Validation of the Components and Elements of Computational Thinking for Teaching and Learning Programming using the Fuzzy Delphi Method

Karimah Mohd Yusoff[1], Noraidah Sahari Ashaari[2], Tengku Siti Meriam Tengku Wook[3], Noorazeen Mohd Ali[4]
Matriculation Division, Ministry of Education Malaysia, Putrajaya, Malaysia[1]
Software Technology and Management SystemUniversiti Kebangsaan Malaysia
Bangi, Selangor, Malaysia[2, 3, 4]

***Abstract*—Computational Thinking is a phrase employed to explain the developing concentration on students' knowledge development regarding designing computational clarifications to problems, algorithmic Thinking, and coding. The difficulty of learning computer programming is a challenge for students and teachers. Students' ability in programming is closely related to their problem-solving skills and their cognitive abilities. Even though computational thinking is a problem-solving skill in the 21st century, its use for programming needs to be planned systematically taken into account the appropriate components and elements. Therefore, this study aims to validate the main components and elements of computational thinking for solving problems in programming. At the beginning of the study, researchers conducted a literature review to determine the components and the elements of computational thinking that could be used in teaching and learning programming. This validation involved the consensus of a group of experts using the Fuzzy Delphi method. The data were analysed using the Fuzzy Delphi technique, where the experts individually evaluated the components and elements agreed upon prior discussion. A group of experts consisting of 15 people validated 14 components and 35 elements. The results showed that all components and elements reached a threshold (d) value of less than 0.2, a percentage of agreement exceeded 75%, and the Fuzzy score (A) exceeded 0.5. The finding indicates that the main components and elements of the proposed computational thinking are suitable for problem-solving approaches in programming.**

***Keywords*—Expert consensus; focus group; problem-solving; components; elements**

I. Introduction

Teaching and learning methods have evolved globally, where various advancements have introduced over the years. Recently, computer programming is of growing interest in line with the efforts to enhance Science, Technology, Engineering and Mathematics (STEM) based education and career. Besides government and non-governmental agencies, industries also suggest learning institutions to prepare students who have knowledge, understanding, and skills in programming and problem-solving [1]. Indirectly, educators should continuously enrich their experience and skills to provide effective teaching and learning environment.

Programming is a subject that involves problem-solving skills starting from problem formulation to complete program development. Therefore, structured teaching and learning methods for programming should be established by including all steps to solve the problem. Amongst the steps are formulating the problem, planning the solutions, designing the solutions, translating the solutions into programming codes and testing and evaluating the complete program. The main challenge faced by novice programmers in learning programming was related to the cognitive ability of an individual [2][3][4][5]. Based on the cognitive load theory, the teaching design is tailored to reduce the student's load during the thinking process to achieve optimal learning outcomes [6].

Computational thinking is gaining attention among educators, and it is often linked to problem-solving [7]. Computational thinking is considered as a 21st-century skill [8][9] that can build the essential cognitive skills of students [10]. Relationship between computational thinking implementation and students' cognitive level was reported in previous studies [11][12][13], for different purposes. A blended learning model is developed for students to acquire basic programming skills through activities tailored to students' cognitive levels [11]. The activity is designed by considering the three levels of computational thinking skills which is basic, intermediate, and advanced that could be used on the Moodle platform. In contrast, computational thinking is introduced in the context of creative programming activities using Scratch software [13]. Besides, a study to provide new instruments for the measurement of computational thinking and prove the nature of computational thinking through its relationship with cognitive psychological constructs consist of spatial ability, reasoning ability and problem-solving ability [12]. These studies shows the potential of computational thinking in education.

In this study, we proposed the components and elements of computational thinking for problem solving in programming. We believe that by involving appropriate components and elements, computational thinking is potentially to develop problem solving skills for programming. Hence, this article reports the validation process systematically of the components and elements of computational thinking for problem-solving approach in programming. The validation in performed by a group of experts through the Fuzzy Delphi Method (FDM).

The discussion of the following section as follows: Section II is a literature review that leads to the components of CT. Section III describes the validation process in this study.

Section IV discussing about data analysis. Section V details about findings and discussion. Section V conclude the study and further work.

II. Literature Review

Computational thinking skills that derived from computer science [14] is an approach to problem-based teaching and learning that meets the needs of problem-solving skills in the 21st century that has gained the attention among researchers and educators [15]. Computational thinking provides a set of cognitive skills to solve problems that are appropriate for all areas [16][17]. In 1980, Seymour Papert introduced the idea of computational thinking. Later, computational thinking was defined as the application of some basic concepts of Computer Science to solve the problems, designing systems, and understanding human behavior [18]. The computational thinking definition was revised as a thought process for formulating and solving problems in a form that information processing agents can effectively execute [19]. Apart from that, several definitions of computational thinking differ in meaning but generally focused on solving problems [20][21]. Latest, computational thinking is defined as the thinking skills and also the practice to design computation that enable computers to execute the instructions they receive. Computational thinking also explains and interprets what happens in reality as a complex processing of information that takes place in a computer [22]. Based on the proposed definitions, in this study, computational thinking is regarded as a thinking approach to develop problem-solving skills through computing to find solutions.

Computational thinking is the primary skills that are used in the problem-solving process. Various computational thinking skills have been suggested in previous studies [23-26][19][7][27-29] as shown in Table I. As a pioneer of computational thinking, Jannette Wing proposed abstraction, decomposition, generalisation, algorithm and automation skills.

The computational thinking skills proposed by the researchers were almost similar with a few differences. However, the concepts presented in all areas are practically uniform [16]. Based on Table I, similar computational thinking skills include abstraction, decomposition, generalisation (pattern recognition) and algorithm incorporated in this study.

Computational thinking is a cognitive process that involves logical thinking, including the ability to perform abstraction, decomposition, identification of patterns through generalisation, solving the problems sequentially, and evaluation of the results. Therefore, logical reasoning identified as a new component of computational thinking for problem-solving [24]. In programming, programs need to be tested and evaluated; thus, evaluation skills among the best talents in programming [24][25]. As this study focuses on the use of problem-solving skill in programming, computational thinking skills should play a role in line with the problem-solving step in programming. Table II shows the description of computational thinking skills components identified for problem-solving in programming.

