Abstracting the Understanding and Application of Cognitive Load in Computational Thinking and Modularized Learning

Taylor Gabatino(✉), Michael-Brian C. Ogawa, and Martha E. Crosby

Department of Information and Computer Sciences, University of Hawaii at Manoa, 1680 East-West Road, Honolulu, HI 96822, USA
{tgabatin,ogawam,crosby}@hawaii.edu

Abstract. The purpose of this study is to determine whether modularized standalone sections within topics in computer science are deterministic in the performance of students studying subjects that involve computational thinking. Teaching methods regarding this form of cognition within the realm of computer science is presented with a limited understanding in how students think and analyze problems when presented material with ambiguous forms of approach. The method and scope of the work involve the presentation of topics in computer science in a modularized form that determines whether correctness is a function of time based on cognitive load introduced in computational thinking concepts, involving base conversions of transposition ciphers and programming fundamentals.

Keywords: Cognitive load and performance · Shared cognition · Team performance and decision making · Understanding human cognition and behavior in complex tasks and environment

1 Introduction

Computational Thinking (CT) has become one of the most important skills of the 21st century, with problems being solved not only in the realm of computer science, but its applicable partners within the educational community, and a primary fundamental in modern scientific disciplines [2, 5]. As CT skills increase within the digital society, teaching these abilities become an essential factor in the arsenal of educators; however, there is a limitation of understanding between both students and instructors in what methodologies are best employed in CT. Although Wing [5] states CT to be a fundamental skill, there is still much speculation as to what it truly embodies, with a general knowledge consensus that it comprises of problem-solving skills such as abstraction and decomposition [3, 5]. This invokes a need to design teaching methodologies that are best proposed for this - in particular, how cognitive load changes with the level of difficulty in CT.

Methodologies involved in teaching CT are often invoked in assisting students to understand concepts in computer science that are devised to test pattern recognition, abstraction, and algorithmic design [8]. As the development of teaching computer science grows substantially through the years, there exists the correlation of the demand for CT

D. D. Schmorrow and C. M. Fidopiastis (Eds.): HCII 2022, LNAI 13310, pp. 273–286, 2022.
https://doi.org/10.1007/978-3-031-05457-0_22

education to increase as a result of the parallels of technological growth in the digital society [4]. Although various studies have been introduced to meet the demands of teaching CT, there is a growing complexity in the field to also meet the difficulties of employers outside of the technological realm.

As computational degrees become increasingly lucrative in modern society, it is argued that it should be a basis for comprehension, as it begins to influence disciplines in fields beyond STEM, with areas of study active in the algorithmic design within social sciences and humanities [1, 6]. In addition, curriculum across the United States along with an increasing number of other countries have begun to adopt CT as a necessity in addition to the base requirements in K-12 schools [14]. Furthermore, universities have also begun to propose computational thinking as a requirement for non-STEM majors systematically introducing them to the teaching curriculum [2]. Various assertations recognize that there is also a common misconception associated with CT, with its alignment primarily associated in computer science [14]. Wing [17] however, states that students should be given the opportunity to participate in a digital society by becoming competent in CT, regardless of their relation to computer science and STEM.

The understanding of CT to abstraction and problem solving; however, is vastly misunderstood. In addition to the pilot study performed, a survey amongst students in the experiment were distributed asking what they best believed computational thinking to be.

