether these factors are comparable with the rubric's criteria.

To determine the suitable number of factors, a scree plot was employed first, as shown in Fig. 2. We also utilized parallel analysis, a statistical technique for extracting the number of appropriate factors by comparing the eigenvalues of actual data with those generated from random data. Parallel analysis scree plots are depicted in Fig. 3.

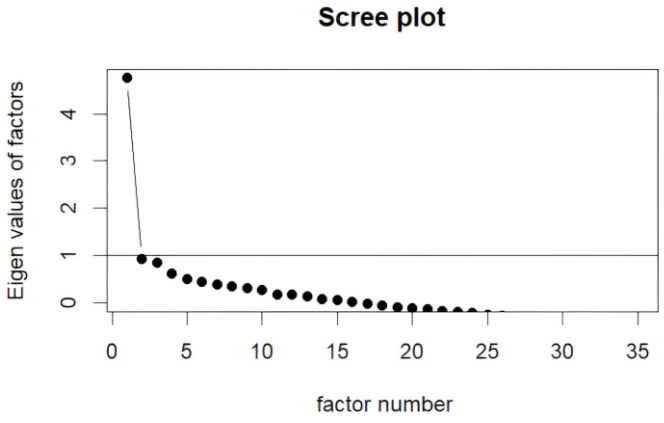

Fig. 2. Scree Plot of data

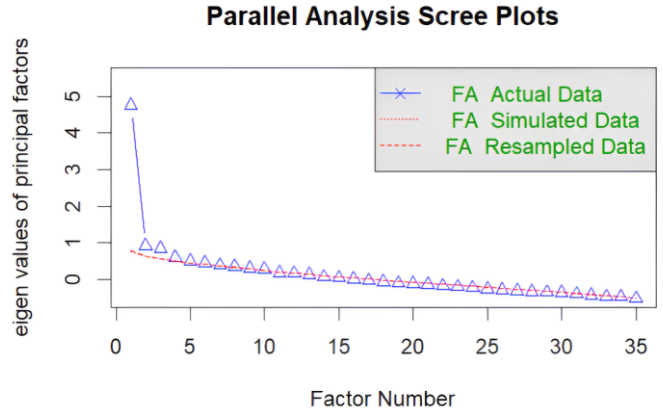

Fig. 3. Parallel Analysis of Data

The scree plot's careful examination revealed an elbow point between 3 and 5. Additionally, the parallel analysis suggested five factors. After thoughtful consideration, domain experts' consultation, and JAVA outcomes analysis, we decided to perform factor analysis considering four factors. Additional criteria for choosing these four factors include their meaningfulness, interpretability, and the extent to which they explained the variance in the data.

EFA with Principal Axis Factoring extraction method was applied with varimax rotation specifying four factors using SPSS 20. The rotated factor loadings are shown in Fig. 4. Using this model, CFA was conducted using SPSS AMOS 26. Initially, the model fit was satisfactory; however, some items needed to have improved factor loadings, falling below 0.30. The model was further modified by removing these items and re-running the analysis. Eleven items were removed, which led to the improved model fit. The model fit indices indicated the chi-square statistic as 361.497 ($p<.001$). This means that the observed covariance matrix matched the model-implied covariance matrix well. In addition, the Comparative Fit Index (CFI) was 0.892, and the Tucker-Lewis Index (TLI) was 0.879. These measures were close to the recommended threshold of 0.90, indicating a good fit. Further, the Root Mean Square Error of Approximation (RMSEA) was 0.040, with a 90% confidence interval (CI) ranging from 0.031 to 0.048, which suggests a close fit of the model. The Root Mean Square Residual (RMR) was 0.082, supporting a model's adequacy.

Fig. 5 presents the four factors' standardized factor loadings in the refined model. As shown in Figure., all the factor loadings and correlations are now significant at $p<.001$. This refined model accurately captures the underlying structure of the variables, providing insights into the items' relationships and the factors they represent. These four factors were termed Programming Abilities (encompassing Theory and Concepts, Syntax Knowledge, and Conceptual Thinking & Skills), Analytical Skills (encompassing Critical Thinking and Logic Building & Thinking), Optimization Skills, and Applications Design. The four factors are the required Computational Competencies for programming learners.

