컴퓨팅 사고 개념 학습과 프로그래밍 역량 평가를 위한 루브릭 개발

김재경†

요 약

오늘날 컴퓨팅 사고 과목은 초·중등 교육과정뿐만 아니라 고등 교육과정에서도 필수 영역으로 개설하고 있다. 특히 고등 교육과정에서는 학습자가 컴퓨팅 사고의 개념을 전공 분야에 실제로 적용할 수 있도록 교육하여 융합 인재의 기반을 마련하는 것이 중요하다. 그러나 이와 같은 목표에 맞는 교육이 이루어지고 있는지를 평가할 수 있는 도구와 관련 연구가 부족하다. 이에 본 연구에서는 비전공자를 대상으로 한 컴퓨팅 사고 과목을 개념 교육과 프로그래밍 실습 교육의 두 가지 관점으로 평가하는 루브릭을 개발하고 학습 성취도를 평가하고 이론과 실습간의 연관성을 분석하였다. 제안 루브릭은 고등 교육과정의 컴퓨팅 사고 개념 및 실습 학습을 평가하기 위하여 컴퓨팅 사고의 세부 요소를 4개의 분류 및 8가지의 항목으로 구성한 두 개의 루브릭을 정의하였다. 또한 제안 루브릭을 대학교 컴퓨팅 사고 관련 교양 수업에 적용하여 학습도를 평가하여 결과를 분석하고 향후 개선점을 도출하였다.

주제어 : 컴퓨팅 사고, 루브릭, 고등 교육, 프로그래밍

Development of Rubric for Assessing Computational Thinking Concepts and Programming Ability

Jae-Kyung Kim†

ABSTRACT

Today, a computational thinking course is being introduced in elementary, secondary and higher education curriculums. It is important to encourage a creative talent built on convergence of computational thinking and various major fields. However, proper analysis and evaluation of computational thinking assessment tools in higher education are currently not sufficient. In this study, we developed a rubric to evaluate computational thinking skills in university class from two perspectives: conceptual learning and practical programming training. Moreover, learning achievement and relevance between theory and practice were assessed. The proposed rubric is based on Computational Thinking Practices for assessing the higher education curriculum, and it is defined as a two-level structure which consists of four categories and eight items. The proposed rubric has been applied to a liberal art class in university, and the results were discussed to make future improvements

Keywords : Computational Thinking, Rubric, Higher-education, Computer Programming

† 정 회 원: 연세대학교 학부대학 조교수
논문접수: 2017년 8월 3일, 심사완료: 2017년 11월 7일, 게재확정: 2017년 11월 13일

1. 서론

지난 2006년에 J. M. Wing[1]은 컴퓨팅 사고의 개념을 제시하면서 컴퓨터 과학자뿐만 아니라 모든 학습자가 읽기, 쓰기, 셈하기의 기본 학습 능력에 추가하여 컴퓨팅 사고력을 갖추어야한다고 주장하였다. 실제로 오늘날의 4차 산업 환경에서 효과적으로 디지털 미디어를 읽고 해석하고 새로운 디지털 컨텐츠를 작성할 수 있는 능력인 디지털 문해력(digital literacy)이 사회 구성원의 기본 소양으로 자리를 잡고 있다[2].

이와 같이 컴퓨팅 사고력 및 프로그래밍 교육이 필수적으로 요구되면서 국내외 교육도 크게 변화하고 있다.

미국의 정규교육과정인 K-12에서는 컴퓨팅 사고력은 모든 분야에 걸쳐있으며 K-12의 모든 학생들이 반드시 습득해야 하는 능력으로 정의하고 있으며, ISTE(International Society for Technology in Education)와 CSTA(Computer Science Teachers Association)에서는 교사를 위한 지침서인 Computational Thinking in K-12 Education(2011)을 작성하여 컴퓨팅 사고력이 여러 교과과정과 다양하게 융합할 수 있는 사례를 제시하고 있다[3][4].

국내의 경우 교육부와 미래창조과학부는 2017년 3월에 소프트웨어 연구·선도학교 1,200개 학교를 발표하고, SW교육 필수화에 대비하여 정규 교육과정 등을 통해 SW교육을 운영하고 있다. 또한 2018년도부터 단계적으로 초·중등학교에서 의무적으로 SW교육을 실시하도록 하고 있다[5]. 대학교육의 경우에도 SW중심대학이 2017년 기준 20개로 확대되었으며 2019년까지 30개로 늘어날 예정이다[6].

이와 같은 추세에 맞춰 수많은 교육 기관에서 컴퓨팅 사고 및 프로그래밍 교육을 위한 다양한 커리큘럼이나 교육 사례가 연구되고 있으며[7][8][9][10], 이 중에는 컴퓨팅 사고를 정형화한 교육 과정을 개발하려는 연구도 진행되었다[11]. 그러나 컴퓨팅 사고는 문제를 해결하는 일련의 사고 과정이고, 이는 지극히 포괄적이고 추상적인 개념이기 때문에 이를 정형화하여 정의하고 학습도를 평가하기는 어렵다. 기존의 사례와 연구를 살펴보면 교육 과정의 수준, 관점 및 목표에 따라서 서로 다른 개념과 범위 및 교육 방법을 정의하고 있다[12][13].

예를 들어 초·중등 교육과정에서 블록 프로그래밍 등의 프로그래밍 교육을 통하여 주어진 문제의 해를 찾도록 개발물을 구현하는 경우에도 관점에 따라서 추상화나 데이터 표현, 알고리즘 정의, 문제 분할과 같은 컴퓨팅 사고가 사용된다고 볼 수 있으며, 컴퓨팅 도구를 전혀 사용하지 않는(unplugged) 환경에서 문제 풀이 과정을 설명하는 경우에도 컴퓨팅 사고의 개념을 교육한다고 볼 수 있다[14].

