Journal of The Korean Association
of Information Education Vol. 23, No. 3,
June 2019, pp. 207-217

http://dx.doi.org/10.14352/jkaie.2019.23.3.207

초등학생의 스크래치 프로젝트 코드 분석을 통한 컴퓨팅 사고력 평가

박주연

이대부속초등학교

요약

컴퓨팅 사고력을 향상시키기 위해 초등학교에서는 블록형 프로그래밍 언어인 스크래치를 활용한 기초 프로그래밍 교육을 하고 있으나 컴퓨팅 사고력에 대한 평가 연구는 초기 단계이다. 따라서 본 연구에서는 스크래치 프로젝트의 코드를 분석하는 방법을 활용하여 초등학생들의 컴퓨팅 사고력의 개념 수준을 평가하였다. 이를 위해 스크래치 코드 분석 자동화 도구인 Dr. Scratch를 활용하여 초등학교 6학년 학생들이 제작한 179개의 스크래치 프로젝트를 분석하였다. 연구결과 초등학생의 컴퓨팅 사고력의 개념수준은 개발자 수준이 많았고, 성별과 작품 유형에 따라 차이가 있었고, 논리와 추상화 요소에서 가장 낮은 수준을 보였으며, 프로그래밍 과정에서 컴퓨팅 사고력이 향상되는 것으로 나타났다. 본 연구는 초등학생의 프로그래밍 학습에서 교수방법의 개선과 자기주도적 컴퓨팅 사고력의 평가에 대한 시사점을 제공한다.

키워드 : 스크래치, Dr. Scratch, 컴퓨팅 사고력, SW 교육, 코드 분석

Evaluation of Computational Thinking through Code Analysis of Elementary School Students' Scratch Projects

Juyeon Park

Ewha Womans Elementary School

ABSTRACT

In order to improve computational thinking, elementary schools have been using 'Scratch' to provide basic programming education. However, the study on evaluation of computational thinking is at an early stage. Therefore, this study evaluated the conceptual level of computational thinking using the scratch code analyzing. For this, Dr. Scratch was used to analyze 179 scratch projects. The results showed that the conceptual level of computational thinking of most elementary students was at the developing level, and that it varied according to gender and production style, showed the lowest level of logic and abstraction, and improved computational thinking during programming. This study is meaningful in that it provides implications for the improvement of teaching methods and self-directed evaluation in learning.

Keywords : Scratch, Dr. Scratch, Computational Thinking, SW education, Code analysis

논문투고 : 2019-05-03
논문심사 : 2019-06-14
심사완료 : 2019-06-17

1. 서론

컴퓨팅 사고력을 향상시키기 위한 SW교육이 초·중등학교에서 본격적으로 시작되면서 언플러그드 활동, 프로그래밍, 피지컬 컴퓨팅 등 다양한 형태로 SW교육이 이루어지고 있다. 초등학교에서는 대부분 블록형 프로그래밍 언어를 활용한 기초프로그래밍 교육을 실시하고 있다. 블록형 프로그래밍 환경은 문법오류가 없고 학생들의 인지 부하를 발생시키지 않아 초등학생의 인지적 수준에 적합하고, 초등학생의 흥미와 관심을 끌 수 있는 시청각적인 요소를 제시하고 있기 때문이다[15].

MIT 미디어랩에서 개발한 스크래치는 전세계에서 활용하고 있는 대표적인 블록형 프로그래밍 언어로서 학생들은 이를 활용하여 이야기, 애니메이션, 음악, 미술, 게임 등 흥미 있는 결과물을 손쉽게 만들어 낼 수 있다[1][4]. 전 세계의 학생들은 매일 다양한 프로젝트를 스크래치 웹사이트에서 공유하고 있다. 학생들은 자신의 생각을 스크래치 프로젝트로 표현하는 과정에서 논리적이고 창의적으로 생각하고 컴퓨팅 사고력을 계발한다.

그러나 학생들이 프로그래밍 학습 과정에서 계발하고 있는 컴퓨팅 사고력에 대한 평가는 활발히 이루어지지 못하고 있는 실정으로 이에 대한 연구는 시작단계라고 할 수 있다[5][6]. 그 이유로는 컴퓨팅 사고력의 평가의 중요성에 대한 인식의 부재, 컴퓨팅 사고력의 평가 요소에 대한 혼란, 교수자와 학습자가 손쉽게 접근할 수 있는 컴퓨팅 사고력의 평가 도구의 부재, 컴퓨팅 사고력의 평가방법에 대한 복잡성 등을 들 수 있겠다.