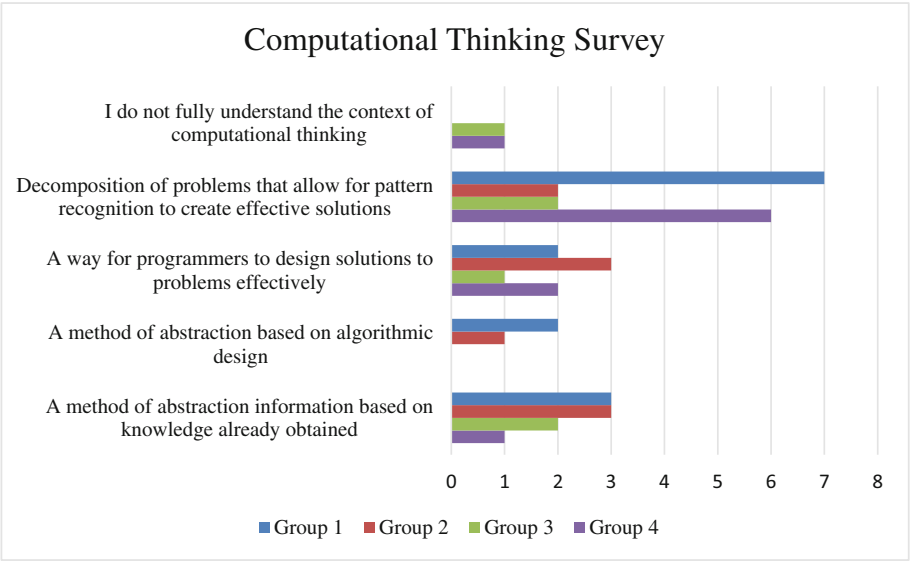

Fig. 1. Survey of students describing computational thinking

Results from the survey showed varying discrepancies between the understanding of CT, and how to best use it for abstraction and decomposition of proposed problems. This is further expanded within the pilot study, where questions introduced in modularized pedagogy propose varying cognitive load as a function of time in seconds.

We specifically begin this analysis through discussion of both CT and Cognitive Load, and its relation to methodologies involved in teaching these approaches and methods of thinking to students. This discussion can then further how human cognitive architecture can thus be chunked into sub-divisions that allow for the highest capacity of working memory to be recorded quantitatively and qualitatively. The corresponding experiment associated with how working memory intertwines with cognitive load are thus introduced. Lastly, we look at the associations between correctness of question and time when tested with topics involving ASCII, HEX, Binary, and Programming Fundamentals, providing the evidence for the principle of cognitive load and its determinant of performance.

2 Background

2.1 Computational Thinking and Cognitive Load

Computational thinking, as defined by Wing are the thought processes that are involved with the formulation of problems and their relative solutions such that they are represented effectively by a potential information-processing agent [6]. It is within reason, thus, that there is a significant level of cognitive load that is placed on individuals when presented with problems that require mental processing contributed to learning, memory, and problem-solving [7]. This cognitive load, based on concepts in cognitive science act as the agent of an information-processing approach, with the mind divided into sensory, working, and long-term memory [7].

Cognitive Load theory as described by Sweller is based on the model that the mind works in two sets [7]. The first set proposes that the human working memory is constrained to a limited capacity, which determines performance based on cognitive load, while the second imposes mechanisms that are designed to elude its weaknesses [7]. As there are a multitude of sensory processors that are used within a cognitive load, the information processing works in conjunction with these sensors to meet the demands of human cognitive architecture [7].

Working memory, often termed *short-term memory* or *intermediate memory* interchangeably, refers to the temporary availability of information that is recalled through a recollection of recent scenarios [10]. This concept, based on Miller [11] introduces a capacity on information processing such that it can be limited to several familiar chunks [10]. These chunks of information are then loaded into working memory, where the cognitive load factor is then introduced; however, the implications presented in the limitation of working memory place a strong emphasis on instructional design and limiting its overstimulation [8]. High working memory otherwise places a complex demand for learners who do not have knowledge that is specific to the domain in where cognitive load is to be implied [19]. Furthermore, the complex knowledge structures that would increase this type of information would be better served through the chunking of individual elements within a single element [19]. Based on the implications imposed on working memory, tasks designed with high cognitive load should be taught in chunks, which allow for better retrieval of information.

Long-term memory is situated on the opposing spectrum of cognitive architecture, with a focus on recollection and processes involved in problem solving [13]. Since

this information is characterized as the ability to store large amounts of information in memory, it is argued the long-term memory can best define human cognitive ability as stored knowledge, as opposed to the ability to engage in complicated reasoning and logic within working memory [8]. This can thus limit the need to access working-memory as a single individual segment in human cognition, increasing the capacity for processing within working memory to be a simpler process [9, 15]. Based on this information, CT can be best taught when modularized structures are introduced in an efficient manner that decrease the cognitive load of working memory by enforcing concepts sequentially through long-term memory.