고등 교육과정인 대학 교양 과목에서 컴퓨팅 사고 교육의 경우에는, 학습자가 컴퓨팅 사고 이론을 확립하고 이를 자신의 전공분야에 실제로 융합할 수 있는 역량을 갖추는 것이 중요하다. 따라서 컴퓨팅 사고의 이론적 개념 및 기술을 학습하고, 이것을 프로그래밍 구현을 통하여 여러 분야의 문제에 실제로 적용하고 문제를 자동화하여 해결할 수 있도록 해야 한다.

따라서 본 연구의 컴퓨팅 사고 교육의 목표는 다음과 같이 정의한다. 대학 교양 수업에서 소프트웨어 비관련 전공자(이하 비전공자)를 대상으로 컴퓨팅 사고의 이론적인 개념을 교육한다. 나아가 학습자가 컴퓨팅 사고를 프로그래밍 언어로 표현할 수 있도록 하여 학습한 개념이 단지 이론으로 그치지 않고 당면한 문제에 적용시킬 수 있는 실용적인 컴퓨팅 기술로 습득하도록 한다.

이와 같은 교육의 목표가 제대로 달성되었는지 확인하기 위하여 본 논문에서는 두 가지 측면에서 컴퓨팅 사고를 평가하도록 한다. 첫째, 컴퓨팅 사고의 기본 개념을 학습자가 숙지하였는지 측정하고, 둘째, 컴퓨팅 사고 개념이 문제 해결 과정에서 구체적으로 프로그래밍을 통하여 표현되었는지를 측정한다. 마지막으로 컴퓨팅 사고의 이론적 개념이 실제로 문제해결에 효과적으로 적용되고 있는지 두 평가 결과의 차이를 계산하여 분석하고 개선점을 도출한다.

컴퓨팅 사고 교육에서는 개념의 습득, 원리를 적용하는 능력, 문제해결 능력, 개발물 구현 등 다양한 형태의 학습 능력이 포함되므로 학습의 결과(learning outcomes)[15]를 수치로 평가하기

위해서 분석적 평가 루브릭(analytic evaluation rubric) 도구를 정의하여 사용한다. 본 논문에서는 두 가지 측면에서의 평가 결과를 분석하기 위해 수행 차원(performance rubric)을 두 개의 레벨로 표현하고 각 레벨에서의 학습 수준을 측정할 수 있는 척도(scale)와 수행기술(descriptors)을 각 각 정의한다.

루브릭의 개발은 평가의 목표 및 특성을 반영하기 위해, 비전공자를 대상으로 하는 대학교 1학년 교양 수업의 수준을 평가하도록 설계되었고, 컴퓨팅 사고를 실례에 적용하는데 중점을 둔 Collegeboard의 AP Computer Science Principles Curriculum Framework의 Computational Thinking Practices(컴퓨팅 사고 실습) 평가 방법[14]을 기반으로 하였다.

2. 컴퓨팅 사고 교육 및 평가 관련 연구

본 장에서는 국내외의 컴퓨팅 사고 교육의 평가와 관련된 선행 연구를 조사하였다. 컴퓨팅 사고 교육과정의 활성화와 함께 루브릭 관련 연구도 최근 활발히 진행되고 있는 추세이다.

2.1 국외 연구

국외에서 컴퓨팅 사고 교육의 효과를 측정하기 위한 평가 기법에 대한 연구는 다음과 같다. Bort & Brylow[15]는 교사가 수업지도안에 컴퓨팅 사고의 기본 개념을 어느 정도 포함시킬 수 있는지를 측정하기 위한 루브릭을 제시하였다. 이들은 CSTA가 컴퓨팅 사고를 구성하는 기본 개념으로 <표 1>과 같이 정의한 9가지 요소에 여러 전공 분야에 해당 개념들을 접목 시킬 수 있는지 여부를 평가하는 Connection to Other Fields를 1개 추가하여 총 10개의 항목을 3개의 척도로 평가하였다.

Gouws, Bradshaw & Wentworth[16]는 대학교 컴퓨팅 사고 과목 수강 전과 후에 학습자들의 역량이 어떻게 변화하는지 세분화하여 측정하기 위해서 6가지 분류(Processes and Transformations, Models and Abstractions, Patterns and Algorithms, Tools and Resources, Inference and Logic, Evaluations and Improvements)로 컴퓨팅 사고 측정 항목을 개발하였다. 이들은 학기 중에 컴퓨팅 사고, 문제 해결 및 프로그래밍 작성 등의 서로 다른 유형의 시험을 4번에 걸쳐 시행하면서 학습자들의 수행 능력을 평가하였다.

<표 1> CSTA의 컴퓨팅 사고의 9가지 기본 요소

요소	정의
자료 수집	실세계에서 필요한 자료를 수집
자료 분석	자료에서 일반화된 패턴을 찾고 의미를 분석
자료 표현	그림, 차트 혹은 이미지 등을 이용하여 자료를 표현
문제 분해	문제를 해결 가능한 수준의 작은 문제로 분해
추상화	불필요한 요소를 제거하여 문제를 단순화
알고리즘과 절차	문제를 해결하기 위한 일련의 절차
자동화	컴퓨터를 활용하여 반복된 작업을 실행
시뮬레이션	문제를 해결하기 위한 과정과 절차를 미리 수행
병렬화	공통된 목표를 달성하기 위해 작업들을 동시에 수행

Seiter & Foreman의 PECT(Progression of Early Computational Thinking) 모형[17]은 국내외 연구에서 자주 인용되고 있다. 이 모형은 학습자의 스크래치 산출물을 3가지 측면(Evidence Variables, Design Pattern Variable, Computational Thinking Concepts)에서 평가하였다. Evidence Variable은 스크래치의 블록 유형을 세분화하여 분류하고, 학습자가 블록의 기능을 좀 더 다양하게 사용하였을 경우 높은 점수를 배정한다. Design Pattern Variables는 블록을 애니메이션, 말풍선 및 사용자 상호작용 등으로 분류하여 평가하며, Computational Thinking Concepts에서는 데이터 표현이나 추상화와 같은 컴퓨팅 사고의 기본 항목에 대해 Design Pattern Variables의 항목을 매핑하고 스프라이트 속성 사용, 변수, 동기화 블록 등의 여러 가지 기능을 다양하게 조합하여 사용하였을 경우 높은 평가를 받는다.