Rotated Factor Matrix[a]

	Factor			
	1	2	3	4
Q1	.031	.130	.062	.376
Q2	.292	.359	-.068	.049
Q3	-.020	.114	.458	-.017
Q4	.303	.257	.020	.073
Q5	.132	.293	.095	.366
Q6	.264	.342	.106	.209
Q7	.565	.212	.139	.078
Q8	.382	.314	.125	.005
Q9	.493	.036	.029	.060
Q10	.259	.167	.279	.081
Q11	.371	-.041	.324	.289
Q12	.422	.183	.080	.009
Q13	.434	.237	.278	.066
Q14	.025	.348	.182	.148
Q15	.143	.201	-.193	.192
Q16	.260	.193	.177	.199
Q17	.405	.054	.102	.116
Q18	.182	.285	.252	.064
Q19	.172	.332	.083	.137
Q20	-.111	-.152	.309	.321
Q21	.091	-.057	.343	.092
Q22	.181	.136	-.077	-.095
Q23	.277	.476	-.064	.208
Q24	.178	.056	.046	.351
Q25	.308	.446	-.015	.109
Q26	.126	.455	-.015	-.084
Q27	-.186	.177	.124	.072
Q28	.030	-.211	-.161	.205
Q29	.195	.111	.397	-.213
Q30	-.174	.018	-.068	.307
Q31	.061	.217	.045	-.016
Q32	-.510	-.161	-.076	.072
Q33	-.461	-.254	-.126	.111
Q34	.604	.206	-.003	.059
Q35	.228	.111	.296	.046

Extraction Method: Principal Axis Factoring.
Rotation Method: Varimax with Kaiser Normalization.
a. Rotation converged in 13 iterations.

Fig. 4. Factor Loadings of EFA

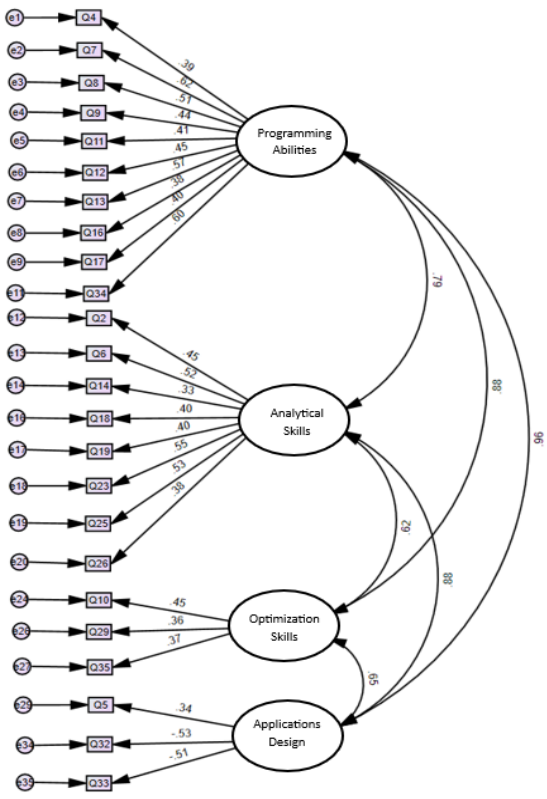

Fig. 5. Confirmatory Factor Model of JAVA Multiple-Choice Questionnaire

The factor labelled Programming Abilities comprised ten items. These items encompassed questions testing the learner's proficiency in programming syntax, theory, and fundamental concepts essential for writing functional codes. Analytical Skills comprised eight items assessing the learner's application capabilities of programming concepts logically and critically. Optimization Skills and Applications Design factors included three items, each assessing the programmer's aptitude for efficient code optimization and real-world programming challenges. Educators can develop targeted instructional strategies and specific assessments by aligning these factors with clearly stated learning objectives to promote the comprehensive skill development of learners.

Further, the descriptive statistics and inter-factor correlations are presented in Table II. As depicted in Table II, learners achieved relatively high scores in the Programming Abilities factor due to its foundational concepts testing items only, and mostly all programming learners are profound in the basic syntax and concepts. Alternatively, learners scored less in the Optimization Skills and Applications Design factors due to their advanced and optimization testing items. Not all programming learners must achieve a high learning threshold in these areas in the early stages. Moreover, the correlations among the latent constructs, as shown in Table II, indicate the relationships between the identified factors.