To date, there is no consensus on the exact components of computational thinking, but computational thinking can involve multiple components and may not necessarily be cognitive [15]. Therefore, other than skills, dimension and approach were included in this study.

TABLE I. Computational Thinking Skills Suggested by Researchers

Computational Thinking Skills	**Reference**
abstraction, decomposition, generalisation, algorithm, automation	Wing, 2006, 2008, 2011
abstraction, decomposition, generalisation (pattern recognition), algorithm, evaluation.	Selby & Wollard, 2013
logical reasoning, abstraction, decomposition, generalisation (pattern recognition), algorithm, evaluation.	Csizmadia et al., 2015
abstraction, decomposition, generalisation, algorithm, debugging.	Angeli et al., 2016
abstraction, decomposition, pattern recognition, algorithm	Shute, Sub & Asbell-Clarke, 2017
abstraction, decomposition, generalisation, algorithm	Denning, 2017
abstraction, decomposition, data representation, pattern recognition, algorithmic thinking	Rodriguez et al., 2017
abstraction, decomposition, pattern recognition, algorithm	Burbaite, Drasute & Stuikys, 2018

TABLE II. Description of Computational Thinking Skills

Skills	**Description**
Abstraction	The skill to identify and retrieve relevant information to determine key ideas and to remove unnecessary details.
Decomposition	The skill to breakdown the problem to a small section and easy to manage for complex problems. The solution can be implemented part by part until the whole problem is solved.
Pattern Recognition	Skills in observing patterns, tendencies and regularity of data through similarities.
Algorithm	Skill to perform tasks or solve problems step by step.
Logical reasoning	Skill explain what happens by analysing and studying facts by thinking clearly and accurately.
Evaluation	Skill determines whether the algorithm, system or process is working correctly and following its purpose.

Beside skills, dimensions of computational thinking framework that consisted of computational concepts, practices and perspectives [30] is proposed to ensure the delivery and development of computational thinking skills as shown in Table III. The efforts include teaching delivery, student involvement practically and assessment of student performance.

Apart from dimensions, it is essential to stimulate the thinking processes that lead to computational thinking skills. In this study investigates computational thinking approach, which is a practice applied during teaching and learning session. There are five approaches of computational thinking, which are tinkering, creating, debugging, collaborating, and persevering [24] as presented in Table IV.

The idea of tinkering emerged since [31] introduced the concept of computational thinking in the 1980s. Tinkering is trying something new through exploration, trying repeatedly and making improvements. The problem-solving process involves thinking and tinkering to obtain the best solution [32]. The tinkering approach for adult learning implemented through exploring and building, which are carried out through trial leading to improve solutions [33]. Tinkering activities which are performed repeatedly can assist a novice in learning programming [34].

Creating refers to the planning, designing and evaluating, for example, a program [33]. Programming involves the process of developing algorithms in the form of flow charts or pseudo-codes and then followed by programs. One learning programming should undergo these steps and procedures. Therefore, the creating approach is in line with the learning of programming.

Debugging is a component of computational thinking by [23]. Debugging refers to the process of tracking and fixing errors [35] either an algorithm or a program [33]. However, debugging is usually related to improving programs because it involves syntax and semantics. This activity performed after testing programs as a programmer can identify the error and know how to fix it [36]. Novices need to expose with debugging approach to be on par with experienced programmers [37]. Therefore, students need to practice in debugging and evaluating programs while being monitored by the instructors [38].

Meanwhile, collaborative learning allows the process of knowledge acquisition, sharing, creation, and dissemination. Collaborating is one of the computational thinking approaches [24] to obtain the right solutions and motivate students to complete misleading assignments [33]. When students work together to solve problems or engage in activities, they also have the opportunity to apply new concepts they have learned, facilitates the application of concepts for the specific problem through exploration, critical thinking and analysis. Indirectly, collaborative learning can enhance assessment skills when group members use different approaches [39]. The collaborative approach is ideal for new programmers as they can build an understanding of problems, plan alternative solutions, learn with peers, build knowledge, and engage actively in programming learning [15]. Other than face-to-face collaboration approach in the classroom, this approach is also implemented through different mediums such as online learning systems [40], online training tools [41] and networks such learning management system (moodles) [42]. Therefore, students who learn programming course collaboratively can develop computational thinking skills as reported by [15]. Besides that, opportunities to get ideas from their peers and explain the knowledge gained to other friends can help students develop logical skills and increase their perseverance [33].

Programming is difficult and challenging to produce effective programs. In addition to problem-solving skills, using computational thinking skills as a solution strategy and mastering a programming language, programmers have to be resilient. Persevering is a computational thinking approach introduced by [24] defined as never giving up, determined, resilient and persistent. For example, educators play a role to avoid an environment that can cause students to give up or lose motivation by interacting with them and always give feedback to students if necessary [43]. Teaching strategy or teaching aids should be able to motivate students and help them to learn interestingly. There are teaching aids introduced by past researchers to motivate students to learn programming such as simulations, games, visualizations and robotics. However, these teaching aids focuses on learning the concepts of programming. Current studies concern about the strategy for problem solving as well as program development. Hence, we suggest the use of computational thinking to be implemented as teaching strategy since it offers the components consisted of skills, dimensions and approaches as listed in Tables I to III. As the skill components play the primary role in problem-solving, a detailed element is required to implement it. Hence, 35 elements were proposed representing abstraction, decomposition, pattern recognition, algorithm, logical reasoning and evaluation. These components and elements are potentially integrated as teaching strategy. We believed that when students are able to master in learning they will be more motivated to learn and educators are considered effective if they can help and motivate students in learning.

TABLE III. DESCRIPTION OF COMPUTATIONAL THINKING DIMENSIONS

Computational Thinking Skills	**Descriptions**
Computational concepts	The concept used by programmers during programming activities.
Computational practices.	Problem-solving in programming practice that focuses on thinking and learning processes.
Computational perspectives	Students' knowledge of themselves, their relationships with others, and the ability to use technology around them.

TABLE IV. DESCRIPTION OF COMPUTATIONAL THINKING APPROACHES

Approaches	**Descriptions**
Tinkering	Trying something new through exploration, experimentation, and improvement.
Creating	Creating is related to planning, designing, and evaluating something like programs and animations.
Debugging	The process of finding and identifying mistakes.
Collaborating	Work with others to ensure the best results.
Persevering	Never despair, determination, resilience, and perseverance.