2.2 Computational Teaching Methodologies

The pedagogy of computational teaching methodologies raises an interest within significant educational communities, as CT becomes a universal skill set not limited to only computer scientists [9]. The further development involved in CT can thus result in the growth of a learners' abilities to make connections within their knowledge, question what is known, and further express themselves through active engagement [12]. Methodologies further designed to adapt to this difference in teaching must then be implemented in a manner that consider its effectiveness within instructional design, regarding working memory and the limitations of chunks of information, and long-term memory and the cognitive ability to manage storage and retrieval [8].

The importance of understanding CT and teaching it relies on the significant factor that there is an introduction of new challenges for instructors that do not hold a background in computational thinking, with institutions struggling with how to coordinate information technologies in their systems [15]. Given CT is more adept to problem-solving and its processes, it is best understood when decomposed into a staged cycle, where the congruence of CT and problem-solving intertwine [15]. These staged cycles, as introduced in modularized formats, is best designed when taught in progression, where the buildup of one concept is dependent on understanding another [17]. This can decrease the cognitive load introduced on working-memory when instructional design is paired with methodologies that employ understanding of problem approach and deconstruction through introducing them in formats that are designed with long-term teaching methods. Introducing algorithmic concepts to best improve these methods then, are based on modularized design that have yet to be determined. In this scenario, the following questions are imposed to enhance the teaching of computational thinking principles:

1. Does modularized structure in computer science education have a dependency in its placement within curriculum?
2. Does teaching concepts within computational thinking depend on the level of cognitive load introduced in a specific topic?
3. Will the restructuring of modules based on this cognitive load have an effect on the understanding of basic CT principles, such as basic transposition and conversions in hexadecimal, binary, and ASCII?

It is thus important to design modules in teaching computational thinking that are best designed to augment cognitive overload, where the processing of demands that are created from the learning task exceeds the cognitive system [20]. These teaching

methodologies much then employ the steps involved in CT, with the process of decomposing a problem, recognizing these specific patters, and thus using a form of abstraction to both generalize and formulate the probable solutions [1]. These logical methodologies can then be augmented to solve the potential problems imposed [1]. Those without prior computational thinking skills may have a more limited capacity in processing complex topics – therefore, design for cognitive overload must be implemented in teaching methodologies, with respect to schemas that are best augmented for long-term retention and less cognitive overload on working memory.

3 Pilot Study

The study below seeks to determine whether modularized formats in an introductory computer science course demonstrating methodologies in computational thinking are time-dependent based on the cognitive load of individual questions and their placement in a schema. Considering a given architecture of a lesson plan by Dr. Ogawa introducing hexadecimal, binary, and ASCII, modules are restructured to be determinants in the cognitive load introduced in the sample experiment, with time being the indicating factor in the measurement of cognitive load and relationships to accuracy. Students in the study were given segmented chunks of videos from two lectures introducing transposition ciphers and simple base number conversions. Given the modular formats did potentially have an impact on cognitive load, there should be statistically significant data correlating results to this hypothesis.

4 Methodology

4.1 Modularized Format

The examination of the experiment was completed within students in an Information and Computer Science introductory level course. The course, labeled as ICS 101 is affiliated with the University of Hawaii at Manoa, covering introductory level concepts in computer science and software used in a variety of productivity environments. The pilot study of this experiment was conducted in an online format in the Fall 2021 semester, with a series of questions in a sequential structure summarizing knowledge based in *Transposition Ciphers*, *ASCII, HEX, and Binary*, *Encryption and Decryption*, and lastly *Programming Fundamentals*. The total group of students, ($n = 39$) were divided into 4 independent subsets based on class section, where each individual group were given the same segments and topics; however, they were organized and presented in varying preparations depending on the class section the group belonged in.

The insight into the development of CT skills of students was taken with cognitive load being a function of time. Given the difficulty involving understanding binary, hexadecimal, and ASCII, the experiment seeks to determine whether changing the structure of these modules is a factor in best understanding concepts introduced in CT, with respect to working-memory and the chunking of information.