TABLE II
MEANS AND STANDARD DEVIATIONS OF THE SUM VARIABLES ON THE ITEMS OF THE FACTORS AND CORRELATIONS BETWEEN THE LATENT CONSTRUCTS

Criteria	Mean	SD	PA	AS	OS	AD
Programming Abilities (PA)	29.68	6.364	-			
Analytical Skills (AS)	21.94	5.167	0.79	-		
Optimization Skills (OS)	7.44	2.742	0.88	0.62	-	
Applications Design (AD)	4.9	2.71	0.96	0.88	0.65	-

D. Internal Structural Aspect: Item Analysis

Table III presents the fit statistics for the latent factors extracted in the factor analysis step. As shown in Table III, all items exhibit infit and outfit MNSQ values within the acceptable range of 0.60-1.40 (Wright & Linacre, 1994). This means that all the items will discriminate all the factors.

TABLE III
ITEM FIT STATISTICS FOR THE QUESTIONNAIRE INSTRUMENT

Dimension	Item No.	Measure	Infit MNSQ	Outfit MNSQ	Cronbach's Alpha if Item Deleted
Programming Abilities	Q4	-0.97	0.97	0.92	0.74
	Q7	-0.38	1.11	1.14	0.71
	Q8	-0.44	1.14	1.13	0.73
	Q9	-0.33	0.98	1	0.73
	Q11	-0.46	1.05	0.94	0.73
	Q12	-0.79	1.03	0.84	0.73
	Q13	0.07	0.99	0.88	0.72
	Q16	-1.30	0.82	0.76	0.74
	Q17	-0.26	0.98	0.92	0.73
	Q34	-0.35	1.04	1.01	0.71
Analytical Skills	Q2	-0.05	0.92	0.98	0.64
	Q6	-0.27	1.04	1.03	0.63
	Q14	-0.31	1.02	0.93	0.65
	Q18	-0.72	1.04	1.00	0.65
	Q19	-0.64	1.00	1.05	0.64
	Q23	0.52	1.06	1.06	0.62
	Q25	-0.97	0.99	0.88	0.63
	Q26	-0.49	1.21	1.33	0.64
Optimization Skills	Q10	0.37	1	1	0.68
	Q29	-0.70	1	1	0.62
	Q35	0.62	1	1	0.66
Applications Design	Q5	1.07	1	1	0.69
	Q32	-0.19	1	1	0.66
	Q33	0.06	1	1	0.65

E. Relations to the JAVA Final Exam Marks of Learners

A correlation and regression analysis has been conducted using IBM SPSS 20 to examine the relationship between the latent factors and JAVA final exam marks. The findings are summarized in Table IV.

According to the results shown in Table IV, the correlations between the factors and final exam marks are almost moderate. The factors of Programming Abilities, Analytical Skills, and Optimization Skills exhibited a significant positive correlation of 0.353 ($p<0.001$), 0.365 ($p<0.001$), and 0.210 ($p<0.001$) respectively with the final exam marks. This indicates that the learners who possess these skills in programming tend to achieve high scores in the final programming examination. The factor – Applications Design exhibits a negligible correlation of 0.010 with the final exam marks. However, this correlation was not statistically significant ($p = 0.862$), revealing no substantial relationship between this factor and final marks. This result is justified since the final programming examination focuses more on assessing the essential and conceptual competencies, and Applications Design factor focuses on higher-level programming constructs and application building.

A deeper understanding of the association between the JAVA final exam marks and the factors is carried out by Linear Regression analysis, as shown in Table IV. The aim is to predict the final exam marks based on the four factors. As shown in Table IV, Programming Abilities and Analytical Skills exhibit a beta coefficient of 0.247 ($p = 0.001$) and 0.232 ($p < 0.001$), respectively, which indicates that these two factors had a statistically significant positive impact on the predicted final exam marks. Applications Design also expected the final exam marks, but with less impact (beta coefficient = 0.126 with $p < 0.05$). However, Optimization Skills yielded a beta coefficient of 0.043 with $p = 0.467$, which was not statistically significant. This indicates that this factor did not significantly impact the predicted final exam marks. This model explained 18.2% of the variance in the predicted final exam marks ($R^2 = 0.182$).

To further understand the role of these factors and the impact

of adaptable rubrics on learning outcomes, this study could further extend towards longitudinal analysis and deeper exploration of relationship between final marks and refined rubric criteria. It is essential to reveal how these factors influence learners' advanced programming skills and professional competencies over a period of time. In addition, including project-based evaluations in modern assessment practices may offer a profound comprehensive understanding of the role of higher-order skills such as Applications Design and Code Optimization. These adjustments help enhance the applicability and efficiency of adaptable rubrics in the programming domain.