III. Validation

This study used the Fuzzy Delphi method, a method improved from the Delphi method using Fuzzy theory. This method employed expert opinion and consensus to evaluate and validate each component and element of computational thinking for teaching and learning programming course as illustrated in Fig. 1. The verification process employed a focus group discussion involving 15 expert panels. Several steps were taken before validation using the Fuzzy Delphi method, to ensure that the components and elements are suitable for the problem-solving in a programming course and meet the needs of students. The processes involved were identifying the components of computational thinking for solving problems in programming, identification of components operational definitions, pre-evaluation of the operational definition, improvement of the operational definition and construction of elements for each component.

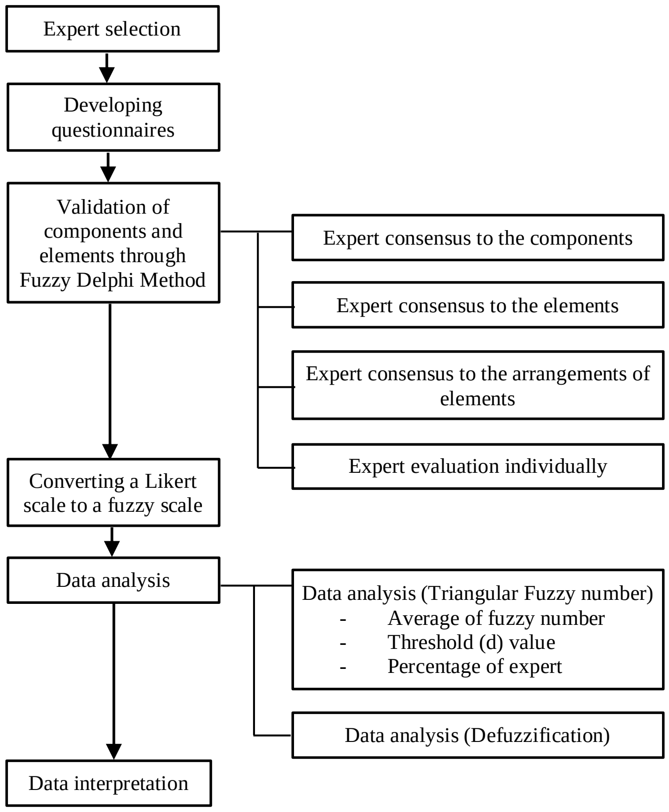

Expert selection
Developing questionnaires
Validation of components and elements through Fuzzy Delphi Method
Expert consensus to the components
Expert consensus to the elements
Expert consensus to the arrangements of elements
Expert evaluation individually
Converting a Likert scale to a fuzzy scale
Data analysis
Data analysis (Triangular Fuzzy number)
- Average of fuzzy number
- Threshold (d) value
- Percentage of expert
Data analysis (Defuzzification)
Data interpretation

Fig. 1. Validation Procedures of the Computational thinking Components.

A. *Expert Selection*

Experts in the field of study were selected for validation of computational thinking components using the Fuzzy Delphi method. There are several perspectives in determining the number of experts. According to the Delphi method, the number of experts should be between 10 to 50 people [44]. In this study, 15 experts in the programming field were selected as there was a uniformity among experts and is sufficient, according to [45]. The panel of experts consisted of lecturers from pre-university, public and private higher education institutions, vocational college, and teachers. All the selected experts have more than ten years of experience in teaching and learning programming. Instructors can also be considered as an expert if they have been in service for five to 10 years [44]. First, experts must give their consent to contribute their opinions within their expertise to evaluate and improve the proposed questionnaire that comprised of computational thinking components and elements for problem-solving in programming.

B. *Development of Questionnaires*

Based on the literature, 14 components of computational thinking that represented computational skills, dimensions and approaches as listed in Tables I to III were identified and characterised as 14 questionnaire items in this study. As the skill component plays a leading role in the problem-solving process or activity, the skill components are detailed with appropriate elements (Table V) to suit their use in the study context and included as 35 questionnaire items.

The questionnaire used a 7-point Likert scale representing strongly disagree to strongly agree.

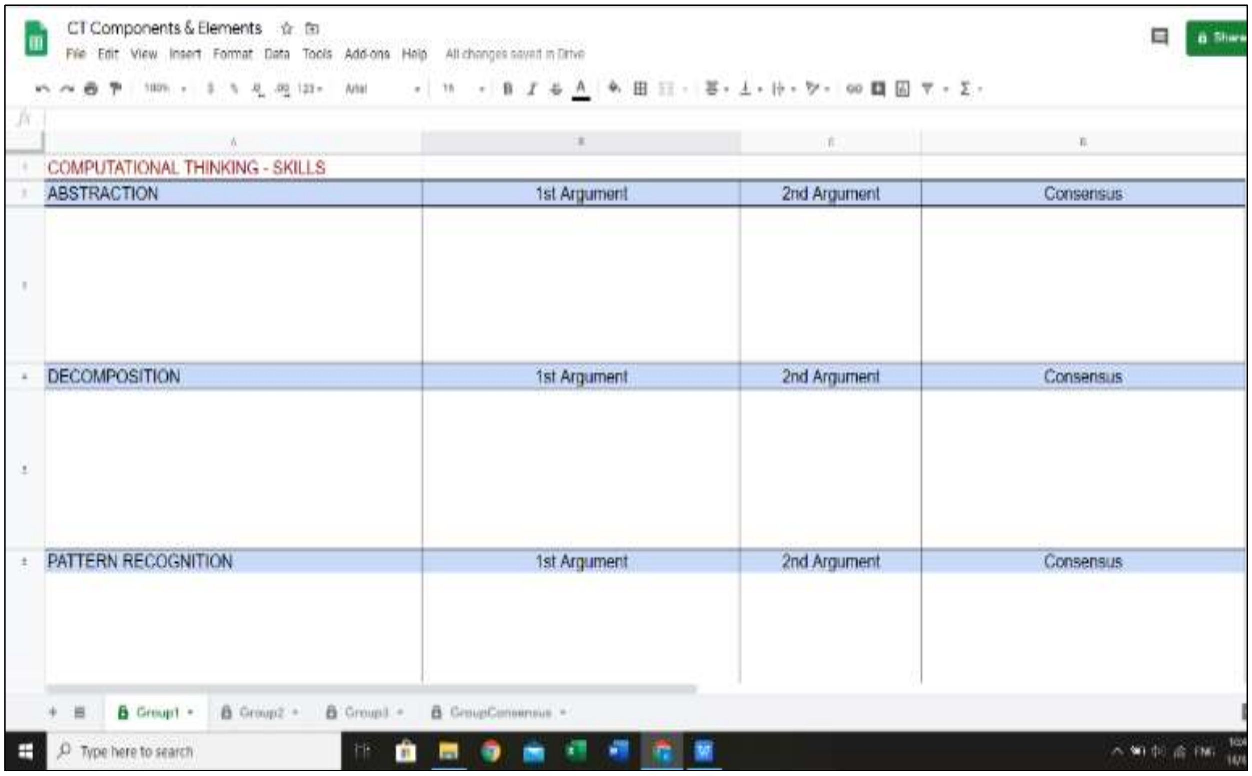
TABLE V. DESCRIPTION OF ELEMENTS FOR EACH COMPUTATIONAL THINKING (COMPUTATIONAL THINKING) SKILLS