The proposed lectures involve recordings of pedagogy performed by Dr. Ogawa, highlighting concepts involved in introductory level programming courses. These lectures were then chunked, in a similar fashion to Miller [11] and his proposal of the working memory. The formatting of these modularized schemas was designed based

on the division of curriculum into discrete modules that were short in duration to test the limits of working memory [18]. By dividing and chunking these subcategories into modularized formats, students were limited in their abilities to access the next or previous chunk depending on which version of the test they received. The following diagram organizes these original pedagogies, with the modular formats representing the chunks of information they would receive in varying orders based on which group received the test. In contrast to methodologies that create a linear format of the presentation of information, modularized learning allows stand-alone independent units that can be taken in different orders at varying speeds, associated with the chunking of information and greater intensity in delivery of information [16]. The decomposition of the potential ordering of the tests were then divided, re-structured, and distributed to the respective groups. Data that was collected in this study ensured no personal information was released, and that the confidentiality of those involved would be protected. The results of this study hope to improve the understanding and methodologies involved in teaching computational thinking to those both within and outside of the STEM community.

Fig. 2. Structure of original modular lecture in transposition ciphers

It is observed in these pedagogies that although presented in a sequential format, the introduction of quizzes in between each module does not determine the level of understanding of students if there is no qualitative factor that indicates significant knowledge gained in the previous module. This leads to a recurring cycle of students returning to the previous section in which no form of redirection can best be determined without a quantitative approach. In addition, students may attribute their inability to understand current schemas to the previous module, when alternative segments of the schema may be required for further understanding. Furthermore, it is observed in these lectures that background knowledge associated with transposition ciphers is assumed; therefore, the addition of this knowledge was introduced as a subset of number representations and their potential bases, such as hexadecimal, ASCII, and binary. Given CT is a process of decomposition, this also relies heavily on the need to match specific teaching methodologies to the level of proficiency and understanding of the student while introducing new concepts and assessing the CT skills [3]. Since this is an introductory level course of topics in Computer Science, the proficiency of students was considered, and additional information was introduced to match the level of understanding of students to allow them to continue the task.

As cognitive load is to be observed between modules, the introduction of scored quizzes was introduced between each schema to determine level of understanding. These quizzes would be used in the statistical collection, with scoring and time recorded between each question being the indicating variables of cognitive load.

The first group received the original set of lecture activity, with no changes prior to the original teaching methodology. The order in which they received their modules are as follows (Table 1).

Table 1. Group 1 chunked module ordering

Order	Module
1	Transposition ciphers
2	ASCII, HEX, and binary
3	Encryption and decryption
4	Programming fundamentals

Using this as a control group, as the original base lecture, the modules were then organized in a manner introduced by Fig. 2 above. Since these sections were to be divided into smaller chunks, it was imperative to also link the information relevant to these divisions that students may not have access to. This is where the additions to module 2 were introduced, as it was retrieved from a previous lecture that introduced ASCII, HEX, and Binary as an independent lecture. The following tables show the finalized ordering of the chunked modules based on grouping, in addition to the ASCII, HEX, and Binary module which included supporting material (Tables 2, 3 and 4).

Table 2. Group 2 chunked module ordering

Order	Module
1	ASCII, HEX, and binary
2	Transposition ciphers
3	Encryption and decryption
4	Programming fundamentals

Table 3. Group 3 chunked module ordering

Order	Module
1	Encryption and DECRYPTION
2	Transposition ciphers
3	ASCII, HEX, and binary
4	Programming fundamentals

4.2 Analyzing Student Outcome

The resulting groups were then tested on video lectures ordered and chunked into their corresponding subsections. Questions were introduced between each schema with relation to the module content, ranging from a variety of multiple-choice types to radio-box

Table 4. Group 4 chunked module ordering

Order	Module
1	Programming fundamentals
2	Encryption and decryption
3	ASCII, HEX, and binary
4	Transposition ciphers

selections. The test used to determine statistical significance were ANOVA tests to determine variance in results. Given there were more than three groups independent of each other, this was determined to be the best test to determine statistically significant data based on time and correctness.

Groups 1 and 4, with group 1 being the primary control group scored similarly in average in comparison to group 2 and 3, who also had similar results in respect to scores. In relation to previous findings, although group 1 and 4 scored worse on average in comparison to group 2 and 3, the previous survey introduced asking what students believed computational thinking to be had conflicting results when compared to the average scores across all sections. In addition, when considering the placement of each module, it is observed that group 1 and 4 had noticeable inverses with respect to their structure, receiving scores on the lower end of the study, while groups 3 and 3 received higher averages.

