Seven Components of Computational Thinking: Assessing the Quality of Dr. Scratch Metrics Using 230K Scratch Projects

Gal Bubnič
gb78843@student.uni-lj.si
Faculty of Natural Sciences and Engineering,
University of Ljubljana
Aškerčeva cesta 12
SI-1000 Ljubljana, Slovenia

Tomaž Kosar
tomaz.kosar@um.si
Faculty of Electrical Engineering and Computer Science,
University of Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia

Bostjan Bubnic
bostjan.bubnic@student.um.si
Faculty of Electrical Engineering and Computer Science,
University of Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia

ABSTRACT

Computational thinking has extended beyond traditional computing education recently and is becoming a broad educational movement, focused on teaching and learning critical problem-solving skills across various disciplines. Originating from computer science and programming, the most common learning method still involves educational programming languages like Scratch. Dr. Scratch is a tool designed to assess Scratch projects based on seven components of computational thinking, including abstraction, parallelism, logic, synchronization, flow control, user interactivity, and data representation. This study examines the quality of Dr. Scratch measurement scale. The proposed model considers computational thinking as a latent variable with seven indicators. According to the results of confirmatory factor analysis, five of the computational thinking components were measured satisfactorily, while two were below the accepted level. Based on the results, we recommend conducting an exploratory factor analysis for the potential scale refinement.

KEYWORDS

Computational thinking, Dr. Scratch, Confirmatory factor analysis

1 INTRODUCTION

Although the phrase computational thinking was introduced as a computer science concept in the early 1980s, the concept was popularized by Janette Wing in 2006 [18]. Wing described it as the ability to solve problems, design systems, and understand human behavior by leveraging fundamental computer science concepts. Recently, computational thinking has extended beyond traditional computing education into various interdisciplinary fields. It has been integrated into K-12 education, fostering problem-solving skills from an early age [10]. In addition, disciplines such as biology, physics, and social sciences are adopting computational thinking principles to tackle complex problems, analyze data, and model systems. This broadening of scope highlights the versatility and importance of computational thinking as one of the fundamental skills for the 21st century [12].

Despite its widespread adoption, there is still no consensus on the precise definition of computational thinking. Moreover, there is no consensus concerning its definitive or necessary components [5]. However, several studies have investigated the components that form its foundation. Based on the literature review: 1) Kalelioglu et al. [9] advocated that the most important components are abstraction, problem-solving, algorithmic thinking, and pattern recognition; 2) Bubnic and Kosar [5] identified abstraction and algorithms as relevant, domain independent components; 3) Lyon and J. Magana [11] argued that abstraction is the most definitional term.

Since computational thinking originates from computer science and programming, it is commonly learned and assessed today through educational programming languages like Scratch. Scratch was created by the Lifelong Kindergarten Group at the MIT Media Laboratory to provide a new environment for beginner programmers. Scratch programs are created using scripts assembled by dragging and dropping blocks, which symbolize various programming elements, such as expressions, conditions, statements, and variables. This approach helps avoiding common syntax errors, which often frustrate students. The programming environment also features interactive, 2-dimensional animations called sprites, which move on the screen according to user input or script commands. In addition, audio and video clips from webcams can be incorporated into Scratch projects.

Following the creation of Scratch, its user base grew rapidly. Due to its rapid expansion, the need for evaluation tools became more evident. In response, researchers developed several tools aimed at evaluating Scratch projects, such as Dr. Scratch [13] and Hairball [2]. Dr. Scratch is an on-line tool, which automatically assesses Scratch projects based on seven components of computational thinking, including abstraction, parallelism, logic, synchronization, flow control, user interactivity, and data representation.

The objective of this study was to examine Dr. Scratch's method for measuring computational thinking. Utilizing a publicly available dataset containing over 230,000 Scratch projects, a latent variable model was proposed. The model considers computational thinking as a latent variable with seven indicators. Confirmatory factor analysis was used to assess the quality of the proposed model. Our results showed that five components of computational thinking were measured satisfactorily, while two were below the accepted level.