이밖에도 프로그래밍 결과물을 중심으로 평가하는 연구가 다수 진행되었다. Cateté[18] 등은 스크래치 프로그램의 결과물을 4개의 분류

(Accuracy, Efficiency, Reasoning 및 Readability)와 척도로 분석하여 평가하였으며, Eugene[19] 등은 학습자가 프로그래밍 과제 구현을 올바르게 하기 위한 가이드라인을 제시하는 루브릭을 개발하였다.

2.2 국내 연구

국내 연구에서 최형신[20]은 초등예비교사의 컴퓨팅 사고 수업 내용과 평가 방법을 고안하기 위해서 6개의 컴퓨팅 세부 요소(절차 및 알고리즘, 병행화 및 동기화, 자료 표현, 추상화, 문제 분해, 시뮬레이션)를 CSTA에서 제시하는 9개 요소를 기반으로 하여 제시하였다. 이를 통하여 학생의 스크래치 프로젝트 산출물을 기초, 발달 및 능숙의 척도로 평가하여 컴퓨팅 사고 능력의 증거를 분석하도록 하였다.

김민자 등[21]은 고등교육의 비전공자를 대상으로 하는 방학 특강에서 학생들이 개발한 프로젝트 산출물을 컴퓨팅 사고력 관점에서 평가하는 루브릭을 개발하기 위한 연구를 진행하였다. 이 루브릭은 CT Practices Design Pattern[22]에 기반을 두었으며 영역, 역량요소, 평가요소, 평가자료, 평가단계의 다섯 가지를 구성요소로 하고, 영역은 '추상화 모델의 설계', '창의적 산출물의 설계 및 적용', '산출물의 자가 평가'의 3가지로 구성하였다.

김수환[23]은 Brennan & Resnick[12]의 평가요소를 반영하여 절차, 반복, 분기, 상호작용, 데이터 표현, 추상화, 병렬 및 동기화로 컴퓨팅 사고 개념을 평가하였으며, 권정인[24]은 ISTA의 9가지 개념을 채용하여 대학교 신입생 중 비전공자의 컴퓨팅 사고 교과목의 학습 성과를 평가하였다.

2.3 개선점

기존의 루브릭 관련 연구를 정리하면 다음과 같다. 먼저 컴퓨팅 사고력에 대한 평가의 대상이 프로그래밍 구현물에 국한되는 경향을 보인다. 컴퓨팅 사고는 추상적이고 포괄적인 사고 과정이며 구체적 결과인 프로그래밍 구현물을 포함하는 충분조건이다. 예를 들어 앞절에서 살펴본 바와 같이 스크래치 프로그램 구현물에서 사용된 블록이 컴퓨팅 사고의 개념들에 대응된다고 할 때, 필요충분조건에 의해 학습자가 반드시 컴퓨팅 사고에 입각하여 구현물을 작성하였다고 보기 어렵다. 따라서 문제 해결의 추상적 사고과정 단계와 이것이 프로그래밍으로 구체적으로 반영되는 단계를 입체적으로 분석하고 두 단계 간의 관계를 분석 및 평가하는 것이 필요하다.

다음으로 대부분의 선행 루브릭 연구가 미국의 K-12나 초·중등 과정의 컴퓨팅 사고 교육 환경에 맞춰 어린이 혹은 청소년을 대상으로한 스크래치 산출물의 평가에 관한 연구가 대부분이다. 초·중등 교육과정에서는 블록 조합이나 언플러그드(unplugged) 방식의 교육을 통한 학습자의 창의성, 논리력 향상 및 흥미 유발 등이 중요한 평가지표가 될 수 있다.

그러나 고등 교육과정에서는 컴퓨팅 사고가 다양한 전공 분야에서의 문제 해결에 적용될 수 있도록 실용적인 교육과정을 구성해야 하며, 이에 맞는 컴퓨팅 사고력 평가에 대한 연구가 부족한 실정이므로 적합한 루브릭 개발이 필요하다.

3. 루브릭 설계 및 개발

본 장에서는 제안하는 평가 루브릭의 설계 기반과 정의 원칙 및 내용에 대해서 살펴본다.

3.1 설계기반

컴퓨팅 사고 요소와 프로그래밍 산출물을 동시에 평가하기 위한 루브릭 개발을 위해 선행 연구를 분석하였다. 널리 인용되고 있는 CSTA 모델의 경우 컴퓨팅 사고 요소를 정형화하여 표현하고 있으므로 추상적인 개념들을 평가하기에는 적합하였으나, 이것을 프로그래밍 요소에 적용하여 평가하기에는 적합하지 않았다. 반대로 프로그래밍 코드를 평가하기 위한 루브릭[25]은 구현물의 실행여부나 논리적 구조 등에 초점이 맞춰져 있어서 컴퓨팅 사고에 대한 평가가 부족하였다.

이에 반해 CollegeBoard의 AP Computer Science Principles Curriculum Framework 의 Computational Thinking Practices(컴퓨팅 사고

실습) 평가 방법은 컴퓨터 과학 과목의 학습자들이 컴퓨팅 사고에 의한 산출물을 효율적으로 개발할 수 있도록 학습 목표를 [표 2]와 같이 6가지로 분류하여 정의하고 있다[14].