Table 5. Average total scores across groups

Group	Average score
Group 1	55%
Group 2	64%
Group 3	65.6%
Group 4	41.75%

Average Total Scores Across Groups. Table 5 shows the average scores between each group who had the same set of questions presented in varying orders depending on the test version received. In relation to Fig. 1, which displays the results of students' thoughts on computational thinking, those in Group 1 and 4 who determined that computational thinking is defined as the decomposition of problems that allow for pattern recognition to design effective solutions scored worse overall in comparison to their similar counterparts. Group 2 and 3 on average, determined that computational thinking was either solely for those in the computer science domain, or computational thinking was based on knowledge already obtained. A possible explanation between these differences in average scores is that students who assumed computational thinking being a decomposition

of problems approached the quizzes this way, while those that solved the problem in their working memory based on knowledge obtained prior approached these questions in a similar manner. This can thus relate to working and long-term memory, and how these associations can be determining factors in how students solve problems. Since problems can be identified within specific problem categories, relevant schema associated can be retrieved from memory, and utilized with the information that is otherwise specific to the problem [17].

5 Results

The following sections present the findings of the data collected in qualitative and quantitative schemes from the modularized components of each section. Given the assumption that independent schemas have an impact of the cognitive load of students, a significant recording of time and accuracy should be observed. The following observations were then analyzed through the analysis of variance (ANOVA) in order to determine whether there were significant differences between groupings. This statistical analysis will determine the effective between the factors contributing to the cognitive load between groupings.

5.1 Quantitative Data

While Table 5 displayed data that showed the average total scores across groups as a sum of their sections, the following tables depict the statistical range and average based on the standalone sections. This however, depicted no findings of statistically significant data based on ANOVA test; however, this gave insight into the variance of scores based on groupings and position in the test, as depicted through the following tables and figures.

Table 6. Statistical data scores of transposition ciphers

Groups	Count	Sum	Average	Variance
G1S1	14	8.3	60%	0.037
G2S1	9	4.6	52%	0.058
G3S1	6	3	50%	0.166
G4S1	10	6	60%	0.069

Table 6 shows the statistical data of the average scores between groupings on the Transposition Cipher schema. Between these groupings, groups 1 and 4, and 2 and 3 respectively had similar results in the average scores of their tests.

The module for Sect. 2 tested the understanding of ASCII, HEX, and binary. Here, groups 1, 2, and 3 scored within a similar range, while group 4 scored significantly lower (Tables 7 and 8).

Table 7. Statistical data scores of ASCII, HEX, and binary

Groups	Count	Sum	Average	Variance
G1S2	14	9.75	70%	0.088
G2S2	9	6	67%	0.062
G2S3	6	4.25	71%	0.085
G2S4	10	4.75	48%	0.061

Table 8. Statistical data scores of encryptions and decryption

Groups	Count	Sum	Average	Variance
G1S3	14	8.5	61%	0.151
G2S3	9	5.5	61%	0.095
G3S3	6	4.25	71%	0.160
G4S3	10	7	70%	0.136

The module for Sect. 3 tested the understanding of encryption and decryption. Here the average scores between groupings remained within the same relative range, with groups 1 and 2 recording a similar average range in score, and groups 3 and 4 also showing close averages (Table 9).

Table 9. Statistical data scores of programming fundamentals

Groups	Count	Sum	Average	Variance
G1S4	14	5.5	39%	0.074
G2S4	9	4	44%	0.137
G3S4	5	2.5	50%	0.156
G4S4	10	3.5	35%	0.016

The module for Sect. 4 tests the understanding of basic programming fundamentals. It is observed here that similar scorings of averages across tests were detected between groups 1 and 4 and groups 2 and 3 respectively.

5.2 Modularized Results and Cognitive Load

In correspondence to statistically significant results, one of the questions within the modules exhibited a statistically significant difference with respect to both time and results. In relation to time differentials, a similar question depicted trends with relation to cognitive load factors introduced through speculation of time. The questions that thus

determined the analysis of cognitive load as a function of time through this survey are as follows, with the first question being the statistically significant result.