2 BACKGROUND

After the creation of Scratch, researchers have started analyzing Scratch programs. However, evaluating programs in Scratch proved to be challenging due to the platform's block-based, visual format and the wide range of programming approaches used by beginners. These challenges led to the creation of tools for assessing Scratch programs. Hairball [2] was one of the first tools created, designed to

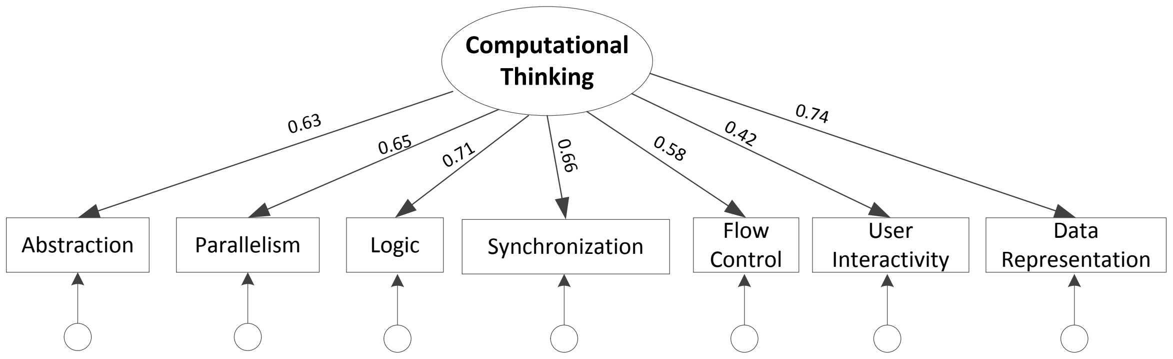

Figure 1: Latent variable model of Dr. Scratch with factor loadings derived from confirmatory factor analysis (CFA)

analyze the code of Scratch projects for common programming patterns and potential coding issues. Following Hairball, Ninja Code Village [17] emerged as a platform, which offered more interactive feedback by helping users improve their coding skills through identifying areas in their projects that could be optimized or corrected. Finally, Dr. Scratch [13] was introduced to provide a more comprehensive evaluation, assessing computational thinking skills across seven indicators: abstraction, parallelism, logic, synchronization, flow control, user interactivity, and data representation. Each component is measured on a scale from 0 to 3, with 0 being the lowest and 3 the highest. The final score is the sum of all 7 components, thus ranging from 0 to 21 [13].

Several studies have investigated the quality of the Dr. Scratch metrics. A study by Moreno-León et al. [14] compared Dr. Scratch scores with Halstead's metrics and McCabe Cyclomatic Complexity, where vocabulary and length were the selected measures. Ninety-five Scratch projects were selected for the study, with a wide range of Dr. Scratch scores, varying from 5 to 20. According to the results, both complexity measures exhibited a strong positive correlation with the scores from Dr. Scratch. Another study by Moreno-León et al. [13] examined the ecological validity of Dr. Scratch. The sample size consisted of 109 participants, aged between 10 and 14 years, from 8 different Spanish schools. Each participant submitted a Scratch project to Dr. Scratch first. Based on the feedback, students could improve their Scratch projects using the recommendations and suggestions provided by the tool. The results showed a statistically significant score increase based on the feedback by Dr. Scratch. Convergent validity of Dr. Scratch was studied by Moreno-León et al. [16]. Fifty-three Scratch projects were evaluated by 16 specialists with a solid understanding of computer science education in the first stage of the experiment. More than 450 evaluations were conducted. The same projects were graded by Dr. Scratch in the second stage. A strong correlation was identified between scores from Dr. Scratch and evaluations by computer science education specialists. Last but not least, a discriminant validity of Dr. Scratch was demonstrated by Moreno-León et al. [15], who examined 250 Scratch projects, which were segmented into five categories, including games, art, music, stories, and animations.

3 METHOD

To assess the quality of the Dr. Scratch measurement scale, we first obtained a dataset of Scratch projects [1], which was constructed by Aivaloglou et al. [1]. The authors collected data from more than 250,000 Scratch projects, from more than 100,000 different users. After collecting data from the Scratch repository, authors also analyzed the collected projects with Dr. Scratch. As a result, the dataset comprises 231,050 Scratch projects, which were successfully evaluated using Dr. Scratch metrics [1].

After obtaining the dataset, a Grades table was extracted. A latent variable model was constructed based on the data in the Grades table. Latent variable models are statistical models that relate a set of unobservable (latent) variables and a set of observable (indicator) variables [4]. In our study, computational thinking was introduced as a latent variable with seven indicators, namely, abstraction, parallelism, logic, synchronization, flow control, user interactivity, and data representation. We used confirmatory factor analysis to examine the validity, reliability, and factor structure of the proposed measurement model. The model is presented in Figure 1.

4 DATA ANALYSIS AND RESULTS

Confirmatory factor analysis (CFA) was conducted using IBM AMOS 26. We used Microsoft Excel and IBM SPSS for calculating means, standard deviations, composite reliabilities (CR), and average variances extracted (AVE). The model with estimates for factor loadings is presented in Figure 1.