<표 2> AP Computer Science Principles의 Computational Thinking Practices 구성요소

기호	구성 요소
P1	컴퓨팅과 실세계의 연결
P2	컴퓨팅 산출물 작성
P3	추상화
P4	문제 및 산출물 분석
P5	의사소통
P6	협업

P1은 실세계의 다양한 현상들을 컴퓨팅 개념으로 연결하는 것이며, P2는 적절한 알고리즘 및 컴퓨팅 기법을 활용하여 산출물을 생성하는 것이다. P3는 문제 해결과정에서 전반적으로 사용되는 추상화 개념이며, P4는 수학, 과학, 논리 및 미술 등 다양한 전공 분야의 지식을 활용하여 문제를 분석하고 해를 도출하며 P5는 문제를 기호나 시각적 도형 등을 사용하여 표현한다. 마지막으로 P6는 협업을 통해 보다 효율적인 문제 해결을 위한 요소이다.

이 프레임워크는 2장에서 언급한 두 가지 개선점을 반영할 수 있는 특성을 가지고 있다. 첫째로 컴퓨터과학의 이론적인 개념을 학습하고, 이를 바탕으로 한 컴퓨팅 산출물(computational artifacts)을 개발하는 것을 목표로 하고 있다. 즉 '컴퓨팅 사고 실습(Computational Thinking Practices)'이라는 제목처럼 사고와 실습 교육을 모두 고려한 교육과정을 목표로 한다. 또한 이 프레임워크는 대학교 1학년 컴퓨터과학 과목과 동일한 수준의 수업을 위해 디자인되었다. 따라서 대학 1학년생을 대상으로 하는 본 연구 환경에 가장 적합한 특성을 가진다.

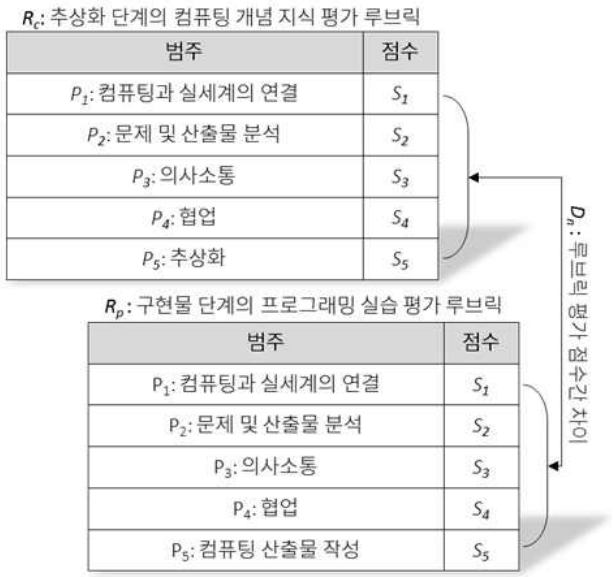
추가적으로 그룹별 문제 해결을 위한 구성원간의 소통(communication)과 협업(collaborating) 요소를 포함하고 있어 단순히 산출물에 대한 평가를 벗어나, 프로젝트 수행에 필수적인 사회적 상호작용에 대한 평가도 가능하도록 하였다.

이에 위 프레임워크는 본 연구의 목표와 가장 부합하므로 <표 2>에 소개된 구성 요소를 기반으로 루브릭을 개발하였다.

3.2 정의

학습자의 컴퓨팅 사고 개념에 대한 이해와 실습 능력을 평가하고 그 차이를 분석할 수 있는 루브릭을 정의하기 위해, Computational Thinking Practices를 기반으로 하되 본 연구의 목표에 맞게 기존의 요소들을 재구성하였다.

제안 루브릭은 [그림 1]과 같이 컴퓨팅 사고의 추상적 개념에 대한 평가 루브릭 R_c와, 프로그래밍 산출물에 대한 실습 평가 루브릭 R_p의 두 단계(level)로 구성하였다.



[그림 1] 제안 루브릭의 전체 구성

설계기반이 된 <표 2>는 학습자의 컴퓨팅 사고 이론과 실습을 함께 평가하기 위한 총 6가지의 구성요소로 이루어져 있다.

이 중 개념 지식과 프로그래밍 구현물 평가에 공통적으로 적용될 수 있는 P1, P2, P4 및 P6는 R_c와 R_p 에 공통적으로 도입하였고 추상화 개념인 P3는 R_c의 P_5로, 구현 부분에 해당하는 P2는 R_p의 P_5로 분류하였다.

즉 P_5는 R_c는 추상적 지식에 대한 이해를 나타내고 R_p는 산출물 생성 능력을 의미한다. 각 수행 차원의 평가 평균 점수는 S_n(n:차원의 순번)이며 S_5는 S_1부터 S_4까지의 합계로 계산되어 해당 루브릭의 전체 평가 결과를 나타낸다.