Skill Components	**Elements Descriptions**
Abstraction	There are five (5) elements related to the process of understanding and formulating problems as well as identifying relevant information.
Decomposition	There are five (5) elements related to the process of decomposing the problem.
Pattern recognition	There are five (5) elements to integrate existing knowledge and experience as a problem-solving strategy.
Algorithm	There are seven (7) elements to develop an algorithm and the consequences if the algorithm is not perfect.
Logical reasoning	There are seven (7) elements related to logic in programming.
Evaluation	There are six (6) elements related to evaluation to ensure that solutions are accurate, appropriate and meet its purpose.

C. *Validation of Components and Elements using the Fuzzy Delphi Method*

The validation of components and elements referring to the questionnaire items were done using the Fuzzy Delphi method, where focus group discussions took place involving expert panels. There are several steps in a validation process including validation of principal components and elements and arrangement of elements based on expert opinions and consensus; evaluation of components and elements by experts individually; and finally data were collected and analysed using the Fuzzy Delphi technique. The details of the processes are explained as follows:

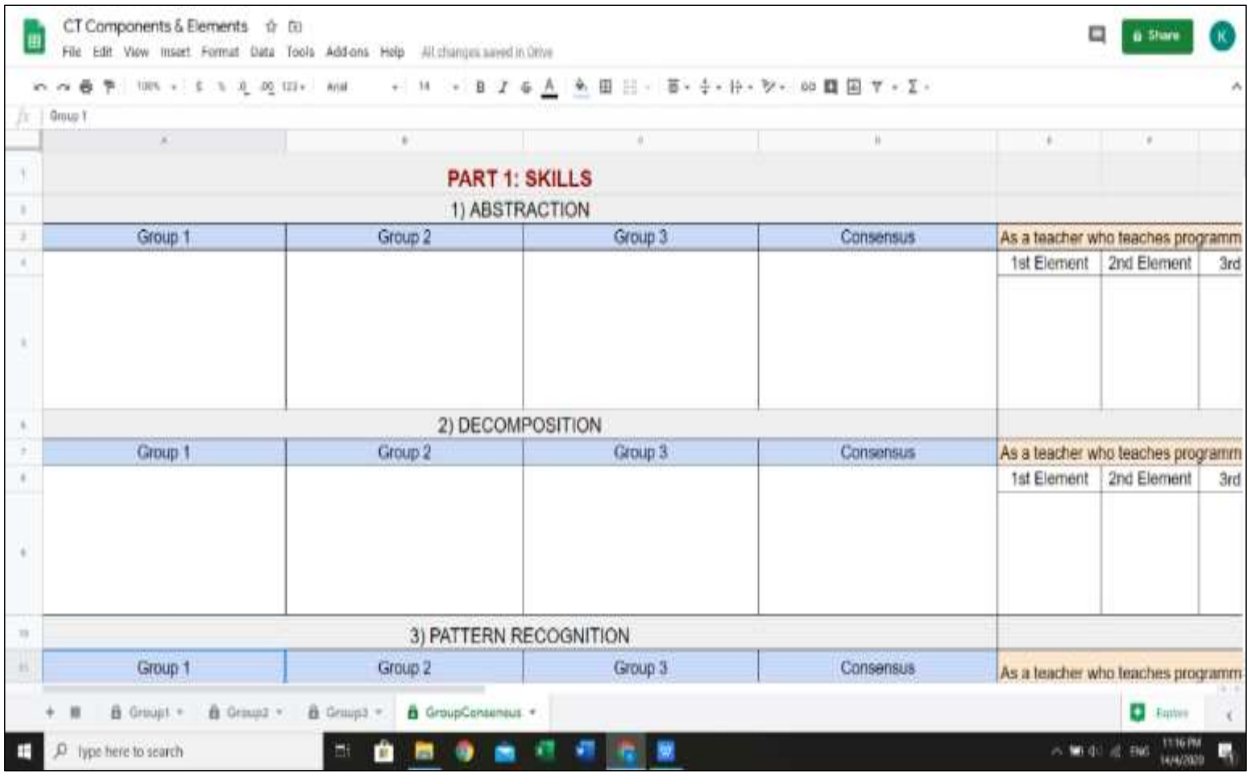
1) Expert consensus regarding main components: The components and elements of computational thinking were provided to the experts using Google sheets and shared via email a week before the discussion to provide them with research information, comfortable period to understand the context of the study and to generate ideas to improve the questionnaire. The statements and views were presented during the discussion. During the discussion, each expert was provided again with the details of computational thinking components. There were four worksheets used during the discussion. The first, second, and third worksheets were the tables for the first, second, and third groups, respectively as shown in Fig. 2 while the fourth column is for list of suggested elements for the component as shown in Fig. 3.



COMPUTATIONAL THINKING - SKILLS

ABSTRACTION	1st Argument	2nd Argument	Consensus
DECOMPOSITION	1st Argument	2nd Argument	Consensus
PATTERN RECOGNITION	1st Argument	2nd Argument	Consensus

Fig. 2. Google Drive Templates for each Group.



