JPEE 실천공학교육논문지
JOURNAL OF PRACTICAL ENGINEERING EDUCATION

실천공학교육 관련

J. Pract. Eng. Educ. 13(1), 199-206, 2021

컴퓨팅 사고 시각화에 의한 정량적 코드 평가 방법 연구

A Study on Quantization of a Code Evaluation Method through Visualization of Computational Thinking

허 경*

경인교육대학교 컴퓨터교육과

Kyeong Hur*

Department of Computer Education, Gyeong-In National University of Education, Anyang 13910, Korea

[요 약]

일반대학에서 컴퓨팅 사고를 교육하는 과목은, 모든 재학생이 필수로 이수하는 과목으로 지정되는 추세이다. 이를 통해, 컴퓨팅 사고에 의한 SW 개발 과정을 학습한 학생들이 다양한 전공 분야에 실제 적용할 수 있는 SW 융합 역량을 강화하고자 하고 있다. 이에 따라, 학생들의 컴퓨팅 사고력을 평가하는 방법에 관한 연구가 진행되어 왔다. 기존 연구결과에서는 컴퓨팅 사고 개념 이해와 코드 작성 역량에 대한 평가를 모두 포함하고 있으나, 정성적인 평가 기준과 코드 단위로 세분화된 평가 기준으로 인해, 실제 적용하는 데 어려움이 있다. 이에 본 논문에서는 학생들이 추가로 제출하는 컴퓨팅 사고 과정을 시각화한 순서도 결과물과 일반적인 코드 결과물에 대해, 학생들의 컴퓨팅 사고력을 정량적으로 상대 평가할 수 있는 방법을 제안하였다. 이를 위해, 컴퓨팅 사고 과정을 시각화한 모델을 제안하고, 상대 평가식을 도출하였다. 그리고 제안한 평가방법의 이해를 돕기 위해, 기초적인 피지컬 컴퓨팅 순서도 결과물과 코드 결과물로부터 특정 학생의 평가 측정값들을 구한 사례를 제시하였다. 마지막으로 특정 학생의 결과물로부터 상대 평가 점수가 정량적으로 도출되는 사례를 제시하였다. 그리고 본 평가 방법을 적용한 강좌에서 학생 및 현장 교사들의 평가 방법 만족도 설문 결과를 바탕으로 제안한 정량적 상대평가 방법의 유효성을 분석하였다.

[Abstract]

The subjects that teach computational thinking in general universities are being designated as compulsory subjects for all enrolled students. Through this, students who have learned the software development process by computational thinking are trying to strengthen their SW convergence capabilities that can be applied to various major fields. Accordingly, research has been conducted on a method of evaluating students' computational thinking ability. Existing research results include both the understanding of the concept of computational thinking and the evaluation of the ability to write code, but it is difficult to apply it in practice due to the qualitative evaluation criteria and the evaluation criteria subdivided into code units. Therefore, in this paper, we proposed a method

http://dx.doi.org/10.14702/JPEE.2021.199

Received 23 March 2021; **Revised** 22 April 2021
Accepted 22 April 2021

***Corresponding Author**
E-mail: khur@ginue.ac.kr

that can quantitatively evaluate students' computational thinking ability relatively using the additional flow chart result of visualizing the computational thinking process and the general code result. To this end, a model that visualized the computational thinking process was proposed, and a relative evaluation equation was derived. In addition, in order to help understand the proposed evaluation method, a case of obtaining a specific student's evaluation measurement values from the result of the basic physical computing flow chart and the code result is presented. Finally, a case in which the relative evaluation score was quantitatively derived from the outcome of a specific student was presented. And, in the course to which this evaluation method was applied, the effectiveness of the proposed quantitative relative evaluation method was analyzed based on the results of the satisfaction survey of the evaluation method of students and field teachers.

Key Words: Computational thinking, Evaluation of computational thinking, Non-major undergraduates, Quantitative evaluation, SW education

I. 서 론

컴퓨팅 사고(Computational Thinking: CT)는 정답이 정해지지 않은 문제에 대한 해답을 SW를 통해 해결할 수 있도록 일반화하는 과정이다. 이러한 컴퓨팅 사고력은 추상화, 문제 분해, 패턴 인식 그리고 문제해결 알고리즘, 즉, 코드 설계 역량으로 구성된다[1-3]. 이러한 컴퓨팅 사고를 가르치는 과목에서, 학생들의 컴퓨팅 사고력을 평가하는 방법에 관한 연구가 진행되어 왔다. 학생들이 프로그래밍 입문 도구로서 블록 기반 프로그래밍 도구를 사용한 코드를 제출한 경우, 제출한 코드의 컴퓨팅 사고력을 평가할 수 있다. 블록기반 프로그래밍 코드의 우수성을 평가하는 기존 연구에서는 프로그래밍과 연관된 특정 카테고리의 블록 코드 사용 개수를 측정한 결과들을 제시하였다[4]. 이에 참고문헌 [4]에서는 기존 연구와 달리, 블록 단위 코드에서 나타나지 않는 프로그래밍 개념 이해도를 평가하기 위해, 블록 기반 코드가 내포한 프로그래밍 개념을 구조화한 평가 매트릭스를 제안하였다.