스크래치 프로그래밍 학습에서 학생들의 컴퓨팅 사고력을 평가하는 방법으로는 여러 가지가 있으나 프로젝트의 코드 분석은 컴퓨팅 사고력의 개념을 평가하고 이를 프로그래밍 학습에 활용하는데 유용한 방법이라고 할 수 있다[14]. 스크래치 코드 분석을 통해 교수자는 학생들의 스크래치 프로젝트에서 나타난 컴퓨팅 사고력을 평가하여 이를 통해 교수학습 과정에서 고려해야 하는 교육내용과 활동에 대한 아이디어를 얻을 수 있다. 더불어 학습자 스스로 자기평가를 실시할 수 있는데, 프로젝트의 제작 과정에서 각 코드를 사용한 목적을 가장 잘 알고 있는 학습자가 자신의 프로젝트의 코드를 분석함으로써 학습 과정을 돌아보고 학습의 개선점을 찾아낼 수 있다. 이러한 스크래치 코드분석 도구로서 Dr. Scratch는 학생들이 만든 스크래치 프로젝트의 코드를 컴퓨팅 사고력의 개념을 구성하는 각 요소별로 분석하므로 학습 과정에서 손쉽게 활용하기에 유용하다[7][10][11][14].

따라서 본 연구에서는 학생들의 스크래치 프로젝트 코드 분석 방법을 활용하여 학생들의 성별, 스크래치 프로젝트의 작품 유형에 따라 컴퓨팅 사고력의 개념 수준을 평가하고, 프로젝트 제작 과정에서 컴퓨팅 사고력의 각 요소들의 발전 과정을 분석함으로써 소프트웨어 수업을 위한 학습코스 및 전략 개발 및 자기주도적인 컴퓨팅 사고력의 평가 방법을 개발하는데 기초자료를 제공하고자 한다.

2. 이론적 배경

2.1 스크래치 프로젝트의 평가방법

컴퓨팅 사고력은 컴퓨터 과학을 통해 복잡한 문제를 해결하는 절차적 사고력으로 Papert에 의해 시작하여 Wing에 의해 모든 사람들에게 필요한 사고력으로서 그 중요성이 널리 알려지고 있다. Wing은 컴퓨팅 사고력의 개념을 추상화와 자동화의 두 축으로 설명하였고[17][18], CSTA & ISTE(2011)는 자료수집, 자료분석, 자료표현, 문제분해, 추상화, 알고리즘 및 절차, 자동화, 병렬화, 시뮬레이션을 그 구성요소로 제시하였다[3]. 또한, MIT에서는 스크래치 프로그래밍 과정에서 활용하는 컴퓨팅 사고력이 컴퓨팅 개념, 컴퓨팅 실천, 컴퓨팅 관점의 세 개의 차원으로 구성된 것으로 보았다[1]. 컴퓨팅 개념은 프로그래밍을 위한 기초 개념으로서 순서, 고리, 평행적 관계, 사건, 조건문, 작동, 자료 등에 대한 것이다. 컴퓨팅 실천은 반복적, 증가적 작동, 오류 수정, 재사용, 재조합, 추상, 모듈화에 대한 것이고, 컴퓨팅 관점은 표현하기, 연결하기, 의문갖기와 관련된다[1].

스크래치 프로그래밍 학습에서 컴퓨팅 사고력의 평가 방법에 대해 Brennan과 Resnick(2012)은 스크래치 프로젝트의 코드 분석, 작품 인터뷰, 디자인 시나리오의 세 가지 평가방법을 제안하였다[1][13]. 그 중 작품 인터뷰와 디자인 시나리오의 분석 방법은 과정에 대한 이해와 학습자의 생각과 관점까지 평가할 수 있는 장점이 있으나, 학습자의 기억력에 의존해야 한다는 것과, 실제 교육 현장에서 사용하기에는 시간 확보가 어렵고, 내용을 분석하는

시간과 여건에서 여러 가지 제한점이 있다[13]. 또한 평가자가 학습자를 평가하는 방식이므로 자기 평가가 이루어지기 어렵다. 학습의 주체인 학생 스스로가 학습의 과정에서 자기평가를 통해 자기주도적으로 학습을 구성하고 이끌어 가기 위한 목적으로는 활용되기 어렵다고 하겠다.