<표 3> 컴퓨팅 사고 개념 및 실습 평가를 위한 루브릭

단계	범주	평가항목	미숙함(Poor)	좋음(Good)	뛰어남(Excellent)
R_c	P_1	1. 데이터 표현 및 변환 인코딩 변환 기법 이해 여부	데이터 유형과 인코등에 대한 개념이 부족	부분적인 데이터 표현과 변환 기법의 이해	데이터의 유형을 정확히 이해하고 데이터 표현 및 변환함
		2. 추상적인 논리 기호의 의미를 이해하고 실세계 논리를 표현	추상적 표현을 이해하지 못하고 표현 하지 못함	추상적 표현의 이해가 미숙하여 불완전한 표현을 작성함	추상적 표현에 익숙하여 완전한 표현이 가능
		3. 데이터 구조의 종류, 원리 및 필요성과 연산에 이해 여부	데이터 구조와 연산을 이해하지 못함	데이터 구조 종류는 이해하나 원리와 연산이 미숙	데이터 구조의 특징과 연산을 이해하고 있음
	P_2	1. 제어 흐름 명령어의 종류와 기능 이해 및 표현	명령어를 이해하지 못하고 표현이 올바르지 못함	추상적 표현 방식을 사용하였으나 부분적 오류가 발견됨	추상적 표현방식을 이용하여 논리를 잘 표현함
		2. 문제를 분할하여 표현하고 모듈 정의 원칙에 따라 문제를 모듈화	문제 분할 개념과 모듈화 원칙에 대한 인식이 없음	일부분만 문제를 분할 함	문제를 잘 분할함
		3. 문제 해결에 필요한 패턴, 논리 및 기술을 파악	문제 해결과 관련이 없는 것을 도출함	부분적 오류가 있음	정확한 패턴을 도출
	P_3	1. 문제 해결 과정을 표현하기 위한 추상적 기호의 이해 및 작성	기호를 이해하지 못하고 도표를 거의 표현하지 못함	전체 과정을 표현하였으나 약속된 규칙 미사용	전체 과정을 명확하고 적절히 약속된 규칙에 의해 표현함
	P_4	1. 병렬처리와 협업을 통한 문제 해결 비용 단축 개념	개념 이해를 거의 하지 못함	부분적인 개념 이해	개념을 정확히 이해
R_p	P_1	1. 데이터형에 따른 올바른 상수 및 형변환 사용	데이터형 표현에 오류가 많음	데이터형의 표현에 간혹 오류가 발견됨	데이터형에 대한 명확한 사용과 형변환을 사용
		2. 올바른 논리와 문법의 Boolean 표현식 사용	표현과 논리에 오류가 있거나 작성하지 못함	표현과 논리 중 한쪽에 오류가 있음	정확한 논리와 표현을 사용
		3. 리스트 및 사전 정의 및 연산 활용	데이터 구조를 사용하지 않음	비효율적인 구조를 사용하거나 연산에 오류가 있음	적절한 데이터 구조와 연산을 잘 사용함
	P_2	1. 조건 및 반복문을 이용한 제어 흐름 작성	거의 코드를 작성하지 못함	미완성하였으나 의도한 제어 흐름은 이해할 수 있음	명확하고 논리에 오류가 없는 제어 흐름 구현
		2. 함수와 인자를 활용한 모듈화 프로그래밍	명령어를 단순 나열함	작성한 함수가 Rc::P2의 모듈 정의 원칙에 위배됨	함수 정의 원칙에 따라 프로그램이 모듈화됨
		3. 산술, 논리, 비교 연산식 및 정규식의 작성	표현식을 작성하지 못하거나 문법 오류가 있음	연산이 비효율적이거나 예외 발생의 여지가 있음	모든 부분에 올바른 표현식이 사용됨
	P_3	1. Rc::P3의 모델링 결과의 코드 반영 여부	모델링 결과와 무관하게 산출물을 작성함	모델링 결과가 부분적 반영되거나 임의 수정되었음	모델링 결과가 제대로 반영되어 있음
	P_4	1. 공동 작업을 통한 모듈 작성, 통합 및 테스트 여부	역할 분담이 불분명하고 통합이 이루어지지 않음	역할을 분담하여 모듈을 작성하였으나 통합이 불완전함	역할을 분담 모듈을 작성 및 책임자의 통합/테스트 수행

D_n은 두 루브릭의 대응되는 S_n의 거리로 정의된다. 점수 차이가 작은 경우 D_n을 최소화하고, 점수 차이가 큰 경우 D_n의 값을 극대화시켜서 항목 간의 거리를 명확히 나타내기 위해서 D_n을 아래 <수식 1>로 정의하였다(m은 평가 대상 학습자 수).

$$Dn = \frac{1}{m}\sum_{i=1}^{m}(R_c :: S_n - R_p :: S_n)^2 \quad \text{.....<수식 1>}$$

제안 루브릭의 기반이 된 'Computational Thinking Practices'은 컴퓨팅 사고 개념, 인터넷의 기능, 사이버 보안, 빅데이터 활용 등을 포함하여 매우 다양하고 세분화된 44개의 학습 목표를 제시하고 있다. 제안 루브릭의 정의 목표에 맞도록 이 중에서 <표 1>에 해당하는 항목을 선별하고 이를 학습 내용의 평가에 적합하도록 수정 및 병합한 16개의 항목을 구성하였다.

제안 루브릭에서는 컴퓨팅 사고 개념과 프로그래밍 구현 부분을 분리하여 다음과 같은 평가 내용으로 구성하였다.

R_c의 P_1에서는 문제 해결에 필요한 데이터의 표현 방식, 종류 및 구조를 이해하고, 인간의 연역적 사고를 기호 로직(symbolic logic)으로 표현할 수 있지를 검사한다. P_2에서는 알고리즘을 구성하는 기초 제어흐름 명령어를 이해하고 이를 의사코드(pseudo code) 형식으로 표현할 수 있는지를 검사한다. P_3에서는 통합 모델링 언어를 이해하고 문제해결 과정을 시각화하여 표현할 수 있는지를 확인하며, P_4에서는 병렬 처리 개념을 평가한다.

R_p의 P_1은 프로그래밍 언어에서 올바른 변수 및 상수 표현과 자료 구조 및 형변환(type casting)의 사용, 그리고 불린(Boolean) 표현식 작성 능력을 평가한다. P_2에서는 연산식, 조건문, 반복문 및 함수 등과 같은 제어 흐름 명령어 작성 능력을 검사한다. P_3에서는 소프트웨어 디자인 단계에서 통합 모델링 언어로 작성된 시각화 도표를 대응되는 프로그래밍 구조로 구현하는 능력을 확인하고, P_4에서는 협업 작업을 통하여 소프트웨어를 모듈화하여 작성하고 통합하는 능력을 평가한다.

P_5는 평가 항목을 따로 가지지 않으며 각 루브릭의 특성을 대표하는 항목이므로 <표 3>내용에는 포함하지 않는다.