한편, 참고문헌 [5]에서는 컴퓨팅 사고를 이론 교육과 프로그래밍 실습 교육의 관점에서 평가하는 방법을 개발하였다. 학생의 이론 및 실습 학습 성취도를 평가하고, 이론과 실습 성취도 간의 연관성을 컴퓨팅 사고력 4개 요소와 8가지 항목으로 분석하였다. 이러한 기존 연구결과에서는 컴퓨팅 사고 개념 이해와 코드 작성 역량에 대한 평가를 모두 포함하고 있다. 그러나, 기존 연구에서는 학생들이 작성한 코드 결과물만 제출 받아 컴퓨팅 사고력을 평가하고 있다. 따라서, 정량적인 코드 단위의 기계적인 평가가 실시될 수밖에 없고, 정성적인 평가의 경우, 학생들의 코드와 컴퓨팅 사고 수준을 분석하는 데 많은 시간이 소요되는 단점이 있다. 결국, 다수 학생들의 컴퓨팅 사고력을 평가하는 데, 다수의 인력과 시간이 소요된다. 즉, 코드 결과물만 수집한 후, 정성적인 평가 기준과 코드 단위로 세분화된 평가 기준을 적용함에 따라, 기존 컴퓨팅 사고 평가 방법은 복잡도가 높아 실제 적용하는 데 어려움이 있다.

이러한 선행 연구를 분석한 결과에 기초하여, 본 논문에서는 컴퓨팅 사고력을 평가하는 방법을 간략화 하는 것에 초점을 맞추었다. 이에 본 논문에서 제안한 평가 방법에서는 학생들이 컴퓨팅 사고 과정을 시각화한 순서도 결과물을 추가로 제출한다. 이후, 학생들이 제출한 순서도 결과물과 코드 결과물에 대해, 컴퓨팅 사고력을 학생들 간 코드 비교평가 영역과 코드 내부 평가 부문으로 분류한다. 그리고 이 두 영역에 대해, 4개 하위 사고력 평가 영역(문제분할, 패턴인식, 추상화, 알고리즘 설계) 단위로 세부적인 상대 평가 기준들을 개발한 후 정량적인 평가를 실시한다. 본 논문에서 제안하는 방법은 학생들이 제출한 순서도 결과물로부터 시각화된 컴퓨팅 사고 과정에 대한 정보를 수집하고 이 정보에 따라 정량적인 평가를 실시함에 따라, 기존 평가 방법과 달리 복잡도를 낮출 수 있다.

2장에서는 컴퓨팅 사고 과정을 시각화한 모델과 상대 평가식을 도출하였다. 그리고 3장에서는 제안한 평가방법의 이해를 돕기 위해, 기초적인 피지컬 컴퓨팅 순서도 결과물과 코드 결과물로부터 특정 학생의 세부 평가 요소 측정값들을 구한 사례를 제시하였다[6-8]. 마지막으로 4장에서는 특정 학생이 제출한 결과물로부터 상대 평가 점수가 정량적으로 도출되는 과정을 설명하였고, 본 평가 방법을 적용한 강좌에서 학생 및 현장 교사들의 평가 방법 만족도 설문 결과를 통해 제안한 정량적 상대평가 방법의 유효성을 분석하였다. 5장에서는 결론을 통해, 제안한 평가 방법의 활용 영역과 추후 연구 과제에 대해 기술하였다.

II. 컴퓨팅 사고 시각화 모델과 정량적 상대 평가식

문제 해결에 필요한 SW를 개발하는 컴퓨팅 사고 과정을 구성하는 4가지 핵심 요소가 있다. 이들 중 첫번째로, 추상화 사고는 주어진 문제를 SW가 해결할 수 있는 다수 n개의 입력 X와 출력 Y의 함수, 즉, m번째 출력 Y값에 대해, $Y_m=F(X_1, \ldots X_n)$의 관계로 모델링하는 것이다. 이러한 추상화 사고는 주어진 문제를 SW가 해결할 수 있는 하나의 큰 입출력 함수(Function: F)로 정의하는 것에서부터 적용된다. 두번째로, 문제분해 사고는 추상화 사고를 통해 정의된 큰 문제(함수)를 작은 문제들(함수들)로 나누는 것이다. 세번째로, 패턴 인식 사고는 분해된 문제들을 해결하는 과정에서 반복되는 패턴을 찾아, 이미 정의한 함수를 재사용하는 것을 의미한다. 마지막으로 알고리즘 사고는 분해된 각 문제를 해결하는 순차적, 반복적 그리고 선택적 솔루션을 완성하는 것이다[1-3].