PART 1: SKILLS

1) ABSTRACTION

Group 1	Group 2	Group 3	Consensus	As a teacher who teaches programm		
				1st Element	2nd Element	3rd

2) DECOMPOSITION

Group 1	Group 2	Group 3	Consensus	As a teacher who teaches programm		
				1st Element	2nd Element	3rd

3) PATTERN RECOGNITION

Group 1	Group 2	Group 3	Consensus	As a teacher who teaches programm

Fig. 3. Google Drive Template for the Final Consensus.

During the discussion, experts were divided into three groups consisted of five people for each group. The component verification process was carried out in two stages. In the first stage, experts in each group evaluated and validated the components of the research based on the definitions provided. The opinions by experts were recorded in the Google Sheet document accordingly. The facilitator then transferred the consensus from each group to the fourth worksheet according to the group column. The second stage of the component verification process was a discussion for the consensus to evaluate and validate the components based on the suggestions from each group. The final consensus was filled in the fourth column. The validated components were used for the evaluation and validation of the proposed elements for the component. The focus group discussion procedure used is aimed at addressing the weakness of the iterative process identified when using the Delphi method (DM) [46], but at the same time retaining the features of Fuzzy Delphi method such as research time frame compared to DM. Fig. 1 and Fig. 2 shows the Google drive document for the validation process.

D. *Expert Consensus to the Arrangement of Elements*

After validating the skill components, the expert evaluated and validated the proposed element for each skill components. The list of elements was displayed next to the component, which was validated by the experts. The validation process involved a discussion among the experts to improve the suggested elements. Improvements included language structure to be clear and in line with the skills' definition and according to the context of the study; avoid repetitive, inappropriate, or unnecessary elements and suggest new elements as necessary to meet the skills' definition.

E. *Expert Consensus to the Arrangement of Elements According to Priority*

Questionnaire items for components and elements were transferred into Google forms for individual expert evaluation.

The use of Google forms allowed data to be transferred to Microsoft Excel and facilitated data analysis.

The questionnaire was then distributed to experts via email and Whatsapp using the Google form URL. The expert then answered the questionnaires individually to evaluate the components and elements by choosing the option on a 7-point Likert scale that represents strongly disagree to strongly agree. The answered questionnaire by all the experts through Google forms was saved directly in Google sheets. Fig. 3 shows the process of verifying components and elements.

The validated elements were then sorted in order of priority to fit the problem-solving approach in programming. The priority order considered the dimensions of their use during delivery, student engagement practically, development and evaluation of student performance. The arrangement process was performed together by all the experts.

F. *Expert Evaluation Individually*

Questionnaire items for components and elements were transferred into Google forms for individual expert evaluation. The use of Google forms allowed data to be transferred to Microsoft Excel and facilitated data analysis. The questionnaire was then distributed to experts via email and Whatsapp using the Google form URL. The expert then answered the questionnaires individually to evaluate the components and elements by choosing the option on a 7-point Likert scale that represents strongly disagree to strongly agree. The answered questionnaire by all the experts through Google forms was saved directly in Google sheets. Fig. 4 shows the process of verifying components and elements.

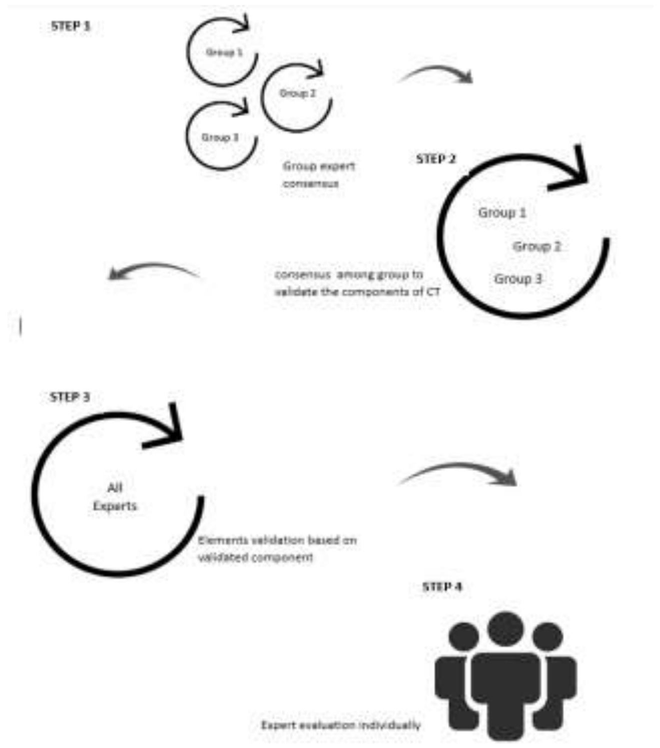

STEP 1
Group 1
Group 2
Group 3
Group expert consensus
STEP 2
Group 1
Group 2
Group 3
consensus among group to validate the components of CT
STEP 3
All Experts
Elements validation based on validated component
STEP 4
Expert evaluation individually

Fig. 4. Google Drive Template for the Final Consensus.

IV. Data Analysis

A. *Converting Likert Scale to Fuzzy Scale*

The Fuzzy scale was determined for each Likert scale, as shown in Table VI. Data in the Likert scale were converted to Fuzzy numbers through Microsoft Excel using the VLOOKUP function to be analysed using the Fuzzy Delphi (FDM) method. The Fuzzy set theory [47] allows the use of linguistic terms such as the level of agreement in Table VI by converting them to appropriate fuzzy sets and numbers. Respondents' answer on the Likert scale was translated into the Fuzzy scale, which was divided into three values: minimum value (m1), most reasonable value (m2) and maximum value (m3).

B. *Data Analysis using the Fuzzy Delphi Method*

In analysing the Fuzzy Delphi method, importance is given to the Triangular Fuzzy Number and the Defuzzification process. Both analyses aimed to determine whether a component or element is accepted or rejected based on the expert consensus [48]. Element acceptance was determined by the threshold (d) and per cent of consensus. The Defuzzification process aimed to obtain a Fuzzy score (A) to determine the acceptability of components and elements and its priority.