TABLE IV
CORRELATIONS AND REGRESSION ANALYSIS BETWEEN JAVA FINAL EXAM MARKS AND SCORES OF THE LATENT FACTORS

	Correlations	**Regression Analysis**		
		β (S.E.)	**p**	**R^2**
Programming Abilities	0.353 (p<0.001)	0.247	0.001	
Analytical Skills	0.365 (p<0.001)	0.232	0.000	
Optimization Skills	0.210 (p<0.001)	0.043	0.467	0.182
Applications Design	0.010	0.126	0.025	

F. Rubric Adaptation

A comprehensive JAVA rubric was formulated in the initial phases of this study comprising seven distinct factors - Theory and Concepts, Syntax Knowledge, Conceptual Thinking & Skills, Critical Thinking, Logic Building & Thinking, Optimization Skills & Complexity, and Applications Design. This rubric was built to align the course outcomes of JAVA with experts' opinions. Subsequently, this rubric was utilized to assess a JAVA multiple-response-based questionnaire of 35 items using a partial grading scheme. This assessment process generated initial rubric scores encompassing various programming skills and conceptual proficiencies.

The partial grading scores of students were then subjected to factor analysis to identify further the potential latent factors present in the data of learners' responses. The factor analysis resulted in four significant factors: the refinement of the initial seven-factor rubric. These four factors (Programming Abilities, Analytical Skills, Optimization Skills, and Applications Design) were the required computational competencies for any comprehensive programming curriculum. This transformation from an extensive rubric to a comparatively concise set of factors, driven by the statistical examination of learners' responses, is the process of rubric adaptation, which captures the core competencies that influence learners' performances in the computer programming domain.

This work highlighted rubric development's dynamic nature through the iterative rubric adaptation and statistical analysis process. The findings emphasized the potential of rubrics not only as a mere assessment tool but also to assist in refining and adapting itself and pedagogical approaches. Using various statistical analysis tools and techniques, rubrics may evolve to precisely capture the desired core competencies in a particular domain, thereby effectively enhancing the quality and relevance of educational assessments. Currently, existing programming assessment tools often emphasize final outputs or specific coding tasks, whereas the adapted rubric in this work assesses a broader range of programming competencies and provides meaningful levelled feedback assessing the knowledge attainment by the student.

VI. DISCUSSION

The first research question (RQ1) inquired about the competencies in programming emerging from a comprehensive analysis of rubric results and objective assessments. Initially, a basic JAVA rubric possessing seven factors (Theory and Concepts, Syntax Knowledge, Conceptual Thinking & Skills, Critical Thinking, Logic Building & Thinking, Optimization Skills & Complexity, and Applications Design) has been designed concerning the course outcomes and experts' insights to address this RQ1. Further analysis of this rubric-based assessment provided profound insights into learners' computational strengths and challenges. The descriptive statistics of rubric scores revealed that Syntax Knowledge was easier to gain than all the other criteria. Learners also performed well in conceptual, critical, and logical thinking criteria. They faced challenges in application design and Code Optimization. Learners' varying programming stages influenced the disparity in performance across all rubric criteria. The general competency in understanding syntax and applying core concepts was uniform across the cohort. These observations divulged the dynamic nature of programming education, shaping the computational competencies of learners based on their exposure and proficiency levels in various programming aspects. This initial exploration results in a broader set of computational competencies, setting the stage for subsequent analysis into the learners' performances and their impact on outcomes that finally influence programming achievements.

The second research question (RQ2) focussed on the factor structure results of the JAVA assessment items. The statistical analysis of JAVA questionnaire items offered interesting observations into their relationships. EFA and CFA models identified four significant factors: Programming Abilities, Analytical Skills, Optimization Skills, and Applications Design. These factors encapsulate the desired computational competencies for programming learners. The descriptive statistics and inter-factor correlations yielded similar results to the preliminary seven-factor rubric scores. Learners achieved high scores in Programming Abilities and Analytical Skills owing to their foundational and concept application-based items. However, Optimization Skills and Applications Design yielded lower scores, reflecting the complexity of advanced and code optimization items. These observations may be explained by the learner's priorities in different stages of their learning process. Not all learners quickly achieve code optimization and application designing techniques, which may be necessary to develop high-end applications. Their primary focus is logical

and critical thinking in programming concepts, which are necessary for their strong foundation-building in the programming path.

This work combines the classical test theory and Item-Response theory (CFA and Rasch Analysis) to enhance the validity structure of the instrument. Rasch analysis further confirmed the best-fitting model obtained from CFA. A regression model was built based on this validated instrument.