프로젝트의 코드 분석은 개념이 포함된 블록의 사용을 평가하는 방법으로 프로그래밍을 통한 컴퓨팅 사고력의 개념을 평가하는 가장 유용하고 간편한 방법이다[7][10][11][14]. 더욱이 온라인에서 프로젝트의 코드 분석 도구를 활용하여 컴퓨팅 사고력의 개념 수준과 구성요소별 수준을 손쉽게 평가하여 피드백을 제공하고, 학습과정에서 학습자는 자기평가 및 형성적인 평가방법으로 사용할 수 있다.

따라서 프로젝트의 코드 분석은 기초적인 프로그래밍 활동을 하는 초등학생을 대상으로 적용하기에 용이하고, 시간이 적게 들고, 교수자 뿐 아니라 학습자가 스스로 자신의 학습과정을 평가할 수 있다는 점에서 유용하게 사용될 수 있다.

2.2 스크래치 프로젝트의 코드 분석 방법

스크래치는 현재 10개의 카테고리로 명령어 블록이 구성되어 있다. 이는 동작, 형태, 소리, 펜, 데이터, 이벤트, 제어, 감지, 연산, 추가블록이다. 이러한 각 블록의 기능을 활용하여 학생들은 자신의 프로젝트를 만드는 과정에서 컴퓨팅 사고력이 발현되고 계발된다[1]. 이러한 컴퓨팅 사고력을 측정하기 위해서는 자동화된 코드 분석 도구인 Scrape, Dr. Scratch 등을 활용할 수 있다. 이 도구들은 스크래치 프로젝트에 사용된 각 블록들의 사용횟수를 분석함으로써 학생들의 컴퓨팅 사고력 개념을 손쉽게 평가한다[7].

Scrape는 각 블록의 횟수, 저장횟수, 스프라이트 수, 블록 수, 스택 수 등을 분석하는 자동화 도구이다. Dr. Scratch는 이를 발전시켜 컴퓨팅 사고력의 속성으로 플로우 제어, 데이터 표현, 추상화, 사용자 상호작용, 동기화, 병렬화, 논리의 7개 요소를 규정하고, 각 점수를 3점 만점으로 판별하여 그 결과를 보여준다[20]. Dr. Scratch에서 제시하는 7개의 각 요소들은 Brennan & Resnick(2012)이 제시한 컴퓨팅 개념의 평가틀과 유사하다[14]. 분석 결과는 각 요소별 합계를 기준으로 하여 세 등급으로 전체 수준을 보여주는데, 총점을 기준으로 0~7점은 기본 단계(Basic), 8~14점은 개발 단계(Developing), 15~21점은 마스터 단계(Master)이다. 또한 코드 복사, 스프라이트 이름 변경, 죽은 코드, 스프라이트 속성을 보여준다[20]. Dr. Scratch에서 컴퓨팅 사고력의 7개의 요소별로 점수를 부여하는 기준은 <표 1>과 같다[10].

<Table 1> Competence Level for each CT concept (Moreno-León, et al., 2015)

CT Concept	Competence Level		
	Basic (1 point)	Developing (2 points)	Proficiency (3 points)
Flow control	Sequence of blocks	Repeat, forever	Repeat until
Data representation	Modifiers of sprites properties	Operations on variables	Operations on lists
Abstraction	More than one script and more than one sprite	Definition of blocks	Use of clones
User interactivity	Green flag	Key pressed, sprite clicked, ask and wait, mouse blocks	When %s is >%s, video, audio
Synchronization	Wait	Broadcast, When I receive message, stop all, stop program, stop programs sprite	Wait until, when backdrop change to, broadcast and wait
Parallelism	Two scripts on green flag	Two scripts on key pressed, two scripts on sprite clicked on the same sprite	Two scripts on when I receive message, create clone, two scripts when %s is >%s, two scripts on when backdrop change to
Logical thinking	If	If else	Logic operations