TABLE VI. QUESTIONNAIRE SCALE

Linguistic variables	Likert scale	Fuzzy scale (m_1, m_2,m_3)		
Extremely agree	7	0.9	1	1
Strongly agree	6	0.7	0.9	1
Agree	5	0.5	0.7	0.9
Moderately agree	4	0.3	0.5	0.7
Disagree	3	0.1	0.3	0.5
Strongly disagree	2	0	0.1	0.3
Extremely disagree	1	0	0	0.1

The Likert scale data from Google sheets were transferred into Microsoft Excel worksheet template to analyse the Fuzzy Delphi method by expert numbers (1 - 15). The Triangular Fuzzy Number composed of minimum(m1), reasonable (m2), and maximum (m3) values were used. Data analysis involved the determination of (i) the average value of the Fuzzy scale (m1, m2, m3), (ii) the threshold (d) value (iii) the percentages of consensus on each component and element, and (iv) the Fuzzy score to determine the acceptability and the ranking of components and elements using defuzzification process. Data were analysed using Microsoft Excel software.

1) Triangular fuzzy number: Average of Fuzzy Number (m1, m2, m3).

Fig. 5 shows a triangular graph against triangular values. All the values (m1, m2, m3) are in the range 0 to 1 which refers to the Fuzzy number (0,1).

The average value of a Fuzzy number was determined using the following Formula 1:

$$\frac{m=\sum_1^n mi}{n} \quad (1)$$

where n refers to the number of experts.

2) Triangular fuzzy number: Threshold (d) Value: The threshold value (d) was calculated to obtain the level of expert consensus for all questionnaire items [49]. Based on the Fuzzy numbering (0,1), the threshold value (d) for both Fuzzy numbers m (m1, m2, m3) and n = (n1, n2, n3) can be determined using the following Formula 2;

$$d(\tilde{m},\tilde{n}) = \sqrt{\frac{1}{3}[(m_1-n_1)^2+(m_2-n_2)^2+(m_3-n_3)^2]} \quad (2)$$

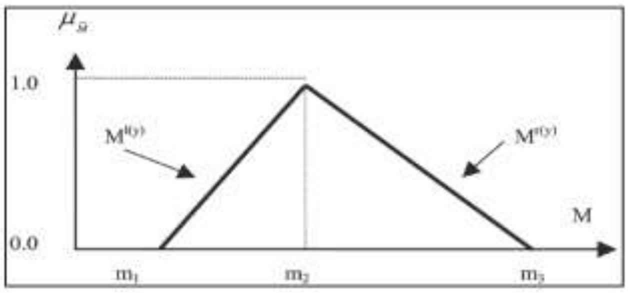


Fig. 5. Triangular Graph against Triangular Values.

If the distance between the mean value and the expert evaluation data is less than or equal to the threshold value (d) = 0.2, then all experts are considered to have reached an agreement [50]. Table VII shows the interpretation of the data based on the threshold value (d).

TABLE VII. INTERPRETATION OF THE DATA BASED ON THE THRESHOLD VALUE (D)

Threshold (d) value	Descriptions	Interpretation
$d \leqslant 0.2$	The threshold (d) value is less than or equal to 0.2	Accepted
$d \geqslant 0.2$	The threshold (d) value is greater than 0.2	Rejected OR conduct the second cycle, which involved only experts who disagreed.

To meet the conditions of acceptance for each item agreed upon by the experts, the percentage value of the expert's consensus must be equal to or greater than 75%. If the expert's consensus percentage is less than 75%, the item needs to be removed or a second round is conducted against the non-consenting expert.

C. *Defuzzification: The Fuzzy Score*

The Fuzzy score (A) obtained using the defuzzification process indicates whether an item is accepted based on the expert's consensus or not. An element is accepted when the Fuzzy score (A) equals or exceeds the median (α - cut) value of 0.5 [42]. The Fuzzy (A) score was calculated using the following Formula 3:

$$\text{Fuzzy score (A)} = (1/3) * (m1+m2+m3) \quad (3)$$

Apart from that, the Fuzzy score value (A) can determine the order and ranking of the questionnaire item. Since this study is about the problem-solving approach in programming, the arrangement of elements of each component based on the experts' discussion was followed. If the priority of the element is considered based on the defuzzification process, the results may not comply with the approach of problem-solving in programming. The expert will re-evaluate the order and priority of the element if any element is rejected after the analysis.

V. FINDINGS AND DISCUSSION

Focus group discussion was conducted, consisting of experts to evaluate and validate computational thinking components and elements. There are 14 main components of computational thinking and 35 elements representing computational thinking skill. All the components and elements evaluated by the experts were accepted. Based on a 7-point Likert scale, components and elements showed the average scores within 6 and 7 for all items, which strongly agree and extremely agree. For analysis, Likert scale scores were converted to Fuzzy scales. The results showed that all main components of computational thinking and skill elements of computational thinking met the first prerequisite of threshold (d) $\leqslant$ 0.2 based on the consensus of 15 experts. For the second prerequisite, the 14 components and 35 elements evaluated showed the percentage of consensus greater than

75%. Hundred percent consensus was achieved for the 14 components of computational thinking, while for the elements of computational thinking skill, the consensus are in the range from 86.6% to 100% was achieved. The third prerequisite was to obtain a Fuzzy score (A) to determine the acceptability of the questionnaire items. If the Fuzzy score (A) exceeds 0.5, then the questionnaire item is accepted. The Fuzzy score (A) in the range of 0.876 to 0.960 was obtained for all the components and elements evaluated. The findings confirmed the acceptance of all the components and elements of computational thinking tested and are suitable for teaching and learning programming.

Based on the analysis, FDM gives effective results in validating the components and elements of computational thinking. The results of the analysis in tandem with other research using FDM analysis for item-based validation [52-54]. The FDM analysis supported the suitability of the components and elements evaluated where all the questionnaires items were accepted, and pre-requisites met based on the threshold value (d), percentage of consensus and fuzzy score (A). The results of the analysis were influenced by the discussion method. All experts had the opportunity to give opinions and ideas to validate questionnaire through open discussion. Expert opinions were merged to get questionnaire items that fit the context of the study. The panel experts consisted of instructors who are curriculum developer and experienced programming lecturers where eight of them have followed trainees of trainers in computational thinking. The experts' evaluation was analysed to determine either the questionnaire items were accepted or rejected. The findings indicated similar responses from experts from the same institution for the items related to the program code. This similar view may be related to the common practices and approaches of teaching and learning used by the experts.