The third research question (RQ3) queried the associations between the modified rubric scores and JAVA programming examination marks. The correlation between the factors and final exam marks was moderate. Programming Abilities, Analytical Skills, and Optimization Skills exhibited significant positive correlations with the final marks, whereas Applications Design manifested an insignificant negligible correlation with the final marks. This observation validates the previous anticipation of prioritizing foundational concepts over higher-order ones. Linear regression analysis predicted the JAVA final exam marks based on the four latent factors. The results anticipated that Programming Abilities and Analytical Skills significantly influenced the predicted final marks. Applications Design also predicted the final exam marks but with less significant impact, and Optimization Skills were not statistically significant in predicting final exam marks. All the factors predicted 18.2% of the variance in JAVA final exam marks.

Finally, the rubric was transformed from a seven-factor preliminary structure to a four-factor concise structure representing programming learners' prominent computational competencies. This adaptive rubric may be used to predict the programming areas where learners need to improvise in the educational assessments. This iterative rubric formation assists educators in developing a robust rubric structure measuring the required competencies in outcome-based education. The distinctiveness of this rubric lies in its adaptability which ensures its relevance across various programming languages, other educational scenarios, and different skill levels, thereby making it a versatile tool for promoting and evaluating computational competencies in diverse contexts.

CONCLUSION

Technology is evolving rapidly, demanding that education systems worldwide adapt their methodologies and approaches to meet new challenges. Extracting computational skills with valid, comprehensive rubric-based assessment strategies encourages progressive and outcome-based learning.

One of the aims of this study is to investigate the computational competencies and skills demonstrated by the learners through a comprehensive analysis of adaptable rubric results and objective assessment in computer programming. These skills can be identified by inspecting the patterns, trends, and performance indicators present in the data.

This study initially designed an essential JAVA rubric, considering course outcomes and experts' opinions. Based on this rubric, an objective assessment has been prepared. Subsequent factor analysis of the assessment scores revealed four significant factors: Programming Abilities, Analytical Skills, Optimization Skills, and Applications Design. These factors encapsulate the overall programming competencies and reflect each level of learning, from grasping foundational concepts to unravelling complex coding challenges. Educators can design their teaching strategies and pedagogies by focusing on targeted and relevant programming skills of learners as per their learning capabilities. This adapted and refined rubric structure aligns assessment practices with learning objectives, assisting educators to identify and address skill gaps efficiently. It also reflects the interconnectedness of learners' specific competencies, leading to personalized learning paths.

The correlations between the identified factors and JAVA final exam marks unveiled that programming abilities and analytical skills are pivotal in shaping learners' programming prowess. Regression analysis also reveals that these factors emerged as robust predictors of final exam performance, emphasizing their role in determining academic success.

This study also highlights the importance of objective assessment techniques in programming education. Educators can gain deeper insights into students' problem-solving abilities, critical thinking skills, and overall programming competence by utilizing rubrics in objective exams, primarily via a partial grading mechanism. This comprehensive evaluation enhances the effectiveness of objective assessments in programming education and promotes the development of well-rounded programming professionals.

This research focuses on bridging the gap between theoretical frameworks and practical implementation by developing, validating and applying adaptable rubrics, thereby contributing to solving the assessment challenges in programming education and computational skills. Educators may utilize the rubric to design formative and outcome-based assessments resulting in immediate feedback on key computational abilities of the learner. The refined rubric may be integrated into the learning management systems for automated and targeted evaluations of programming assessments. Further, policymakers may adopt the strategy to build refined and adapted rubric as a standard procedure for effective evaluations of programming proficiencies across institutions, ensuring uniformity and orientation with the industry demands.

The future scope of this research study includes the longitudinal analysis of the evolvement of competencies over time and their impact on learners' long-term success in the programming domain using adaptable rubrics. Further, experimenting with the rubric refinement process across other programming languages, such as Python, C++, Javascript, and other diverse contexts, helps to reveal how the computational competencies of learners adapt to different educational scenarios. For instance, the Programming Abilities criterion can be utilized for language-specific features such as the indentation rules of Python or the Applications Design criterion can be expanded to assess web application developments in JavaScript. Furthermore, the adapted rubric framework could be utilized to evaluate computational thinking patterns in non-programming contexts as well such as data analytics, system

design, or software testing by focusing on logic, abstraction details, and other suitable criteria. Additionally, sharing the refined rubric with the learners enable them to monitor their learning levels and skill gaps accordingly which leads to further enhancement in their learning outcomes.