그림 1은 컴퓨팅 사고를 구성하는 4가지 사고 과정을 시각화하여 모델링한 것이다. 주어진 문제를 해결하는 데 필요한 하나의 큰 입출력 함수 문제가 추상화 사고를 통해 정의된다. 이후 1단계 추상화-문제분해 사고 과정에서 함수 F1-1부터 F1-n까지 n개의 입출력 함수가 정의된다. 여기서 각 F1-n 블록은 문제분해 수준 1에서 정의된 n개의 함수들을 나타내고, 조건부 프로세스를 통해 입력을 받고 출력하는 기능이 있어야 한다. 학생들이 F1-n 함수들의 기능을 정의하면 1단계 추상화-문제분해 사고에 따라, 1단계 문제 해결 과정을 시각화하여 표현하게 된다.

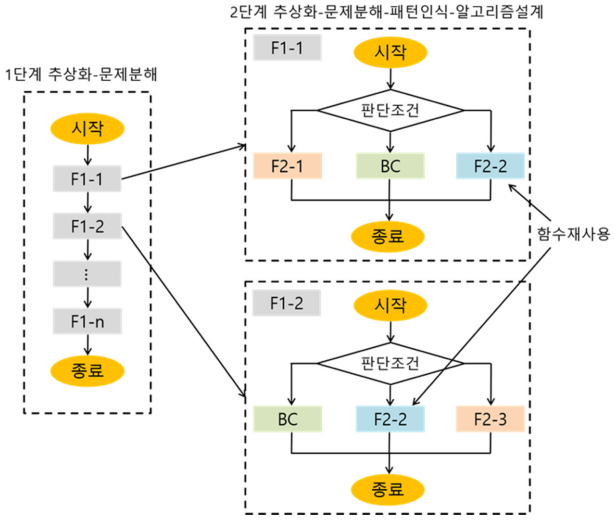

그림 1. 컴퓨팅 사고 시각화모델

Fig. 1. Computational Thinking Visualization Model.

2단계 추상화-문제분해-패턴인식-알고리즘설계 사고 과정에서는 1단계에서 정의된 함수 F1-1부터 F1-n에 대한 추상화-문제분해-패턴인식-알고리즘설계 사고 과정이 모두 발생한다. 그림 1에서 F1-1 함수가 입력되는 값에 따라 판단 조건을 거쳐, 솔루션이 완성되는 것을 볼 수 있다. 이 과정에서 문제 분해 수준 2단계의 함수 F2-1과 F2-2가 등장하는 것을 볼 수 있다. 여기서 각 F2-1, … F2-n 함수 블록들은 문제분해 수준 2에서 정의된 n개의 함수들을 나타내고, 조건부 프로세스를 통해 입력을 받고 출력하는 기능이 있어야 한다. 학생들이 F2-n 함수들의 기능을 정의하면 2단계 추상화-문제분해-패턴인식-알고리즘설계 사고에 따라, 2단계 문제 해결 과정을 시각화하여 표현하게 된다.

F1-1 함수의 실행 알고리즘을 설계하는 과정에 기본 코드(Basic Code: BC)가 등장하는 것을 볼 수 있다. 이 BC 블록은 특정 기능을 수행하는 함수 F로 표현할 필요가 없는 명령어 실행 코드 그룹 영역을 나타낸다. F1-2 함수의 실행 알고리즘을 설계하는 과정에서도 이러한 BC 블록이 있고, 하위 수준의 함수 F2-2과 F2-3가 등장하는 것을 볼 수 있다. 여기서 주목할 것은 F1-1 함수와 F1-2 함수의 실행 알고리즘을 설계하는 과정에서 함수 F2-2가 재사용(Reuse) 되고 있다는 것이다. 이는 패턴인식 사고가 시각화되어 표현된 것을 의미한다. 1단계에서 정의된 모든 함수 F1-1부터 F1-n에 대해 2단계 추상화-문제분해-패턴인식-알고리즘설계 사고 과정이 진행된다. 그림 1에서는 2단계 추상화-문제분해-패턴인식-알고리즘설계 사고 과정까지 한정하여 표현한 것이다. 실제 컴퓨팅 사고에서는 1단계에서 정의된 모든 함수 F1-1부터 F1-n에 대해 임의의 K단계 추상화-문제분해-패턴인식-알고리즘설계 사고 과정이 발생한다. 이러한 현상을 본 논문에서는 컴퓨팅 사고 과정에서 문해 분해 깊이 수 K가 있다고 정의한다.

본 논문에서 제안한 컴퓨팅 사고 평가 방법에서는 학생들이 컴퓨팅 사고 과정을 시각화한 순서도 결과물을 추가로 제출한다. 이후, 학생들이 제출한 순서도 결과물과 코드 결과물에 대해, 학생들 간 코드 비교 평가와 코드 내부 평가를 실시한다. 그리고 이 두 평가 영역에 대해, 4개 하위 사고력 평가 영역(문제분할, 패턴인식, 추상화, 알고리즘 설계) 단위로 세부적인 상대 정량평가 지표들을 정의하였다. 코드 비교 평가 대영역에 대한 세부 평가 영역으로 Flowchart 알고리즘 설계와 코드설계, 두 하위 영역들을 정의하였고, 코드 내부 평가 대영역에 대해서는 Flowchart 알고리즘-코드 일치도와 코드 오류, 두 하위 영역들을 정의하였다. 표 1은 컴퓨팅 사고 과정을 시각화한 순서도를 활용한 정량적 상대 평가식을 각 평가 영역별로 정리한 것이다.