Dr. Scratch를 활용한 연구들을 살펴보면,

Moreno-León 외(2016)는 Dr. Scratch와 전통적인 소프트웨어 공학 매트릭스인 McCabe의 순환복잡도, Halstead의 매트릭스를 비교하여 Dr. Scratch가 컴퓨팅 사고력을 측정하는데 타당한 도구임을 밝혔다[11]. 또한, Moreno-León 외(2015)의 연구는 10~14세 학생들 100명을 대상으로 Dr. Scratch를 활용하여 스스로 컴퓨팅 사고력을 평가하게 하고 피드백을 제공하였는데, 그 결과 학생 스스로 스크래치 프로젝트를 향상시키는데 효과가 있었음을 보고하였다[10]. Moreno-León 외(2017)의 연구에서는 스크래치 온라인 플랫폼에서 250개의 프로젝트를 수집하여 이를 5개의 작품 유형(미술, 음악, 애니메이션, 이야기, 게임)으로 분류하여 컴퓨팅 사고력을 분석하였다. 그 결과 각 작품 유형에서 사용된 컴퓨팅 사고력의 수준이 다르게 나타났음을 밝혔고, 특히 게임 유형에서 가장 높은 컴퓨팅 사고력의 수준을 보이는 것으로 보고하였다[12]. 또한, Browning(2017)의 연구에서는 41명의 4~6학년 학생들을 대상으로 1주일에 1시간씩 9주 동안의 스크래치 워크샵을 실시하고, Dr. Scratch를 형성평가 방법으로 사용하였는데, 이 그룹에서 컴퓨팅 사고력이 향상된 것을 밝혔다[2].

Dr. Scratch를 활용한 국내 연구로 대학생을 대상으로 한 김수환(2015)의 연구에서 학습자 수준이 높을수록 평가요소의 점수가 높은 것을 밝혔고, Dr. Scratch의 각 요소들이 컴퓨팅 사고력의 수준을 평가하는 요소로 활용될 수 있음을 보여주었다[7]. 또한, 박선주(2018)는 예비교사를 대상으로 한 수업에서 스크래치 프로젝트 코드 분석을 통해 학점과의 연관성을 분석하였고, 컴퓨팅 사고력 및 자기평가 도구로서의 Dr. Scratch의 활용가능성을 제안하였다[14].

이러한 결과들을 종합해 보면 컴퓨팅 사고력의 평가 도구로서의 Dr. Scratch를 통한 코드 분석의 타당성과 피드백을 통한 평가의 순환적 기능, 형성평가 방법으로서 Dr. Scratch의 활용가능성을 시사한다.

그러나 이와 같은 국내의 선행연구들은 대부분 대학생을 대상으로 한 것으로, 스크래치는 블록형 프로그래밍 언어로서 초등학생들에게 주로 활용됨에도 불구하고 초등학생의 스크래치 코드를 분석한 국내 연구는 찾아보기 어렵다. 따라서 본 연구에서는 초등학생의 스크래치 프로젝트 코드분석을 통해 컴퓨팅 사고력을 평가하고자 한다.

3. 연구 방법

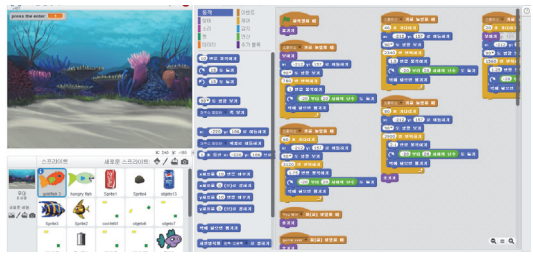
3.1 분석대상

본 연구에서 분석한 스크래치 프로젝트는 2013~2014년에 서울시 소재 A초등학교 6학년 학생들이 제작한 개별 프로젝트 160개와 팀 프로젝트 19개이다.

개별 프로젝트는 남녀 80명씩의 학생들이 사회교과에서 학습한 내용을 바탕으로 만든 프로젝트로 애니메이션 유형 프로젝트 80개, 게임 유형 프로젝트 80개이다.

팀 프로젝트는 3명이 한 팀을 이루어 사회과에서 학습한 내용을 게임 형식으로 만든 것으로, 프로젝트를 완성해가는 초기, 중기, 후기의 3번에 걸쳐서 수집되었고, 프로그래밍 과정에서의 컴퓨팅 사고력을 평가하는 데만 사용되었다.

스크래치 프로젝트는 이전 학년에서 스크래치 각 블록에 대한 기초 기능을 이미 학습한 학생들이 스토리보드 작성 및 기획 단계, 자료 수집 단계, 제작 단계, 수정 및 보완 단계를 거쳐 약 4주에 걸쳐서 완성하였다. 애니메이션으로 제작한 개별 스크래치 프로젝트는 세계 여러 나라의 지리적, 문화적 특징 등을 소개하는 것이고, 게임으로 제작한 개별 및 팀 프로젝트는 환경오염을 줄이기 위한 방법을 게임으로 구현한 것이다. 다음 [그림 1]은 학생들의 개별 스크래치 프로젝트의 예이다.



(Fig.1) Student's Scratch Project

3.2 자료 분석 방법

스크래치 프로젝트의 코드 분석을 통해 컴퓨팅 사고력을 평가하는 도구로 스크래치 코드를 분석하는 자동화 도구인 Dr. Scratch를 사용하였고, 그 결과값을 분석하였다.