REFERENCES

Adler, R. F., Hibdon, J., Kim, H., Mayle, S., Pines, B., & Srinivas, S. (2022). Assessing computational Thinking across a STEM curriculum for pre-service teachers. *Education and Information Technologies,* 1-23. DOI:10.1007/s10639-022-11508-4

Agbo, F. J., Oyelere, S. S., Suhonen, J., & Adewumi, S. (2019, November). A systematic review of computational thinking approach for programming education in higher education institutions. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research* (pp. 1-10). DOI:10.1145/3364510.3364521

Alves, N. D. C., von Wangenheim, C. G., Hauck, J. C. R., & Borgatto, A. F. (2020, February). A large-scale evaluation of a rubric for the automatic assessment of algorithms and programming concepts. In *Proceedings of the 51st ACM technical symposium on computer science education* (pp. 556-562). DOI:10.1145/3328778.3366840

Amoako, P. Y. O., Sarpong, K. A. M., Arthur, J. K., & Adjetey, C. (2013). Performance of students in computer programming: Background, field of study and learning approach paradigm. *International Journal of Computer Applications, 77*(12). DOI:10.5120/13446-1221

Ana, A., Yulia, C. I. C. A., Jubaedah, Y. O. Y. O. H., Muktiarni, M., Dwiyanti, V. I. N. A., & Maosul, A. S. E. P. (2020). Assessment of student competence using electronic rubric. *Journal of Engineering Science and Technology, 15*(6), 3559-3570.

Ardito, G., Czerkawski, B., & Scollins, L. (2020). Learning computational Thinking together: Effects of gender differences in collaborative middle school robotics program. *TechTrends, 64*(3), 373-387. DOI:10.1007/s11528-019-00461-8

Chen, A. M., Cailor, S., Franz, T., Fox, N., Thornton, P., & Norfolk, M. (2019). Development and validation of the self-care counseling rubric (SCCR) to assess student self-care counseling skills. *Currents in Pharmacy Teaching and Learning, 11*(8), 774-781. DOI: 10.1016/j.cptl.2019.04.006

Das, B., Majumder, M., Phadikar, S., & Sekh, A. A. (2021). Multiple-choice question generation with auto-generated distractors for computer-assisted educational assessment. *Multimedia Tools and Applications, 80*(21-23), 31907-31925. DOI:10.1007/s11042-021-11222-2

DeLuca, C., & Klinger, D. A. (2010). Assessment literacy development: Identifying gaps in teacher candidates' learning. *Assessment in Education: Principles, Policy & Practice, 17*(4), 419-438. DOI:10.1080/0969594X.2010.516643

Denny, P., Manoharan, S., Speidel, U., Russello, G., & Chang, A. (2019, February). On the fairness of multiple-variant multiple-choice examinations. In *Proceedings of the 50th ACM technical symposium on computer science education* (pp. 462-468).

Dhanya, N. M., Balaji, R. K., & Akash, S. (2022, February). Aixam - AI assisted Online MCQ Generation Platform using Google T5 and Sense2Vec. In *2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)* (pp. 38-44). IEEE. DOI:10.1109/ICAIS53314.2022.9743027

Dimos, I., Velaora, C., Louvaris, K., Kakarountas, A., & Antonarakou, A. (2023). How a Rubric Score Application Empowers Teachers' Attitudes over Computational Thinking Leverage. *Information, 14*(2), 118. https://doi.org/10.3390/info14020118

Durak, H. Y., Yilmaz, F. G. K., & Yilmaz, R. (2019). Computational Thinking, Programming Self-Efficacy, Problem Solving and Experiences in the Programming Process Conducted with Robotic Activities. *Contemporary Educational Technology, 10*(2), 173-197. DOI:10.30935/cet.554493

Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2021). Computational Thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education, 29*(1), 12-28. DOI:10.1002/cae.22255

Fletcher, R. B., Meyer, L. H., Anderson, H., Johnston, P., & Rees, M. (2012). Faculty and Students Conceptions of Assessment in Higher Education. *Higher Education, 64*(1), 119-133. DOI:10.1007/s10734-011-9484-1

Forero-García, E., Castañeda, D. P., Corredor-Cely, J., & Paternina, J. L. (2022). Energetic Competencies in Electronic Engineering Education: A Sustainable Social Commitment. Journal of Engineering Education Transformations, 36(2), 55–66. Scopus. https://doi.org/10.16920/jeet/2022/v36i2/22154