표 1. 컴퓨팅 사고 과정을 시각화한 순서도를 활용한 정량적 상대 평가식

Table 1. Quantitative relative evaluation formula using a flow chart that visualizes the computational thinking process

평가영역	세부 평가영역	상대 평가식 (각 가중치 Weight 합 = 100)
코드 비교 평가	Flowchart 알고리즘설계	• 알고리즘의 문제분해깊이수 점수(P_NAD: Point of Number of Algorithm Depth) P_NAD = Weight_NAD * (NAD/MAX_NAD) (MAX_NAD: 학생 그룹내 최대 NAD, NAD: 해당 학생의 NAD)
		• 알고리즘의 함수재사용회수 점수(P_NAR: Point of Number of Algorithm Reuse) P_NAR = Weight_NAR * (NAR/MAX_NAR) (MAX_NAR: 학생 그룹내 최대 NAR, NAR: 해당 학생의 NAR)
		• 알고리즘의 함수개수 점수(P_NAF: Point of Number of Algorithm Function) P_NAF = Weight_NAF * (NAF/MAX_NAF) (MAX_NAF: 학생 그룹내 최대 NAF, NAF: 해당 학생의 NAF)
	코드설계	• 코드의 문제분해깊이수 점수(P_NCD: Point of Number of Code Depth) P_NCD = Weight_NCD * (NCD/MAX_NCD) (MAX_NCD: 학생 그룹내 최대 NCD, NCD: 해당 학생의 NCD)
		• 코드의 함수재사용회수 점수(P_NCR: Point of Number of Code Reuse) P_NCR = Weight_NCR * (NCR/MAX_NCR) (MAX_NCR: 학생 그룹내 최대 NCR, NCR: 해당 학생의 NCR)
		• 코드의 함수개수 점수(P_NCF: Point of Number of Code Function) P_NCF = Weight_NCF * (NCF/MAX_NCF) (MAX_NCF: 학생 그룹내 최대 NCF, NCF: 해당 학생의 NCF)
코드 내부 평가	Flowchart 알고리즘-코드일치도	• 문제분해깊이수 일치도 점수(SP_ND: Similarity Point of Number of Depth) SP_ND = 1−|NAD−NCD|/Max(NAD, NCD)
		• 함수재사용회수 일치도 점수(SP_NR: Similarity Point of Number of Reuse) SP_NR = 1−|NAR−NCR|/Max(NAR, NCR, 1)
		• 함수개수 일치도 점수(SP_NF: Similarity Point of Number of Function) SP_NF = 1−|NAF−NCF|/Max(NAF, NCF, 1)
	코드오류	• 코드 동작 점수(P_CO: Point of Code Operation) P_CO = Weight_CO*P_P/F (P_P/F = 1 or 0, IF PASS: 1, IF FAIL: 0)

III. Flowchart 및 코드 결과물로부터 평가 측정값 도출 사례

본 논문에서 제안한 컴퓨팅 사고 평가 방법에서는 학생들이 컴퓨팅 사고 과정을 시각화한 순서도 결과물을 추가로 제출한다. 3장에서는 제안한 평가방법의 이해를 돕기 위해, 기초적인 피지컬 컴퓨팅 순서도 결과물과 코드 결과물로부터 특정 학생의 세부 평가 요소 측정값들을 구한 사례를 제시한다.

추상화 사고는 주어진 문제를 SW가 해결할 수 있는 하나의 큰 입출력 함수로 정의하는 것에서부터 적용된다. 그림 2는 이러한 결과를 나타낸다. 즉, 개발하고자 하는 피지컬 컴퓨팅 장치에 임베드되는 SW를 '입력되는 진동 가속도 정보

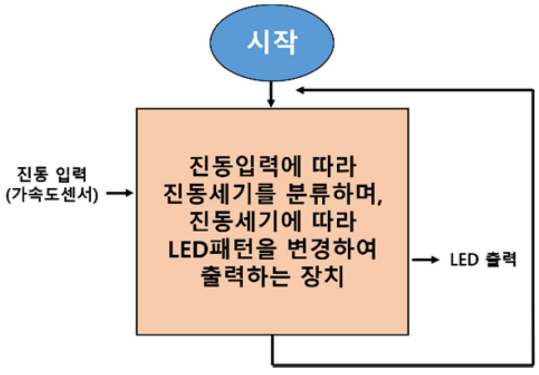

그림 2. 첫번째 추상화 사고가 시각화된 순서도

Fig. 2. Flowchart with visualization of the first abstract thinking.