먼저, Dr. Scratch 분석을 통해 나온 컴퓨팅 사고력 총점의 평균, 표준편차, 최소값, 최대값을 분석하였고, 세 개 등급의 퍼센트를 구하였다.

두 번째로 Dr. Scratch 분석을 통해 나온 컴퓨팅 사고력의 각 요소별 평균, 표준편차, 최소값, 최대값을 분석하였다.

세 번째로 학생들이 제작한 스크래치 프로젝트의 작품 유형(애니메이션, 게임)에 따라 총점과 각 컴퓨팅 사고력 요소별 점수의 평균, 표준편차, 최소값, 최대값을 분석하였고, 두 유형에 따른 차이를 분석하기 위해 독립표본 t 검정을 실시하였다.

네 번째로 학생들의 성별에 따라 스크래치 프로젝트의 전체 점수와 각 컴퓨팅 사고력 요소별 점수의 평균, 표준편차, 최소값, 최대값을 분석하였고, 성별에 따른 차이를 분석하기 위해 독립표본 t 검정을 실시하였다.

마지막으로 스크래치 프로젝트의 제작 과정별(초기, 중기, 후기)로 컴퓨팅 사고력의 총점과 컴퓨팅 사고력 요소별 점수의 평균을 분석하였다.

모든 통계적 분석은 SPSS 18.0 버전을 사용하였다.

4. 연구 결과

4.1 컴퓨팅 사고력의 수준

초등학교 6학년 학생들의 컴퓨팅 사고력의 수준을 분석하기 위해 Dr. Scratch의 총점을 분석한 결과, 컴퓨팅 사고력의 총점은 21점 만점에 최소값은 4점, 최대값은 17점인 것으로 나타났다. 또한, 컴퓨팅 사고력의 평균은 9.73이고, 표준편차는 2.68인 것으로 나타났다.

Dr. Scratch에서는 컴퓨팅 사고력의 총점을 기준으로 0~7점은 기본 단계(Basic), 8~14점은 개발 단계(Developing), 15~21점은 마스터 단계(Master)로 등급을 제시하고 있다[20]. 이같이 학생들의 컴퓨팅 사고력의 등급을 세 개로 나누어 분석한 결과, <Table 2>와 같이 기초단계는 31명, 19.38%, 개발단계는 124명, 77.5%, 숙련단계는 5명, 3.13%로 나타났다.

초등학생의 컴퓨팅 사고력의 등급 중 가장 많이 차지한 것은 개발 단계이고, 그 다음은 기초 단계이고, 소수의 학생들이 마스터 단계인 것으로 나타났다.

<Table 2> Students' Competence Level (n=160)

	Min	Max	M	SD
Score	4.00	17.00	9.73	2.68

Competence Level	persons	percents
Basic	31	19.38
Developing	124	77.50
Master	5	3.13

4.2 컴퓨팅 사고력의 요소별 분석

Dr. Scratch에서는 컴퓨팅 사고력의 요소를 플로우 제어, 데이터 표현, 추상화, 사용자 상호작용, 동기화, 병렬화, 논리의 7개로 평가하는데, 각 컴퓨팅 사고력의 요소별로 분석한 결과를 살펴보면 <Table 3>과 같다.

플로우 제어는 최소값이 1점, 최대값이 3점, 평균은 1.49로 나타났고, 데이터 표현은 최소값이 0점, 최대값이 3점, 평균은 1.22인 것으로 나타났고, 추상화는 최소값이 0점, 최대값이 2점, 평균이 0.99인 것으로 나타났고, 사용자 상호작용은 최소값이 1점, 최대값이 2점, 평균이 1.78인 것으로 나타났고, 동기화는 최소값이 0점, 최대값이 3점, 평균이 1.57인 것으로 나타났고, 병렬화는 최소값이 0점, 최대값이 3점, 평균이 2.10인 것으로 나타났고, 논리는 최소값이 0점, 최대값이 3점, 평균이 0.58인 것으로 나타났다.