개발된 루브릭은 <표 3>와 같으며 R_c에서는 문제 해결을 위한 개념을 습득하고 표현할 수 있는지 여부를 주로 평가하며, R_p에서는 R_c에서 습득한 개념을 적용한 프로그래밍 산출물을 생성할 수 있는 능력을 평가한다. 평가 점수는 3단계로 구성되어 각 단계에서 1점에서 3점을 할당하도록 하였다. 산출물이 전혀 없거나 무응답인 경우는 0점으로 처리하여 0점에서 9점까지 총 9단계의 점수를 가진다.

3.3 제안 루브릭과 평가의 타당성

제안 루브릭의 신뢰도 검증을 위하여 내적 일관성을 측정하기 위해 널리 사용되는 Cronbach's alpha 계수를 구하였다. R_c의 전체 Cronbach's alpha는 0.73이며 R_p는 0.80으로 높게 나타났다. 두 루브릭을 대상으로 한 결과에서도 0.89로 산출되어 두 개의 루브릭 간에도 내적 일관성이 있음을 알 수 있었다.

4. 루브릭 적용 및 결과

개발된 루브릭을 2017년도 1학기에 개설된 대학 1학년 146명을 대상으로 컴퓨팅 사고 및 파이썬 프로그래밍을 학습하는 교양 수업에 적용하였다. 수강생들은 계열별로 경영(29명, 19.9%), 이학(21명, 14.4%), 상경(20명, 13.7%), 인문(15명, 10.3%), 의・치의예(12명, 8.2%), 생활과학(11명, 7.5%), 공학(9명, 6.2%), 사회과학(8명, 5.5%), 교육과학(8명, 5.5%) 및 그 외 계열(13명, 8.9%)로 구성되었으며 수강생들은 이전에 프로그래밍 과목을 수강한 경험이 없었다.

루브릭을 적용한 과목은 대학교의 교양 수업으로 16주간 매주 이론 2차시 및 실습 2차시 학습으로 진행하였으며, 이론 시간에는 컴퓨팅 사고의 개념을 학습하고, 실습 시간에는 컴퓨팅 사고 개념을 파이썬 언어로 구현하는 구성으로 이루어졌다. 따라서 수업은 개발된 루브릭과 대응되는 구조로 설계되었으며, R_c 범주의 학습 목표에 대한 테스트 문항, R_p 범주의 실습 산출물, 과제 및 기말 프로젝트를 3명의 평가자가 평가하였다.

테스트 문항은 SW전공자들이 수강하는 프로그래밍 언어 과목의 경우처럼 알고리즘의 응용을 통한 해답 도출 위주의 문제보다는 교양 과목을 수강하는 다양한 전공의 학생들이 연역적 추론, 순차적 실행, 분기, 반복 및 모듈화와 같이 문제 해결에 필수적인 자료 표현, 논리적 사고, 알고리즘 패턴 및 추상화 등을 얼마나 잘 표현했는가를 평가하도록 난이도를 설정하였다.

4.1 루브릭 적용 결과 및 분석

루브릭의 각 항목에 대한 평가 결과 수치는 <표 4>과 같고 분포는 [그림 2]와 같다. 평가 결과인 S_1부터 S_5에서 유·무의미한 차이가 항목별로 서로 다른 결과를 보였으므로 각 결과에 대해서 원인을 분석하여 살펴보도록 한다. 한 분류에 평가 항목이 여러 개인 경우는 첨자를 추가하여 표현하였다.

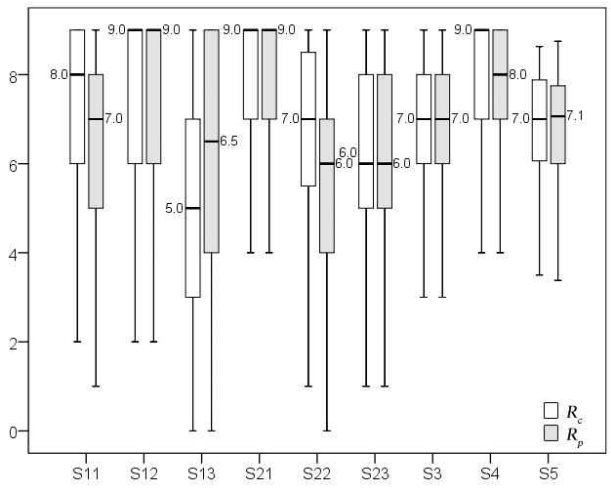
두 루브릭의 S_{12}, S_{21}, S_{23}, S_3 및 S_4 항목에서 평

가 점수간의 상관관계는 유의미하지 않았다($p > .05$). 이는 논리의 추상적 표현, 제어 흐름을 이용한 알고리즘 작성, 문제 해결에 필요한 패턴 및 공식 도출, 해결 과정의 도식화 및 구현, 그리고 병렬처리 개념 및 협업에서 이론 학습과 실습 수행이 모두 비슷한 수준이며 평균도 높아 본 수업의 초기 목표를 달성하였다. 즉, 비SW전공자들이 데이터 표현, 논리적 사고, 추상화, 모듈화 및 협업 등의 개념을 활용하여 문제를 해결하고, 그 과정을 프로그래밍 언어로 표현하여 자동화된 해를 구하는 컴퓨팅 문제 해결 능력을 습득하였다는 것을 의미한다.