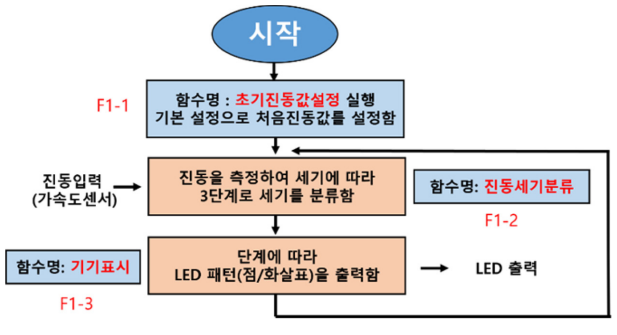

그림 3. 1단계 추상화와 문제분할 사고가 시각화된 순서도

Fig. 3. Flowchart that visualizes the first-stage abstraction and problem division thinking.

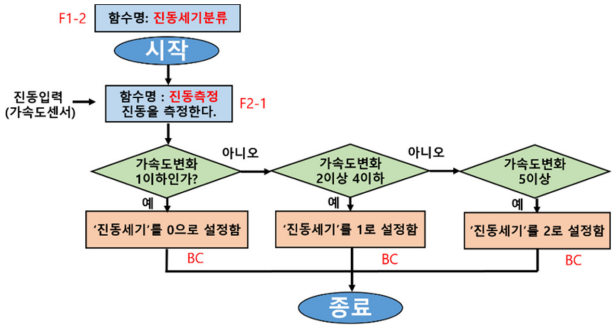

그림 4. 2단계 추상화, 문제분할과 알고리즘 사고가 시각화된 순서도(F1-2)

Fig. 4. Flowchart that visualizes the second-stage abstraction and problem division thinking(F1-2).

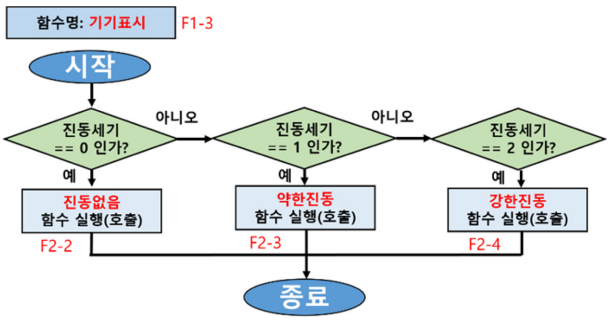

그림 5. 2단계 추상화, 문제분할과 알고리즘 사고가 시각화된 순서도(F1-3)

Fig. 5. Flowchart that visualizes the second-stage abstraction and problem division thinking(F1-3).

에 따라, 진동 세기를 분류하고 LED 패턴을 출력하는 무한 반복하는 입출력 함수' 하나로 정의하고 있다. 그림 3에서는 1단계 추상화-문제분해 사고 과정에서 '초기진동값설정' 함수(F1-1), '진동세기분류' 함수(F1-2) 그리고 '기기표시' 함수(F1-3) 3개의 입출력 함수가 정의된 것을 볼 수 있다.

그림 4에서는 2단계 추상화-문제분해-패턴인식-알고리즘설계 사고 과정에서 1단계에서 정의된 함수 F1-2에 대한 추상화-문제분해-알고리즘설계 사고 과정이 모두 발생한다. 그림 4에서 F1-2 함수가 입력되는 값에 따라 판단 조건을 거쳐, 솔루션이 완성되는 것을 볼 수 있다. 이 과정에서 문제 분해 수준 2단계의 함수 F2-1와 기본 명령어 코드(Basic Code: BC)가 등장하는 것을 볼 수 있다. 이 BC 블록은 특정 기능을 수행하는 함수 F로 표현할 필요가 없는 명령어 실행 코드 그룹 영역을 나타낸다. 그림 5의 F1-3 함수의 실행 알고리즘을 설계하는 과정에서는 하위 수준의 함수 F2-2, F2-3 그리고 F2-4가 등장하는 것을 볼 수 있다. 그림 6에서는 3단계 추상화, 문제분할과 알고리즘 사고를 통해, 함수 F2-2, F2-3 그리고 F2-4에 대한 실행 알고리즘이 설계되어 있고, BC 블록들만 설계된 것을 볼 수 있다.

한편, 그림 3부터 그림 6까지 예시된 피지컬 컴퓨팅 순서도에서는 특정 함수가 재사용되는 패턴인식 사고는 발생하지 않은 것을 알 수 있다. 표 2는 그림 3부터 그림 6까지 예시된 피지컬 컴퓨팅 순서도에 따라 작성된 코드 사례이다. 본 코드 예제는 순서도 예제에 등장하는 내용이 동일하게 구현된 사례로 제시된 것이다. 표 3은 그림 3부터 그림 7, 표 2까지 제시된 피지컬 컴퓨팅 순서도 및 코드 예제로부터, 표 1 '컴퓨팅 사고 과정을 시각화한 순서도를 활용한 정량적 상대 평가식'에서 요구하는 각 세부 측정 요소들을 도출한 결과를 나타낸다.