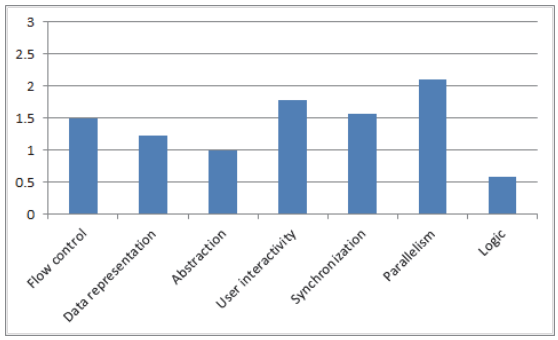
<Table 3> Students' Computational Thinking Dimension (n=160)

Computational Thinking Dimension	Min	Max	M	SD
Flow control	1.00	3.00	1.49	.55
Data representation	0.00	3.00	1.22	.46
Abstraction	0.00	2.00	.99	.15
User interactivity	1.00	2.00	1.78	.41
Synchronization	0.00	3.00	1.57	.87
Parallelism	0.00	3.00	2.10	.94
Logic	0.00	3.00	.58	.91

또한 컴퓨팅 사고력의 요소별로 표준편차가 가장 큰 것은 병렬화로 나타났고, 그 다음이 논리, 동기화 순으

로 나타났다. 표준편차가 가장 낮은 것은 추상화, 사용자 상호작용, 데이터 표현 순으로 나타났다.

[그림 2]에 보듯이 컴퓨팅 사고력의 7개의 요소 중에서 평균이 가장 높은 것은 병렬화이고, 그 다음은 사용자 상호작용, 동기화 순으로 나타났다. 또한, 컴퓨팅 사고력의 요소 중에서 평균이 가장 낮은 것은 논리로 나타났고, 그 다음이 추상화, 데이터 표현 순으로 나타났다.



(Fig.2) Students' Computational Thinking Dimension

4.3 작품 유형별 컴퓨팅 사고력의 분석

학생들이 제작한 스크래치 프로젝트의 유형은 애니메이션과 게임의 두 가지이다. 작품 유형별로 컴퓨팅 사고력을 분석하면 <Table 4>와 같다.

우선, 컴퓨팅 사고력의 등급을 구분 짓는 총점은 애니메이션 유형이 평균 8.95이고, 게임 유형이 평균 10.51로 게임 유형의 총점이 높게 나타났다.

[그림 3]에서 보듯이 컴퓨팅 사고력의 각 요소별로 점수를 분석해 보면, 사용자 상호작용을 제외한 나머지 6개 영역에서 게임 유형에서 컴퓨팅 사고력의 점수가 높게 나타난 것을 알 수 있다.

플로우 제어는 애니메이션에서 평균이 1.33, 게임에서 1.66으로 게임에서 점수가 높게 나타났다.

데이터 표현은 애니메이션에서 평균이 1.15, 게임에서 평균이 1.30으로 게임에서 점수가 높게 나타났다.

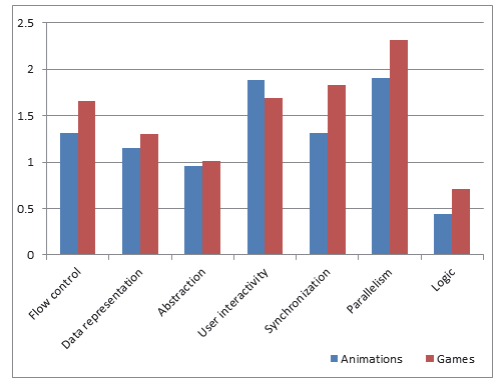
추상화는 애니메이션에서 평균이 0.96, 게임에서 평균이 1.01로 게임에서 점수가 높게 나타났다.

동기화는 애니메이션에서 평균이 1.31, 게임에서 평균이 1.83으로 게임에서 점수가 높게 나타났다.

병렬화는 애니메이션에서 평균이 1.90, 게임에서 평균이 2.31로 게임에서 점수가 높게 나타났다.

논리는 애니메이션에서 평균이 0.44, 게임에서 평균이 0.79로 게임에서 점수가 높게 나타났다.

사용자 상호작용은 애니메이션에서 평균이 1.88, 게임에서 평균이 1.69로 애니메이션에서 점수가 높게 나타났다. 이는 애니메이션 형식으로 제작할 때 키입력, 마우스 블록, 묻고 기다리기, 소리 입력 등을 더 많이 사용한다는 것으로 볼 수 있다.



(Fig.3) Students' Computational Thinking Dimension By Type

다음으로 애니메이션 유형과 게임 유형의 프로젝트 간의 총점의 차이가 유의미 한지를 t 검정으로 분석한 결과, <Table 4>와 같이 t값은 -3.85, 유의도는 .00으로 그 차이가 유의미하였다.