A. *FDM Analysis Effectively*

Generally, the focus group discussion method to get consensus is an effective method where researchers do not have to spend a lot of time to meet experts individually. Uploading materials in Google Drive and sharing with the experts involved in this study allowed retrieval of quick feedback from experts. Besides, this method provides an open discussion space and validated questionnaire items as a result of expert consensus can be updated online. However, expert group discussion requires a high level of commitment by the researchers and experts. The researcher should survey the available time of each expert, identify the appropriate date for all the experts to meet and discuss, and remind them through Google Calendar to ensure their attendance on the selected date. Besides, there are other preparations before the discussions such as preparation of expert invitation letters, printed materials and Google drives, and Google forms templates, a place with internet connections for discussions and refreshments for the experts.

The Fuzzy Delphi (FDM) method can avoid misinformation or loss of important information that can occur when using the Delphi method [36]. However, there are some limitations, even though FDM can give fast and reliable feedback. Researchers must have existing knowledge in the context of the study as relevant elements to be identified from literature review besides the need to communicate with the experts in the field of study who are willing to participate in the study.

The FDM method can be applied in other studies that require an expert's opinion and consensus. The FDM is not only suitable to validate components and elements as used in this study, but it can also be used to validate pre-construction to determine components during the analysis and evaluation phase which involved the development of models, modules, frameworks and products. The data obtained in quantitative form has higher reliability since it undergoes several qualitatively implemented stages. Analysis through FDM can determine the validity of computational thinking components and elements for problem-solving in programming based on expert consensus. The findings from this study can be used to develop problem-solving models in programming as a guide for teaching and learning.

VI. Conclusion

This study aimed to validate computational thinking components and elements as a problem-solving approach in programming by obtaining expert consensus using the Fuzzy Delphi method (FDM). Analysis results showed that all components and elements are accepted based on expert consensus. Hence, these components and elements potentially applied in teaching and learning programming as well as model development as teachers' guide. In the further work of this study, a model as a teachers' guide for teaching and learning programming will be developed by applying the accepted components and elements.

Acknowledgment

This study was funded by the UKM Research Grant (GUP-2018-155). We would like to thank the fifteen experts and Matriculation Division, Ministry of Education Malaysia for their assistance and contribution to this study.

References

[1] C. C. Selby, "Relationships: Computational Thinking, Pedagogy of Programming, and Bloom's Taxonomy," Proc. Work. Prim. Second. Comput. Educ., pp. 80–87, 2015.

[2] B. Du Boulay, "Some Difficulties of Learning to Program," J. Educ. Comput. Res., vol. 2, no. 1, pp. 57–73, 1986.

[3] Y. Qian and J. Lehman, "Students' Misconceptions and Other Difficulties in Introductory Programming," ACM Trans. Comput. Educ., vol. 18, no. 1, pp. 1–24, 2017.

[4] A. Robins, J. Rountree, and N. Rountree, "Learning and Teaching Prgramming : A Review and Discussion," Comput. Sci. Educ., vol. 13, no. 2, pp. 137–172, 2003.

[5] L. E. Winslow, "Programming Pedagogy --A Psychological Overview," ACM SIGCSE Bull., vol. 28, no. 3, pp. 17–22, 1996.

[6] F. Paas and P. Ayres, "Cognitive Load Theory: A Broader View on the Role of Memory in Learning and Education," Educ. Psychol. Rev., vol. 26, no. 2, pp. 191–195, 2014.

[7] V. J. Shute, C. Sub, and J. Asbell-Clarke, "Demystifying computational thinking," Educ. Res. Rev., vol. 22, no. September, pp. 142–158, 2017.

[8] S. Bocconi et al., Developing Computational Thinking : Approaches and Orientations in K-12 Education, no. June. 2016.

[9] A. Yadav, H. Hong, and C. Stephenson, "Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms," TechTrends, vol. 60, no. 6, pp. 565–568, 2016.

[10] A. D. F. Perez and G. M. Valladares, “Development and assessment of computational thinking: A methodological proposal and a support tool,” IEEE Glob. Eng. Educ. Conf. EDUCON, vol. 2018-April, pp. 787–795, 2018.

[11] A. R. Lopez and F. J. Garcia-Penalvo, “Personalized contents based on cognitive level of student’s computational thinking for learning basic competencies of programming using an environment b-learning,” Proc. Fourth Int. Conf. Technol. Ecosyst. Enhancing Multicult. - TEEM ’16, pp. 1139–1145, 2016.

[12] M. Roman-Gonzalez, J. C. Perez-Gonzalez, and C. Jimenez-Fernandez, “Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test,” Comput. Human Behav., vol. 72, pp. 678–691, 2017.

[13] M. Romero, A. Lepage, and B. Lille, “Computational thinking development through creative programming in higher education,” Int. J. Educ. Technol. High. Educ., vol. 14, no. 1, 2017.

[14] E. B. Witherspoon, R. M. Higashi, C. D. Schunn, E. C. Baehr, and R. Shoop, “Developing computational thinking through a virtual robotics programming curriculum,” ACM Trans. Comput. Educ., vol. 18, no. 1, pp. 1–20, 2017.

[15] B. Wu, Y. Hu, A. R. Ruis, and M. Wang, “Analysing computational thinking in collaborative programming: A quantitative ethnography approach,” J. Comput. Assist. Learn., no. January, pp. 1–14, 2019.

[16] F. K. Cansu and S. K. Cansu, “An Overview of Computational Thinking,” Int. J. Comput. Sci. Educ. Sch., vol. 3, no. 1, p. 17, 2019.

[17] A. Yadav, C. Stephenson, and H. Hong, “Computational thinking for teacher education,” Commun. ACM, vol. 60, no. 4, pp. 55–62, 2017.

[18] J. M. Wing, “Computational thinking,” Commun. ACM, vol. 49, no. 3, p. 33, 2006.

[19] J. M. Wing, Research Notebook: Computational Thinking-What and Why? The Link magazine of Carnegie Mellon University. 2011.