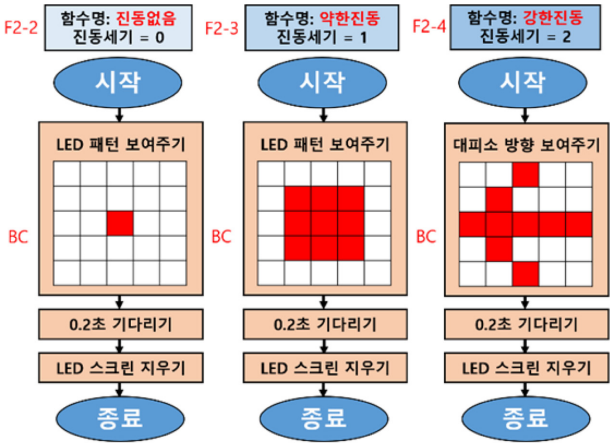

그림 6. 3단계 추상화, 문제분할과 알고리즘 사고가 시각화된 순서도(F2-n)

Fig. 6. Flowchart that visualizes the third-stage abstraction and problem division thinking(F2-n).

IV. 상대 평가 점수 정량적 도출 사례와 평가방법 유효성 분석

본 장에서는 특정 학생이 제출한 순서도와 코드 결과물로부터 상대 평가 점수가 정량적으로 도출되는 사례를 설명한다. 학생이 제출한 순서도에는 코드를 작성하는 과정에 발생한 컴퓨팅 사고 과정이 시각화되어 표현되어 있다. 3장에서는 기초적인 컴퓨팅 순서도 및 코드 예제를 통해, 표 1 '컴퓨팅 사고 과정을 시각화한 순서도를 활용한 정량적 상대 평가식'에 필요한 세부 요소 값들이 측정되는 과정을 설명하였고, 그 결과가 표 3으로 제시되었다. 표 4는 특정 학생 결과물로부터 정량적 상대 평가 점수가 도출된 결과를 설명하고 있

표 2. 피지컬 컴퓨팅 순서도에 따라 작성된 코드 사례

Table 2. Example of code written according to the physical computing flow chart

Javascript code

```
function 초기진동설정 () {
    basic.showString("s")
    현재가속도 = input.acceleration(Dimension.X)
}
function 약한진동 () {
    basic.showLeds(`
        . . . . .
        . # # # .
        . # # # .
        . # # # .
        . . . . .
        `)
    basic.pause(200)
    basic.clearScreen()
}
function 진동세기분류 () {
    진동측정()
    if (가속도변화 <= 1) {
        진동세기 = 0
    } else if (2 <= 가속도변화 && 가속도변화 <= 4) {
        진동세기 = 1
    } else {
        진동세기 = 2
    }
}
function 강한진동 () {
    basic.showLeds(`
        . . # . .
        . # . . .
        # # # # #
        . # . . .
        . . # . .
        `)
    basic.pause(200)
    basic.clearScreen()
}
function 진동측정 () {
    가속도변화 = Math.round(Math.abs(input.
    acceleration(Dimension.X) - 현재가속도) / 30)
}
function 기기표시 () {
    if (진동세기 == 0) {
        진동없음()
    } else if (진동세기 == 1) {
        약한진동()
    } else {
        강한진동()
    }
}
function 진동없음 () {
    basic.showLeds(`
        . . . . .
        . . . . .
        . . # . .
        . . . . .
        . . . . .
        `)
    basic.pause(200)
    basic.clearScreen()
}
let 진동세기 = 0
let 가속도변화 = 0
let 현재가속도 = 0
초기진동설정()
basic.forever(function () {
    진동세기분류()
    기기표시()
})
```

표 3. 컴퓨팅 순서도 및 코드 예제로부터 측정된 세부 요소 값들

Table 3. Detailed element values measured from computational flow charts and code examples

평가영역	세부평가 영역	세부 요소 항목	측정된 정량값
코드 비교평가	Flowchart 알고리즘 설계	• 알고리즘의 문제분해깊이수 NAD	2
		• 알고리즘의 함수재사용회수 NAR	0
		• 알고리즘의 함수개수 NAF	7
	코드설계	• 코드의 문제분해깊이수 NCD	2
		• 코드의 함수재사용회수 NCR	0
		• 코드의 함수개수 NCF	7

다. 표 3에 제시된 측정값들은 표 4의 상대 평가 점수 도출 결과표에서 '특정 학생 결과물의 해당 수치 값' 10 개 필드 값들을 구하는 데 사용된다. 모든 학생들의 결과물로부터 해당 수치 10 개 필드 값들이 수집되면, 표 4의 그룹 내 최대값 또는 이상치(Max val.)의 NAD부터 NCF까지 6개 필드값을 구할 수 있고, 이에 해당하는 값들을 표 4에 예시하였다.

다음으로 표 1에서 설명한 각 세부 평가 항목에 대한 가중치를 설정한다. 표 4에서는 10개 항목에 대해 모두 동일하게 가중치 10을 적용하였다. 모든 가중치 합은 100이 되도록 설정한다. 이후 표 1에서 설명한 각 세부 평가 항목별 상대 평가식에 따라, 코드 비교 평가 영역에 대한 해당 점수가 산출된다. 그리고 코드 내부 평가 영역에 대한 해당 점수도 상대적으로 산출된다. 10개 세부 평가 항목은 표 4와 같이, 추상화-문제분해-패턴인식-알고리즘설계 사고 영역을 반영하여 설계되었다.