또한, 컴퓨팅 사고력의 7개의 요소별로 애니메이션 유형과 게임 유형 간의 차이가 유의미한지를 분석한 결과, 플로우 제어는 t값이 -4.24, 유의도가 .00으로, 데이터 표현은 t값이 -2.08, 유의도가 .00으로, 추상화는 t값이 -2.02, 유의도가 .04로, 사용자 상호작용은 t값이 2.93, 유의도가 .00으로, 동기화는 t값이 -3.87, 유의도가 .00으로, 병렬화는 t값이 -2.85, 유의도가 .00으로, 논리는 t값이 -1.93, 유의도가 .05로 나타났다. 즉, 논리 요소를 제외한 6개의 요소에서 모두 유의미한 차이가 있는 것으로 나타났다.

논리 요소는 두 개의 유형 모두에서 낮은 점수로 나타났고, 그 차이도 유의미하지 않게 나타났는데, 이를 통해 초등학생에게는 작품 유형에 상관없이 조건문의 사용이 쉽지 않은 것으로 볼 수 있다.

<Table 4> Students' Computational Thinking Dimension By Type

Computational Thinking Dimension	Animations		Games		t	p
	M	SD	M	SD		
Score	8.95	2.46	10.51	2.67	-3.85	.00*
Flow control	1.31	.49	1.66	.55	-4.24	.00*
Data representation	1.15	.42	1.30	.49	-2.08	.04*
Abstraction	.96	.19	1.01	.11	-2.02	.04*
User interactivity	1.88	.33	1.69	.47	2.93	.00*
Synchronization	1.31	.95	1.83	.71	-3.87	.00*
Parallelism	1.90	.96	2.31	.87	-2.85	.00*
Logic	.44	.81	.71	.98	-1.93	.05

*p < .05

4.4 성별에 따른 컴퓨팅 사고력의 분석

학생들의 성별에 따라 컴퓨팅 사고력의 총점과 7개의 요소별 차이의 유의성을 분석해 보면 <Table 5>와 같다.

컴퓨팅 사고력의 총점은 남학생의 평균이 10.34, 여학생의 평균이 9.12로 남학생의 총점이 높게 나타났고, t 검정 결과 t값이 2.93, 유의확률이 .00으로 나타나 성별의 차이가 통계적으로 유의한 것으로 나타났다.

플로우 제어는 남학생 평균이 1.54, 여학생 평균이 1.44로 나타났으나, t값이 1.15, 유의확률이 .25로 통계적으로 유의한 차이를 보이지는 않았다.

데이터 표현은 남학생 평균이 1.30, 여학생 평균이 1.15로 나타났고, t값이 2.08, 유의확률이 .04로 그 차이가 유의한 것으로 나타났다.

추상화는 남학생 평균이 1.00, 여학생 평균이 0.98로 나타났고, t값이 1.00, 유의확률이 .32로 통계적으로 유의한 차이를 보이지는 않았다.

사용자 상호작용은 남학생 평균이 1.82, 여학생 평균이 1.73으로 나타났으나, t값이 1.33, 유의확률이 .18로 유의한 차이를 보이지는 않았다.

동기화는 남학생 평균이 1.75, 여학생 평균이 1.39로 나타났고, t값이 2.68, 유의확률이 .01로 통계적으로 유의한 차이를 보였다.

병렬화는 남학생 평균이 2.23, 여학생 평균이 1.99로 나타났고, t값이 1.16, 유의확률이 .11로 통계적으로 유의한 차이를 보이지는 않았다.

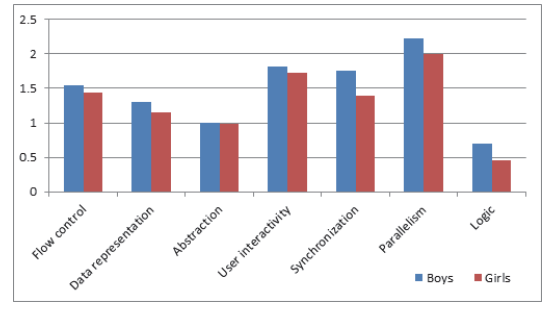
논리는 남학생 평균이 0.70, 여학생 평균이 0.45로 나타났고, t값이 1.75, 유의확률이 .08로 통계적으로 유의한 차이를 보이지는 않았다.