The χ^2 value (χ^2 (14) = 66,057) was significant, and the RMSEA was greater than the suggested threshold of 0.08 (RMSEA = 0.143). This would suggest that we had no statistical support for accepting the proposed model. However, in line with representative literature [e.g., 3], χ^2 may not be the only appropriate standard, particularly when sample sizes are large. Accordingly, we used additional fit indices to assess the goodness of fit, including GFI, RMR, NFI, IFI, and CFI. GFI was above 0.9 and RMR was lower than 0.1, which indicated a good fit of our model [8]. Furthermore, NFI, IFI, and CFI were all slightly below 0.9, but still acceptable. According to these results, we concluded that the overall model-data fit was acceptable. The measurement model fit indices are presented in Table 1.

[1]https://github.com/TUDelftScratchLab/ScratchDataset

Table 1: Model fit indices for Dr. Scratch model

χ^2	df	$p\ (\chi^2)$	GFI	RMR	NFI	IFI	CFI	RMSEA
66,057	14	0.001	0.919	0.058	0.870	0.870	0.870	0.143

df degrees of freedom, GFI goodness of fit index, RMR root mean square residual, NFI normed fit index, IFI incremental fit index, CFI comparative fit index, RMSEA root mean square error of approximation

Table 2: Means, Standard Deviations, Loadings, Composite Reliabilities (CR), and Average Variances Extracted (AVE) for Dr. Scratch model

Latent	Indicator	M	SD	Loadings	CR	AVE
	Abstraction	1.057	0.796	0.63		
	Parallelism	1.148	1.079	0.65		
	Logic	0.756	1.092	0.71		
CT	Synchronization	1.233	1.074	0.66	0.821	0.402
	Flow Control	1.887	0.610	0.58		
	User Interactivity	1.563	0.530	0.42		
	Data Representation	1.273	0.682	0.74		

The standardized factor loadings, composite reliability (CR), and average extracted variance (AVE) are presented in Table 2. Standardized factor loadings for abstraction, parallelism, synchronization, logic, and data representation were all higher than 0.6, varying from 0.63 to 0.74. Such results pointed toward satisfactory convergent validity of these components [7]. On the other hand, factor loadings for user interactivity and flow control were below the acceptable level. CR was higher than the suggested threshold of 0.8 (CR = 0.82), which confirmed the reliability of the computational thinking construct [7]. On the other hand, AVE was lower than 0.5 (AVE = 0.40), which indicated that the convergent validity of the proposed measurement model might be weaker than anticipated.

5 DISCUSSION

Dr. Scratch is an assessment tool for evaluating Scratch projects based on seven components of computational thinking. This study employed confirmatory factor analysis to evaluate the quality of the Dr. Scratch measurement scale. To construct a measurement model, computational thinking was introduced as a latent variable with seven indicators, corresponding to seven components of computational thinking used by Dr. Scratch. While several studies have previously examined validity and reliability of Dr. Scratch on smaller samples, our study utilized a large sample of more than 230,000 Scratch projects.

According to the results of the confirmatory factor analysis, factor loadings of abstraction, parallelism, synchronization, logic, and data representation were above the selected threshold of 0.6. Such a threshold indicates that at least 36% of the variance in the aforementioned components is explained by computational thinking. In this context, we consider that five computational thinking components were measured satisfactorily. In addition, Hair et al. [7] suggested that, ideally, a factor loading should be at least 0.7. In this case, 49% of the variance in the observed variable is explained by the latent. According to our results, only data representation and logic surpass the 0.7 threshold.

Factor loading for flow control was slightly below the threshold (0.58), while a value for user interactivity was only 0.42. Consequently, only 18% of the variance in the user interactivity is explained by computational thinking. Accordingly, flow control and user interactivity where not measured effectively.

Based on the results, CR of the proposed model (CR = 0.82) demonstrated sound reliability and high level of internal consistency. This suggests that seven components consistently measure computational thinking. However, AVE was below 0.5 (AVE = 0.4), showing that, on the average, only 40% of the variance of the computational thinking components is explained by computational thinking. Values of CR and AVE revealed a discrepancy between internal consistency and the amount of variance explained by the computational thinking. Such a discrepancy could be attributed to: low indicator quality, low factor loadings or measurement errors [6]. Regarding the low indicator quality, the high CR suggests that the indicators are reliably measuring the same construct, but the low AVE indicates that the indicators may not be capturing the construct very well. This means that while Dr. Scratch assessment items are consistent with each other, they do not explain much of the variance of the computational thinking. Concerning potential low factor loadings, loadings for user activity and flow control were lower than anticipated. Since AVE is a function of the squared factor loadings, lower loadings can result in a lower AVE even when CR is high. Regarding the potential presence of a measurement error, a lower than expected AVE could be due to high measurement error in the indicators. Namely, even if the indicators are internally consistent, significant measurement errors can reduce the proportion of variance explained by the construct.