<표 4> 루브릭 적용에 대한 평가 결과

	S_{11}	S_{12}	S_{13}	S_{21}	S_{22}	S_{23}	S_3	S_4	S_5
R_c	7.29	7.51	4.89	7.73	6.84	5.61	7.04	7.96	6.86
R_p	6.60	7.53	6.05	7.86	5.71	5.90	7.08	7.58	6.79
D_n	0.75	0.12	2.29	0.44	2.42	0.47	0.39	0.49	0.07
p^*	0.02	0.94	0.00	0.48	0.00	0.35	0.77	0.05	0.64

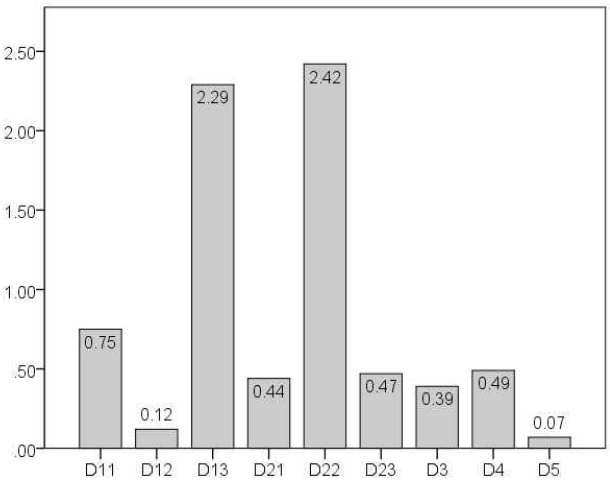
*p는 R_c와 R_p에 대한 유의도



[그림 2] 두 루브릭 문항 점수의 분포 비교

다만 S_4의 경우 평균의 차이는 무의미하지만 다른 항목에 비해 다소 R_p의 값이 R_c에 비해 낮은데, 학습자들이 팀 단위로 협업 작업을 진행하는 과정에서 모듈을 통합하고 검증하는 작업을 수행하지 않아 최종 산출물에서 오류가 발견됐기 때문이다. 이는 이론 학습 과정에서 결과물의 통합 및 테스트 기법을 구체적으로 교육하지 않아 발생한 것으로 이론 수업 내용의 보완이 필요하다.

S_{11}, S_{13} 및 S_{22}에서는 두 루브릭 간의 D_n이 [그림 3]과 같이 차이를 보이며 유의미한 결과를 나타냈다($p < .05$). D_{11}에 해당하는 S_{11} 항목의 경우에 학습자들이 실세계의 정보를 계산 가능한 데이터로 변환하고 표현하는 개념에 대해서는 높은 점수를 받았으며 이를 산출물로 구현하는 과정에서도 정수, 실수 및 텍스트 데이터를 표현하는데 문제가 없었으나, 데이터 유형간의 변환에 익숙하지 못하여 오류를 범하는 경우가 잦았다.



[그림 3] <수식 1>에 의한 항목간 거리

D_{13}의 경우에는 R_p의 S_{13} 값은 높으나 R_c에 속하는 값이 오히려 낮은 특성을 보였다. 자료 구조의 설계 원리에 대한 이해와 삽입, 삭제 및 검색과 같은 연산의 원리를 이해하는데 어려움을 겪었으나, 실제로 프로그래밍에서 자료구조와 관련된 연산을 활용하는 것은 큰 어려움이 없었던 것으로 파악되었다.

가장 높은 값을 가지는 D_{22}는 학습자들이 모듈화 개념을 실제로 구현시키는데 익숙하지 않음을 보여준다. 평가지와 프로그래밍 산출물을 분석해 보면 학습자들이 문제 분할에 대한 개념 이해와 이를 도식화하여 표현하는 능력도 대체적으로 양호한 편이나, 실제로 구현 단계에서 문제를 분할하여 산출물에 반영하지 못하는 경우가 많았다.

즉, 함수로 풀이 과정을 모듈화 하지 않고 순차적으로 명령어를 나열하거나, 함수를 정의하더라도 특정 함수에 지나치게 많은 내용이 치중되어 적절히 모듈화 하였다고 보기 힘든 경우도 있었다. 이러한 현상은 구조적 프로그래밍을 작성하기

에 아직 구현 경험이 부족하여 발생하는 것으로 보이며, 스켈레톤 코드(skeleton code) 제공이나 문제에서 명시적으로 구조화하도록 지침을 주어 학습자들이 모듈화 개념에 익숙해지도록 하여야 할 것이다.

마지막으로 두 루브릭의 S_5는 유의미한 차이가 없었다(p=.64 > .05). 따라서 추상적인 컴퓨팅 개념이 프로그래밍 실습을 통한 구체적인 산출물에도 전체적으로 반영되어 있다는 것을 알 수 있었다. 두 루브릭간의 평균에는 차이가 없었으나 표준편차는 R_c의 S_5는 1.23, R_p의 경우는 1.31로 후자가 좀 더 편차가 높았다. 전자의 경우에는 대부분의 학습자가 단기간의 암기나 학습으로 어느 정도의 성과를 볼 수 있었으나, 프로그래밍 산출물을 구현하는 것은 지속적인 장기간의 연습이 필요하기 때문에 평소 실습을 충실히 한 학습자와 그렇지 못한 학습자들 간의 편차가 다소 높은 것으로 분석된다. 프로그래밍 구현이 암기로 해결될 수 없고 외국어 회화 학습과 마찬가지로 꾸준한 실습이 중요함을 학습자에게 충분히 인식시키는 것이 필요하다.

5. 결론 및 향후 연구

본 논문에서는 컴퓨팅 사고의 개념과 실습에 대한 학습도를 평가하기 위해서 두 단계로 구성된 루부릭을 개발 및 적용하였다. 제안 루브릭은 CollegeBoard의 Computational Thinking Practices 모델을 기반으로 하여 대학교 신입생의 이론과 실습 교육을 평가할 수 있도록 개발되었다.

대학교 정규 수업의 산출물에 제안 루브릭을 적용한 결과, 컴퓨팅 사고의 추상적 개념에 대한 학습도와 구체적 프로그래밍 구현 역량을 평가할 수 있었다. 또한 두 루브릭 간의 대응되는 항목간의 차이를 계산함으로서 이론적 개념이 실제 기술로서 잘 습득되고 있는지를 파악할 수 있었으며, 그렇지 못한 경우 원인 분석을 통하여 향후 개설 수업에서 보완이 가능하도록 방향을 제시하였다.