Fraillon, J., Ainley, J., Schulz, W., Friedman, T., & Duckworth, D. (2020). Preparing for life in a digital world: IEA international computer and information literacy study 2018 international report (p. 297). *Springer Nature*.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher, 42*(1), 38-43. DOI:10.3102/0013189X12463051

Gulikers, J., Brinkman, D., & Runhaar, P. (2021). Using a rubric to grasp intercultural competence development in vocational education. *Journal of Vocational Education & Training, 73*(1), 47-70. DOI:10.1080/13636820.2019.1688854

Hogenboom, S. A. M., Hermans, F. F. J., & Van der Maas, H. L. J. (2021). Computerized adaptive assessment of understanding of programming concepts in primary school children. *Computer Science Education*, 1–30. doi:10.1080/08993408.2021.1914461

Ingaleshwar, S. S., & Jamadar, A. M. (2019). Enhancing faculty competencies through engineering exploration & design project course. Journal of Engineering Education Transformations, 33(Special Issue 1), 55–59. Scopus. https://doi.org/10.16920/jeet/2019/v33i1/149018

Jagtap, A.M., & Powar, A.A. (2016). Evaluation of Culminating B.Tech Project (CBP) Using Assessment Rubrics and Mapping. *Journal of Engineering Education Transformations, 30*. DOI: 10.16920/jeet/2016/v0i0/111637

Jeyamala, C., & Abirami, A.M. (2020). Enhancing Student Learning and Engagement in Freshman Course on Problem Solving Using Computers. *Journal of Engineering Education Transformations, 33*, 192-200. DOI: 10.16920/jeet/2020/v33i0/150144

Jones, C. E., Millar, T. J., & Chuck, J. A. (2019). Development of a Rubric for Identifying and Characterizing Work-Integrated Learning Activities in Science Undergraduate Courses. *International Journal of Work-Integrated Learning, 20*(4), 351-364.

Kayarkaya, B., & ÜNALDI, A. (2020). What You might not be Assessing through a Multiple Choice Test Task. *International Journal of Assessment Tools in Education, 7*(1), 98-113.

Kjällander, S., Mannila, L., Åkerfeldt, A., & Heintz, F. (2021). Elementary Students' First Approach to Computational Thinking and Programming. *Education Sciences, 11*(2), 80. DOI:10.3390/educsci11020080

Knuth, D. E. (2014). The Art of Computer Programming. volume 2: Seminumerical Algorithms. *Addison-Wesley Professional*.

Kukreja, V., Kaur, A., & Aggarwal, A. (2021). What factors impact online education? A factor analysis approach. *Journal of Engineering Education Transformations, 34*(Special Issue), 365–374. Scopus. https://doi.org/10.16920/jeet/2021/v34i0/157180

Lee, J. E., Recker, M., & Yuan, M. (2020). The Validity and Instructional Value of a Rubric for Evaluating Online Course Quality: An Empirical Study. *Online Learning, 24*(1), 245-263.

McTighe, J., & Ferrara, S. (2021). Assessing student learning by design: Principles and practices for teachers and school leaders. *Teachers College Press*.

Mehran Sahami, Andrea Danyluk, Sally Fincher, Kathleen Fisher, Dan Drossman, Elizabeth Hawthorne, Randy Katz, Rich LeBlanc, Dave Reed, Steve Roach, Ernesto Cuadros-Vargas, Ronald Dodge, Robert France, Amruth Kumar, Brian Robinson, Remzi Seker, and Alfred Thompson. 2013. *Computer Science Curricula 2013:Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* ACM Press, New York, NY. https://doi.org/10.1145/2534860

Messick, S. (1995). Validity of psychological assessment: Validation of inferences from persons' responses and performances as scientifc inquiry into score meaning. *American Psychologist, 50*(9), 741–749

Molins-Ruano, P., González-Sacristán, C., Díez, F., Rodriguez, P., & Sacha, G. M. (2014). Adaptive Model for Computer-Assisted Assessment in Programming Skills. *arXiv preprint arXiv:1403.1465*. https://doi.org/10.48550/arXiv.1403.1465

Monbec, L., Tilakaratna, N., Brooke, M., Lau, S. T., Chan, Y. S., & Wu, V. (2020). Designing a rubric for reflection in nursing: a Legitimation Code Theory and systemic functional linguistics-informed framework. *Assessment & Evaluation in Higher Education*, 1–16. doi:10.1080/02602938.2020.1855414

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry, 11*(1), 1-17. https://doi.org/10.1080/20004508.2019.1627844

Pang, T. Y., Kootsookos, A., Fox, K., & Pirogova, E. (2022). Does an Assessment Rubric Provide a Better Learning Experience for Undergraduates in Developing Transferable Skills?. *Journal of University Teaching and Learning Practice, 19*(3), 3.