본 논문에서 제안하는 평가 방법은 특정 문제를 해결하기 위해 설계하는 SW의 이상적인 답을 알 수 없어도 평가 가능하다는 장점이 있다. 개발된 SW의 이상적인 구조를 미리 예상할 수 있는 경우는, 수학적인 문제의 해결 또는 정렬, 탐색 그리고 최단 경로 등과 같은 전통적인 정답 알고리즘이 존재하는 경우이다. 그 외의 SW 개발 과제에서는 다양한 구조의 SW를 학생들이 과제물로 제출할 수 있다. 이러한 상황에서 교수자가 학생들의 SW개발 결과물을 평가하려면, 현실적으로 상대 평가 방식이 해결책일 수 있다. 표 4에서 특정 학생의 SW 결과물은 컴퓨팅 사고력 관점에서 100점 만점에 74.19 점을 받은 것이다. 여기서 중요한 것은 학생의 코드가 오류 없이 올바르게 동작함에도 불구하고 74.19점을 받은 것이다.

본 평가 방법을 적용한 강좌에서 학생 및 현장 교사들의 평가 방법 만족도 설문 결과를 통해 제안한 정량적 상대평가

표 4. 특정 학생 결과물로부터 정량적 상대 평가 점수 도출 결과

Table 4. Results of deriving quantitative relative evaluation scores from specific student outcomes

평가영역	세부 평가영역	상대평가 측정항목	그룹 내 최대값 또는 이상치 Max val.	가중치 W	특정 학생 결과물의 해당수치값	해당평가 점수	CT 평가영역
코드비교 평가	Flowchart 알고리즘설계	문제분해깊이수(NAD)	3	10	2	6.67	문제분해
		함수재사용회수(NAR)	2	10	1	5.00	패턴인식
		함수개수(NAF)	7	10	4	5.71	추상화
	코드설계	문제분해단계수(NCD)	3	10	2	6.67	문제분해
		함수재사용회수(NCR)	2	10	2	10.00	패턴인식
		함수개수(NCF)	7	10	5	7.14	추상화
코드내부 평가	Flowchart 알고리즘-코드 일치도	1-|NAD-NCD|/Max(NAD, NCD)	1	10	1	10	알고리즘 자동화
		1-|NAR-NCR|/Max(NAR, NCR, 1)	1	10	0.5	5	
		1-|NAF-NCF|/Max(NAF, NCF, 1)	1	10	0.8	8	
	코드 오류	PASS/FAIL	1	10	1	10	
합계				100		74.19	

방법의 유효성을 분석하였다. 만족도 평가 설문 결과를 통해 직접적으로 정량적 상대평가 방법의 유효성을 평가받기 위해, 평가 방법 설계 과정과 직결되는 만족도 평가 항목들을 선정하였다. 이에 따라 선정한 만족도 평가 항목으로 CT 시각화 순서도 작성 필요성과 용이성, 코드비교평가 6개 항목의 타당성, 코드내부평가 4개 항목의 타당성, 전체 정량적 점수 산출의 용이성과 타당성이라는 6개 항목을 조사하였다.

표 5와 표 6은 컴퓨팅 사고 시각화에 따른 정량적 상대 평가 방법의 만족도 결과를 나타낸다. 표 5는 20년도 1, 2학기 50명 비전공자 학부생 대상 만족도 조사 결과이며, 표 6은 동일 학기 50명 현직교사 비전공자 교육대학원생 대상 만족도 결과이다. 비전공자 학부생들과 현직 교사 모두 6개 조사 항목에 대해, 부정적인 응답(매우 그렇지 않다, 그렇지 않다) 보다 긍정적인 응답(그렇다, 매우 그렇다)이 높은 비율로 나타

표 5. 학부생 컴퓨팅 사고 정량적 상대 평가 만족도 조사 결과(50명)

Table 5. Results of satisfaction survey for non-major undergraduate students

만족도 평가 항목	매우 그렇지 않다	그렇지 않다	보통	그렇다	매우 그렇다
CT 시각화 순서도 작성 필요성	0%	0%	20%	30%	50%
CT 시각화 순서도 작성 용이성	10%	20%	20%	30%	20%
코드비교평가 6개 항목의 타당성	0%	20%	30%	30%	20%
코드내부평가 4개 항목의 타당성	0%	10%	20%	40%	30%
전체 정량적 점수 산출의 용이성	0%	10%	20%	40%	30%
전체 정량적 점수 산출의 타당성	10%	10%	20%	30%	30%

표 6. 현직교사 컴퓨팅 사고 정량적 상대 평가 만족도 조사 결과(50명)