5.1 Limitations

The results are primarily limited to the data extracted from publicly available dataset, which includes only projects submitted into the Scratch repository up until 2017. It is possible that the programming habits of Scratch users have evolved over time. Another limitation exists, because Scratch projects were not randomly selected from

Scratch repository. Instead, a scraper program collected the most recent projects available at the time it was running [1]. According to Aivaloglou et al. [1], this limitation was somehow mediated by collecting a large dataset, which comprises around 1.3% of 19 million shared Scratch projects.

6 CONCLUSION

This study evaluated the quality of Dr. Scratch measurement scale using a publicly available dataset with more than 230,000 Scratch projects submitted to the Scratch repository up until 2017. According to the results of the confirmatory factor analysis, five computational thinking components were measured satisfactorily, whereas two were below the accepted level. In addition, lower than anticipated average variance extracted indicated potential issues with the measurement model, such as weak indicators. To address these concerns, we plan to further investigate Dr. Scratch scale in the future, using the same dataset. Exploratory factor analysis could be a valuable starting point for potential scale refinement. In addition, it would be beneficial to conduct the same analyses on Scratch projects submitted after 2017 and compare the results.

REFERENCES

[1] Efthimia Aivaloglou, Felienne Hermans, Jesús Moreno-León, and Gregorio Robles. 2017. A dataset of scratch programs: scraped, shaped and scored. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 511–514.

[2] Bryce Boe, Charlotte Hill, Michelle Len, Greg Dreschler, Phillip Conrad, and Diana Franklin. 2013. Hairball: Lint-inspired static analysis of scratch projects. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 215–220.

[3] Kenneth A Bollen. 1989. *Structural equations with latent variables*. John Wiley & Sons.

[4] Kenneth A Bollen. 2014. *Structural equations with latent variables*. John Wiley & Sons.

[5] Bostjan Bubnic and Tomaz Kosar. 2019. Towards a Consensus about Computational Thinking Skills: Identifying Agreed Relevant Dimensions.. In *PPIG*. 69–83.

[6] Claes Fornell and David F Larcker. 1981. Evaluating structural equation models with unobservable variables and measurement error. *Journal of marketing research* 18, 1 (1981), 39–50.

[7] J Hair, B Black, B Babin, and R Anderson. 2010. *Multivariate data analysis, 7th Edition*. Pearson Prentice Hall.

[8] Litze Hu and Peter M Bentler. 1999. Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural equation modeling: a multidisciplinary journal* 6, 1 (1999), 1–55.

[9] Filiz Kalelioglu, Yasemin Gülbahar, and Volkan Kukul. 2016. A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing* 4, 3 (2016), 583.

[10] Michael Lodi and Simone Martini. 2021. Computational thinking, between Papert and Wing. *Science & education* 30, 4 (2021), 883–908.

[11] Joseph A Lyon and Alejandra J. Magana. 2020. Computational thinking in higher education: A review of the literature. *Computer Applications in Engineering Education* 28, 5 (2020), 1174–1189.

[12] Ana Melro, Georgie Tarling, Taro Fujita, and Judith Kleine Staarman. 2023. What else can be learned when coding? A configurative literature review of learning opportunities through computational thinking. *Journal of Educational Computing Research* 61, 4 (2023), 901–924.

[13] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2015. Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia* 46 (2015), 1–23.

[14] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2016. Comparing computational thinking development assessment scores with software complexity metrics. In *2016 IEEE global engineering education conference (EDUCON)*. IEEE, 1040–1045.

[15] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2017. Towards data-driven learning paths to develop computational thinking with scratch. *IEEE Transactions on Emerging Topics in Computing* 8, 1 (2017), 193–205.

[16] Jesús Moreno-León, Marcos Román-González, Casper Harteveld, and Gregorio Robles. 2017. On the automatic assessment of computational thinking skills: A comparison with human experts. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2788–2795.

[17] Go Ota, Yosuke Morimoto, and Hiroshi Kato. 2016. Ninja code village for scratch: Function samples/function analyser and automatic assessment of computational thinking concepts. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 238–239.

[18] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.