Papadakis, S. (2020). Evaluating a game-development approach to teach introductory programming concepts in secondary education. International *Journal of Technology Enhanced Learning, 12*(2), 127-145. https://doi.org/10.1504/IJTEL.2020.106282

Parker, M. C., Guzdial, M., & Tew, A. E. (2021, August). Uses, revisions, and the future of validated assessments in computing education: A case study of the FCS1 and SCS1. In *Proceedings of the 17th ACM Conference on International Computing Education Research* (pp. 60-68).

Pedro Company, Contero, M., Otey, J., Camba, J. D., Agost, M. J., & Pérez-López, D. (2017). Web-Based System for Adaptable Rubrics: Case Study on CAD Assessment. *Educational Technology & Society, 20*(3), 24-41.

Peña, B. (2018). Defining quantitative and automated rubrics from the assessment activities scores. In *EDULEARN18 Proceedings of 10th International Conference on Education and New Learning Technologies* (pp. 2054-2058). IATED. DOI:10.21125/edulearn.2018.0574

Qushem, U. B., Christopoulos, A., Oyelere, S. S., Ogata, H., & Laakso, M. J. (2021). Multimodal technologies in precision education: Providing new opportunities or adding more challenges?. *Education sciences, 11*(7), 338.

Relkin, E., de Ruiter, L., & Bers, M. U. (2020). TechCheck: Development and validation of an unplugged assessment of computational thinking in early childhood education. *Journal of Science Education and Technology, 29*(4), 482-498.

Relkin, E., de Ruiter, L. E., & Bers, M. U. (2021). Learning to code and the acquisition of computational thinking by young children. *Computers & Education, 169*, 104222. https://doi.org/10.1016/j.compedu.2021.104222

Ridlo, Z. R., Supeno, S., Wahyuni, S. ., Mahardika, I. K. ., Wicaksono, I. ., & Ulfa, E. M. . (2022). The Analysis of Implementation Project-Based Learning Model of Teaching Integrated with Computer Programming in Improving Computational Thinking Skills in a Classical Mechanics Course. *Jurnal Penelitian Pendidikan IPA, 8*(4), 1734–1742. https://doi.org/10.29303/jppipa.v8i4.1789

Simkin, M. G., & Kuechler, W. L. (2005). Multiple-Choice Tests and Student Understanding: What is the Connection?. *Decision Sciences Journal of Innovative Education, 3*(1), 73-98. DOI:10.1111/j.1540-4609.2005.00053.x

So, H. J., Jong, M. S. Y., & Liu, C. C. (2020). Computational Thinking Education in the Asian Pacific Region. *The Asia-Pacific Education Researcher, 29*, 1-8. https://doi.org/10.1007/s40299-019-00494-w

Sonsilphong, S., Sonsilphong, A., Hormdee, D., & Sae-Joo, P. (2022, July). A Design and Development of Internet of Things (IoT) System and Learning Activity to Promote Computational Thinking. In *2022 7th International STEM Education Conference (iSTEM-Ed)* (pp. 1-4). IEEE.

Steinhorst, P., Petersen, A., & Vahrenhold, J. (2020, August). Revisiting self-efficacy in introductory programming. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (pp. 158-169).

Tan, B., Jin, H. Y., & Cutumisu, M. (2023). The applications of machine learning in computational thinking assessments: a scoping review. *Computer Science Education*, 1-29. https://doi.org/10.1080/08993408.2023.2245687

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education, 148*, 103798. https://doi.org/10.1016/j.compedu.2019.103798

Tsai, M. J., Wang, C. Y., & Hsu, P. F. (2019). Developing the Computer Programming Self-Efficacy Scale for Computer Literacy Education. *Journal of Educational Computing Research, 56*(8), 1345-1360. https://doi.org/10.1177/0735633117746747

Ung, L. L., Labadin, J., & Nizam, S. (2021). Development of a rubric to assess computational thinking skills among primary school students in Malaysia. *ESTEEM Academic Journal, 17*, 11-22.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wright, B. D., & Linacre, J. M. (1994). Reasonable mean square ft values. *Rasch Measurement Transactions, 8*(3), 370.