– "If it is true that uppercase letters have different ASCII codes than lowercase letters, what is the difference in value needed to change an uppercase letter to a lowercase letter in ASCII?"
– "Conversion of the word "tEsT" to ASCII yields what set of decimal digits when decrypted?"

The first question that showed statistical significance in relation to time and results was Question 4 on the exam under *ASCII, HEX, and Binary*, which prompts: "If it is true that uppercase letters have different ASCII codes than lowercase letters, what is the difference in value needed to change an uppercase letter to a lowercase in ASCII?" Results determined from the ANOVA in Table 10 reveal the statistically significant differences between the groups, pertaining to this question, with ($p = 0.001$). Since this value denotes a significant level less than ($p = 0.05$), the null hypothesis can be rejected noting that there is a significant difference between groupings. Furthermore, the time differential and statistical significance of the results align with the results of accuracy as depicted in Table 11, where ($p = 0.02$), again depicting statistical significance between groupings.

Table 10. ANOVA factor of statistically significance of time in question 4 (ASCII, HEX, binary)

Source	SS	df	MS	F	P-value	F crit
Between Groups	8.6E–05	3	2.87 E-05	6.77	0.001	2.87
Within Groups	0.00014	35	4.24 E-06			
Total	0.00023	38				

Table 11. ANOVA factor of statistically significance of correctness in question 4 (ASCII, HEX, binary)

Source	SS	df	MS	F	P-value	F crit
Between Groups	2.14	3	0.713	3.65	0.02	2.87
Within Groups	6.83	35	0.195			
Total	8.97	38				

Groups 1 and 3 as depicted in Table 12 under the average column, with the values depicting seconds, performed significantly faster than groups 2 and 4. With reference to Table 6, the average scores on this section showed a relation between the time differential and accuracy. The modularized format between these groupings had similarities

Table 12. Summary of time averages on question 4 (ASCII, HEX, binary)

Groups	Count	Sum	Average	Variance
G1S2Q4	14	0.0062	39	4.60 E-08
G2S2Q4	9	0.0045	44	2.29 E-07
G3S2Q4	6	0.0276	37	2.90 E-05
G4S2Q4	10	0.0049	43	1.16 E-07

in structure, with the transposition cipher module occurring directly before the ASCII, HEX, and Binary schema. This could serve as a potential explanation of the similarities in score and time between these questions. Although not directly relation to the cognitive load in analysis, the data presented within this variation could present an interpretation of differences between mental models and understanding, where further study can be performed through content analysis.

The second question depicting an observed time differential was Question 4 under the Encryption and Decryption module, which prompted students to convert the word "tEsT" to its ASCII code and corresponding decimal digits. As cognitive load being a function of time, it can be observed in Table 13 that a similar trend between groups 1 and 4 were depicted when asked questions that were determinant on the understanding of knowledge based on ASCII, HEX, and Binary Fundamentals. Time differentials observed in Table 13 thus show a longer time spent on attempting to convert the word, with shorter time spent on average between groups 2 and 3.

This trend can be further observed through a reference to Table 5, where the average scores across groupings were collected. Similarly, the average scores across subsections of the groups also showed similarities to this trend, with groups 1 and 4 scoring similarly on Transposition Ciphers and Programming Fundamentals respectively. This can potentially be attributed to the inverse and "flip" in the modules, where although similar in design, were different in comparison to groups 2 and 3, who had some form of ASCII, HEX, and Binary or Encryption and Decryption before Transposition Ciphers and Programming.

Table 13. Summary of time averages on question 4 (encryption and decryption)

Groups	Count	Sum	Average	Variance
G1S2Q4	14	0.0078	48	2.23 E-07
G2S2Q4	9	0.0034	33	6.36 E-08
G2S3Q4	6	0.0068	39	2.01 E-06
G2S4Q4	10	0.0131	54	1.79 E-06

6 Conclusions

6.1 Future Development and Research

Results from the study allow us to revisit the questions imposed in this experiment, where we relate the individual discoveries within the study to questions involving cognitive load and potential for future research. The analysis of the findings summarizes and identify themes for future development of this study, in relation to cognitive load factors that are determinant between the most common accurate and inaccurate answers to questions between subsets and groups. Although cognitive load as a function of time was determined between these sets, potential for future study may be determine through content analysis, where the choice of wording in questions can identify themes in how students analyze algorithmic thought and problem-solving solutions. Furthermore, analysis on the reordering of these modules and student performance can be quantified to determine the best ordering of computational material to prevent cognitive overload amongst those with little computational strength.