[20] P. J. Denning, “the Profession of It Beyond Computational thinking,” no. 6, pp. 5–7, 2009.

[21] A. Yadav, C. Mayfield, N. Zhou, S. Hambrusch, and J. T. Korb, “Computational Thinking in Elementary and Secondary Teacher Education,” ACM Trans. Comput. Educ., vol. 14, no. 1, pp. 1–16, 2014.

[22] P. J. Denning and M. Tedre, Computational Thinking. London, England: The MIT Press, 2019.

[23] C. Angeli et al., “A K-6 computational thinking curriculum framework: Implications for teacher knowledge,” Educ. Technol. Soc., vol. 19, no. 3, pp. 47–57, 2016.

[24] A. Csizmadia et al., “Computational thinking - A guide for teachers,” Comput. Sch., 2015.

[25] C. Selby and J. Woollard, “Computational Thinking: The Developing Definition,” Univ. Southampt., 2013.

[26] J. M. Wing, “Computational thinking and thinking about computing,” Philos. Trans. R. Soc. A Math. Phys. Eng. Sci., vol. 366, no. 1881, pp. 3717–3725, 2008.

[27] P. J. Denning, “Remaining trouble spots with computational thinking,” Commun. ACM, vol. 60, no. 6, pp. 33–39, 2017.

[28] B. Rodriguez, S. Kennicutt, C. Rader, and T. Camp, “Assessing computational thinking in CS unplugged activities,” Proc. Conf. Integr. Technol. into Comput. Sci. Educ. ITiCSE, pp. 501–506, 2017.

[29] R. Burbaite, V. Stuikys, and V. Drasute, “Integration of Computational Thinking Skills in STEM-Driven Computer Science Education,” IEEE Glob. Eng. Educ. Conf. EDUCON, pp. 1824–1832, 2018.

[30] K. Brennan and M. Resnick, “New frameworks for studying and assessing the development of computational thinking,” Proc. 2012 Annu. Meet. Am. Educ. Res. Assoc. Vancouver, Canada., vol. Vol. 1, pp. 1–25, 2012.

[31] S. Papert, Mindstorms: Children, computers and powerful ideas, vol. 1. 1980.

[32] F. Anna, Sabariah Sha’rif, W. Wong, and Muralindran Mariappan, “Computational Thinking and Tinkering : Exploration Study of Primary School Students’ in Robotic and Graphical Programming,” Int. J. Assess. Eval. Educ., vol. 7, no. 1993, pp. 44–54, 2017.

[33] CAS Barefoot, “Computational Thinking,” 2014.

[34] M. Berland, T. Martin, T. Benton, C. P. Smith, and D. Davis, “Journal of the Learning Using Learning Analytics to Understand the Learning Pathways of Novice Programmers,” no. January 2014, pp. 564–599, 2013.

[35] J. Krauss and K. Prottsman, Computational Thinking and Coding for Every Student: The Teacher’s Getting- Started Guide. Corwin Press: Sage Publishing Company, 2016.

[36] R. McCauley et al., “Debugging: A review of the literature from an educational perspective,” Comput. Sci. Educ., vol. 18, no. 2, pp. 67–92, 2008.

[37] B. Özmen and A. Altun, “Undergraduate Students’ Experiences in Programming: Difficulties and Obstacles Üniversite Öğrencilerinin Programlama Deneyimleri: Güçlükler ve Engeller,” Turkish Online J. Qual. Inq., vol. 5, no. 3, pp. 9–27, 2014.

[38] K. Kwon and J. Cheon, “Exploring problem decomposition and program development through block-based programs,” Int. J. Comput. Sci. Educ. Sch., vol. 3, no. 1, p. 3, 2019.

[39] M. Tom, “Five Cs framework: A student-centered approach for teaching programming courses to students with diverse disciplinary background,” J. Learn. Des., vol. 8, no. 1, pp. 21–37, 2015.

[40] M. Othman, N. M. Zain, U. H. Mazlan, and R. Zainordin, “Assessing cognitive enhancements in introductory programming through online collaborative learning system,” 2015 Int. Symp. Math. Sci. Comput. Res., vol. 2015, pp. 7–12, 2015.

[41] M. Karyotaki and A. Drigas, “Online and other ICT-based training tools for problem-solving skills,” Int. J. Emerg. Technol. Learn., vol. 11, no. 6, pp. 35–39, 2016.

[42] M. Tiantong and S. Teemuangsai, “The four scaffolding modules for collaborative problem-based learning through the computer network on moodle lms for the computer programming course,” Int. Educ. Stud., vol. 6, no. 5, pp. 47–55, 2013.

[43] A. Gomes and A. Mendes, “A teacher’s view about introductory programming teaching and learning: Difficulties, strategies and motivations,” Proc. - Front. Educ. Conf. FIE, vol. 2015-Febru, no. February, 2015.

[44] D. C. Berliner, “The Near Impossibility of Testing for Teacher Quality,” J. Teach. Educ., vol. 56, no. 3, pp. 205–213, 2005.

[45] M. Adler and E. Ziglio, Gazing into the oracle. Bristol, PA: Jessica Kingsley Publishers, 1996.

[46] N.Amira M.Saffie, Nur’Amirah Mohd Shukor, and Khairul A. Rasmani, “Fuzzy delphi method: Issues and challenges,” pp. 1–7, 2017.

[47] L. A. Zadeh, “Fuzzy set,” Inf. Control, vol. 8, pp. 338–353, 1965.

[48] Mohd Ridhuan Mohd Jamil, Saedah Siraj, Zaharah Hussin, Nurulrabihah Mat Noh, and Ahmad Ariffin Sapar, Pengenalan Asas Kaedah Fuzzy Delphi Dalam Penyelidikan Reka Bentuk dan Pembangunan. Minda Intelect, 2017.

[49] N. S. Thomaidis, N. Nikitakos, and G. D. Dounias, “The evaluation of information technology projects: A fuzzy multicriteria decision-making approach,” Int. J. Inf. Technol. Decis. Mak., vol. 5, no. 1, pp. 89–122, 2006.

[50] C.-H. Cheng and Y. Lin, “Evaluating the best main battle tank using fuzzy decision theory,” Eur. J. Oper. Res., vol. 142, pp. 174–186, 2002.