현재 SW중심대학교를 출발점으로 컴퓨팅 사고 과목이 전교생 필수 과목으로 지정되는 추세이다. 그러나 교육의 효과를 평가할 수 있는 도구나 기술은 부족한 실정이며, 다양한 평가 도구 및 기술의 개발이 이루어져야 보다 효과적인 컴퓨터 과학 교육이 이루어 질 수 있을 것이다.

참 고 문 헌

[1] Wing, J. M. (2006). Computational Thinking. *Commun. ACM, 49*(3), 33-35.

[2] Pérez-Escoda, A. & José Rodríguez-Conde, M. (2015). Digital literacy and digital competences in the educational evaluation: USA and IEA contexts. *3rd International Conference on TEEM, ACM, NY*, 355-360.

[3] Computer Science Teachers Association and the International Society for Technology in Education (2011). Computational Thinking in K‐12 Education, teacher resources.

[4] Barr, V. & Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community, *ACM Inroads, 2*(1), 48-54.

[5] 한국컴퓨터교육학회 (2014). 국내외 SW교육 운영현황 및 요구사항 조사 보고서, 정보통신산업진흥원.

[6] 전자신문 보도 (2016). http://www.etnews.com/20160825000363

[7] 권정인, 김재현 (2017). Computational Thinking기반 SW교육과 문제해결력에 관한 연구, **한국컴퓨터교육학회 학술발표대회논문집**, **21**(1), 9-10.

[8] 안성훈 (2016). 초·중학교 SW교육을 위한 프로그래밍 평가지표 개발. **한국컴퓨터교육학회 논문지**, **19**(4), 11-20.

[9] 전수진 (2017). SW 교육에서의 컴퓨팅 사고력 기반 디자인 중심 모형(NDIS)의 효과분석. **한국컴퓨터교육학회 논문지**, **20**(2), 13-21.

[10] 박성빈, 안성진 (2016). 컴퓨팅사고력의 역량 탐색 연구: 소프트웨어개발자를 중심으로. **한국컴퓨터교육학회 논문지**, **19**(5), 41-53.

[11] Dierbach, C., Hochheiser, H., Collins, S., Jerome, G., Ariza, C., Kelleher, T., Kleinsasser, W., Dehlinger, J. & Kaza1, S. (2011). A Model for Piloting Pathways for Computational

Thinking in a General Education Curriculum. *the 42nd ACM Computer science education, ACM, NY*, 257-262.

[12] Brennan, K. & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking, *2012 annual meeting of the American Educational Research Association, Vancouver*, 1-25.

[13] Barendsen, E. & Stoker, I. (2013). Computational thinking in CS teaching materials: a pilot study. *13th Koli Calling International Conference on Computing Education Research, ACM, New York*, 199-200.

[14] Rodriguez, B., Kennicutt, S., Rader, C. & Camp, T. (2017). Assessing Computational Thinking in CS Unplugged Activities. *2017 ACM SIGCSE Computer Science Education, ACM, New York*, 501-506.

[15] Gagne (1985). The conditions of learning (4th ed.). Newyork: Holt, Rinehart & Winston.

[14] CollegeBoard (2017). AP Computer Science Principles Including the Curriculum Framework, CollegeBoard.

[15] Bort, H., Brylow, D. (2013). CS4Impact: Measuring Computational Thinking Concepts Present in CS4HS Participant Lesson Plans, *44th ACM symposium on Computer science education, ACM, NY*, 427-432

[16] Gouws, L., Bradshaw, K. & Wentworth, P. (2013). First year student performance in a test for computational thinking. *Computer Scientists and Information Technologists Conference, ACM, NY*, 271-277.

[17] Seiter, L. & Foreman, B., (2013). Modeling the learning progressions of computational thinking of primary grade students. 9*th annual international ACM conference on International computing education research, ACM, NY*, 59-66.

[18] Veronica Cateté, V., Snider, E. & Barnes, T. (2016). Developing a Rubric for a Creative CS Principles Lab. *2016 ACM Conference on Innovation and Technology in Computer Science Education, ACM, NY*, 290-295.

[19] Eugene, K., Stringfellow, C. & Halverson, R. (2016). The usefulness of rubrics in computer science. *Journal of Computer Science. Coll, 31*(4), 5-20.

[20] 최형신 (2014). Computational Thinking 역량 개발을 위한 수업 설계 및 평가 루브릭 개발, **한국정보교육학회 논문지**, **18**(1), 57-64.

[21] 김민자, 유길상, 김현철 (2017). 비전공자 프로그래밍 수업 창의적 산출물의 컴퓨팅 사고력 기반 평가 루브릭 개발. **한국컴퓨터교육학회 논문지**, **20**(2), 1-10.

[22] Bienkowski, M., Snow, E., Rutstein, D. & Grover, S. (2015). Assessment design patte rns for computational thinking practices in secondary computer science:A first look (S RI technical report). *Menlo Park, CA: SRI International*.

[23] 김수환 (2015). Computational Thinking 개념 평가를 위한 스크래치 코드 분석 시스템 개발. **한국컴퓨터교육학회 논문지**, **18**(6), 13-22.

[24] 권정인, 김재현 (2017). Computational Think ing기반 SW교육과 문제해결력에 관한 연구. **한국컴퓨터교육학회 동계 학술발표논문지**, **21**(1), 9-10

[25] Fitzgerald S., Hanks, B., Lister, R., McCa uley, R., & Murphy, L. (2013). What are w e thinking when we grade programs? *44th ACM technical symposium on Computer s cience education. ACM, NY*, 471-476.

김 재 경

2007 연세대학교
컴퓨터과학과(공학박사)
2014 연세대학교
컴퓨터과학과 객원교수
2017～현재 연세대학교 학부대학 조교수
관심분야: 컴퓨터과학 교육
E-Mail: kim.jk@yonsei.ac.kr