Table 6. Satisfaction Survey Results for Current Teachers of Non-majors

만족도 평가 항목	매우 그렇지 않다	그렇지 않다	보통	그렇다	매우 그렇다
CT 시각화 순서도 작성 필요성	0%	0%	10%	40%	50%
CT 시각화 순서도 작성 용이성	10%	10%	20%	40%	20%
코드비교평가 6개 항목의 타당성	0%	10%	20%	40%	30%
코드내부평가 4개 항목의 타당성	0%	0%	20%	40%	40%
전체 정량적 점수 산출의 용이성	0%	0%	20%	50%	30%
전체 정량적 점수 산출의 타당성	0%	10%	20%	40%	30%

났다. 또한, SW 교육을 학교에서 실시하며, 컴퓨팅 사고 교육의 필요성과 난이도를 체감하고 있는 현직 교사들이 학부생들보다 높은 비율로 긍정적으로 평가하였다. 본 조사 결과는 답변 경향을 분석하기 위해, 50명 표본집단으로부터 5명 단위로 도수를 정리한 것이다. 본 만족도 조사 결과와 표 4의 점수 산출 결과를 통해, 제안한 평가 방법의 일반적 유효성과 적용 가능성은 높은 것으로 분석되었다.

V. 결 론

제안한 컴퓨팅 사고(CT) 시각화에 따른 정량적 상대 평가 방법은, 다양한 구조의 SW를 학생들이 과제물로 제출할 수 있는 SW교육 커리큘럼에서 보다 효과적으로 적용할 수 있다. 또한, 컴퓨팅 사고를 시각화한 순서도를 작성하는 실습이 필요하며, 입력이 1개이고 출력이 1개인 형태에서부터 입력 개수와 출력 개수를 하나씩 늘려가며 보다 복잡한 CT 시각화 표현 훈련이 필요하다. 이러한 과정을 통해, CT에 대한 이해도를 높일 수 있으며, 코드 설계 역량을 강화할 수 있다. 추후 연구로는 비전공자 대상 교육에서 다양한 구조의 SW를 학생들이 과제물로 제출할 수 있는 피지컬 컴퓨팅 커리큘럼을 제안하고자 한다. 제안한 커리큘럼에서는 다양한 입출력 개수를 갖는 시스템을 단계적으로 설계하며, 변화하는 CT 구조를 시각화하여 표현하는 교육 훈련과 코드 설계 실습이 실시된다. 그리고 제안한 커리큘럼에서 설계되는 CT 구조의 수치적인 변화를 NAD, NAR 및 NAF 관점에서 분석함으로써, 특정 SW교육 커리큘럼에서 교육하는 CT 구조 변화 특성을 측정하여 나타낼 수 있는 정량적 평가 지표를 연구하고자 한다. 또한, 추후 연구에서는 학생들의 평균적인 NAD, NAR 및 NAF 관점의 CT 사고력 수준으로부터 개별 학생들의 CT 사고력을 정량 평가하는 방법을 연구하고자 한다. 한편, 기존 문헌에는 CT 사고력을 문제분해, 패턴인식, 추상화 및 알고리즘 관점에 시각화하여 정량적으로 평가한 사례가 없어, 비교 분석을 미실시하였다.

참고문헌

[1] S. O. Yang, “Necessity of computational thinking,” *Korea Information Processing Society Review*, vol. 24, no. 2, pp. 4-12, March 2017.

[2] J. M. Wing, “Computational thinking,” *Communications of the ACM*, vol. 49, no. 3, pp. 33-35, March 2006.

[3] J. M. Wing, “Computational thinking and thinking about computing,” *Philosophical Transactions of the Royal So Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 366, no. 1881, pp. 3717-3725, October 2008.

[4] W. S. Sohn, “A method for measuring and evaluating for block-based programming code,” *Journal of the Korean Association of Information Education*, vol. 20, no. 3, pp. 293-302, June 2016.

[5] J. K. Kim, “Development of rubric for assessing computational thinking concepts and programming ability,” *Journal of Korean Association of Computer Education*, vol. 20, no. 6, pp. 27-36, November 2017.

[6] K. Hur, “Educational method of computational thinking processes using physical teaching devices,” *Journal of Practical Engineering Education*, vol. 10, no. 1, pp. 35-39, June 2018.

[7] K. Hur, “An education method of computational thinking using microbit in a Java-based SW lecture for non-major undergraduates,” *Journal of Practical Engineering Education*, vol. 11, no. 2, pp.167-174, December 2019.

[8] K. Hur, “An education method of Java SW designs for IoT wireless device control using microbits,” *Journal of Practical Engineering Education*, vol. 12, no. 1, pp. 85-91, June 2020.

허 경 (Kyeong Hur)_종신회원

1998년 : 고려대 전자공학과 학사
2000년 : 고려대 전자공학과 석사
2004년 8월 : 고려대 전자공학과 통신공학박사
2004년 8월 ~ 2005년 8월 : 삼성종합기술원(SAIT) 전문연구원
2005년 9월 ~ 현재 : 경인교대 컴퓨터교육과 교수
〈관심분야〉 컴퓨터 네트워크 QoS, 센서네트워크, 무선 MAC, 피지컬 컴퓨팅 SW교육, AI교육