The potential for future research on computational teaching methods for students analyzed through independent modularized formats in CT involve future analysis of concepts to discover variables associated with potential cognitive load. Additional resources can be supplemented between course material in computer science that can support opportunities to research specific domains that affect cognitive load in computational thinking. The discovery of these variables through content analysis of text, discourse analysis between students and instructor, and quantitative analysis of cognitive load can be further researched for beneficial impact in the educational community.

Acknowledgements. This material is based upon work supported by the National Science Foundation (NSF) under Grant No. 1662487. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

1. Imberman, S., Sturm, D., Azhar, M.: Computational thinking: expanding the toolkit. J. Comput. Sci. Coll. **29**(6), 39–46 (2016)
2. Lamprou, A., Repenning A.: Teaching how to teach computational thinking. In: Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (2018)
3. De Jong, I.: Teaching computational thinking with interventions adapted to undergraduate students' proficiency levels. In: Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE, pp. 571–572. Association for Computing Machinery (2020)
4. De Jong, I., Jeuring, J.: Computational thinking interventions in higher education: a scoping literature review of interventions used to teach computational thinking. In: Koli Calling 2020: Proceedings of the 20th Koli Calling International Conference on Computing Education Research, pp. 1–10 (2020)
5. Wing, J.M.: Computational thinking. Commun. ACM **49**(3), 33–35 (2006)
6. Resnick, L.B.: Education and Learning to Think. National Academy Press, Washington, DC (1987)

7. Wing, J.M.: Computational thinking—what and why? In: Article of the Magazine of the Carnegie Mellon University School of Computer Science (2011)
8. Shaffer, D., Doube, W., Tuovinen, J.: Applying cognitive load theory to computer science education. In: Petre, M., Budgen, D. (eds.) 15th Annual Workshop of the Psychology of Programming Interest Group, pp. 333–346 (2003)
9. Sweller, J., van Merrienboer, J.J.G., Paas, F.G.W.C.: Cognitive architecture and instructional design. Educ. Psychol. Rev. **10**, 251–296 (1998)
10. Li, Y., et al.: Computational thinking is more about thinking than computing. J. STEM Educ. Res. **3**, 1–18 (2020)
11. Cowan, N.: George Miller's magical number of immediate memory in retrospect: Observations on the faltering progression of science. Psychol Rev. **122**, 536 (2015)
12. Miller, G.A.: The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol. Rev. **63**(2), 81–97 (1956)
13. Kong, S., Wang, Y.: Formation of computational identity through computational thinking perspectives development in programming learning: a mediation analysis among primary school students. Comput. Human Behav. **106**, 106230 (2020)
14. Franklin, D., Salac, J., Crenshaw, Z., Turimella, S.: Exploring student behavior using the TIPP&SEE learning strategy. In: Proceedings of the 2020 ACM Conference on International Computing Education Research (2020)
15. Sweller, J.: Cognitive load theory. Psychol. Learn. Motiv. **55**, 37–76 (2011)
16. Labusch, A., Eickelmann, B., Vennemann, M.: Computational thinking processes and their congruence with problem-solving and information processing. In: Kong, S.C., Abelson, H. (eds.) Computational Thinking Education, pp. 65–78. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-6528-7_5
17. Wing, J.M.: Computational thinking's influence on research and education for all. Italian J. Educ. Technol. **25**(2), 7–14 (2017)
18. Dejene, W.: The practice of modularized curriculum in higher education institution: active learning and continuous assessment in focus. Cogent Educ. **6**(1), Art. 1611052 (2019)
19. Gerjets, P., Scheiter, K., Catrambone, R.: Designing instructional examples to reduce intrinsic cognitive load: molar versus modular presentation of solution procedures. Instr. Sci. **32**, 33–58 (2004)
20. Walkington, C., Clinton, V., Ritter, S., Nathan, M.J.: How readability and topic incidence relate to performance on mathematics story problems in computer-based curricula. J. Educ. Psychol. **107**(4), 1051–1074 (2015)