<Table 5> Students' Computational Thinking Dimension By Gender

Computational Thinking Dimension	Boys		Girls		t	p
	M	SD	M	SD		
Score	10.34	2.38	9.12	2.84	2.93	.00*
Flow control	1.54	.55	1.44	.55	1.15	.25
Data representation	1.30	.54	1.15	.36	2.08	.04*
Abstraction	1.00	.16	.98	.16	1.00	.32
User interactivity	1.82	.38	1.73	.44	1.33	.18
Synchronization	1.75	.75	1.39	.95	2.68	.01*
Parallelism	2.23	.90	1.99	.96	1.16	.11
Logic	.70	.92	.45	.88	1.75	.08

*p < .05

남학생과 여학생의 컴퓨팅 사고력의 차이를 종합해 보면, 총점은 남학생이 높고 성별의 차이가 유의하며 [그림 4]와 같이 컴퓨팅 사고력의 7개의 모든 요소에서 남학생의 평균이 높은 것으로 나타났다. 그러나 그 차이가 통계적으로 유의한 것은 데이터 표현과 동기화인 것으로 나타났다.



(Fig.4) Students' Computational Thinking Dimension By Gender

4.5 프로젝트 제작 과정에 따른 분석

학생들이 팀으로 프로젝트를 만들어 가는 과정 중에 초기, 중기, 후기의 세 번에 걸쳐서 프로젝트의 코드를 분석한 결과는 <Table 6>과 같다.

총점의 변화 양상을 살펴보면 Step 1에서 6.94, Step 2에서 8.88, Step 3에서 11.11인 것으로 나타나 프로젝트를 완성해 가면서 점차적으로 점수가 증가하였다.

컴퓨팅 사고력의 각 요소별로 살펴보면, 플로우 제어는 Step 1에서 1.41, Step 2에서 1.47, Step 3에서 1.76으로 나타났고, 초기 점수가 다른 요소들에 비하면 상대적으로 높고 점수 증가의 폭은 약 0.3으로 작은 것으로 볼 때, 프로젝트 제작 초기에 플로우 제어가 많은 부분 완성되는 것으로 분석할 수 있다.

데이터 표현은 Step 1에서 0.88, Step 2에서 1.12, Step 3에서 1.52으로 나타나 점차적으로 점수가 증가하였고, 점수 증가의 폭이 약 0.6정도이다.

추상화는 Step 1에서 0.70, Step 2에서 0.88, Step 3에서 0.94으로 나타나 점차적으로 점수가 증가하였으나 그 폭이 약 0.2정도로 작은 것으로 분석할 수 있다.

사용자 상호작용은 Step 1에서 1.77, Step 2에서 1.88, Step 3에서 1.88로 나타나 초기 점수가 가장 높고, 프로젝트의 중간단계와 후반단계 점수가 같게 나타났다. 이는 프로젝트의 중간 단계에서 키입력이나 묻고 답하기, 소리 입력과 같은 기능은 완성이 되는 것으로 분석할 수 있겠다.

동기화는 Step 1에서 0.47, Step 2에서 1.00, Step 3에서 1.35으로 나타나 점차적으로 점수가 증가하였고 점수 증가 폭이 약 1점으로 높게 나타났다.

병렬화는 Step 1에서 1.06, Step 2에서 1.64, Step 3에서 2.05로 나타나 점차적으로 점수가 증가하였고, 점수 증가 폭이 약 1점으로 높게 나타났다.

논리는 Step 1에서 0.65, Step 2에서 0.88, Step 3에서 1.59로 나타나 점차적으로 점수가 증가하였고, 약 0.9점 정도가 증가하였다.

<Table 6> Students' Computational Thinking Dimension In Programming

Computational Thinking Dimension	Step 1 M	Step 2 M	Step 3 M
Score	6.94	8.88	11.11
Flow control	1.41	1.47	1.76
Data representation	.88	1.12	1.52
Abstraction	.70	.88	.94
User interactivity	1.77	1.88	1.88
Synchronization	.47	1.00	1.35
Parallelism	1.06	1.64	2.05
Logic	.65	.88	1.59

종합해 보면, 프로젝트를 완성해 가는 과정에서 대부분의 컴퓨팅 사고력 요소들의 점수가 높아졌다. 그 중에서도 사용자 상호작용과 플로우 제어는 초기 점수가 높은 반면에 증가폭이 작았고, 추상화는 초기 점수와 증가폭이 모두 작았으며, 데이터 표현, 동기화, 병렬화, 논리는 2배에 가까운 점수 증가폭을 보였다.

5. 결론 및 제언

본 연구에서는 초등학생들의 스크래치 프로젝트 코드 분석을 통해서 컴퓨팅 사고력의 총점과 각 요소별 점수를 확인하고, 성별, 작품 유형에 따라 학생들의 컴퓨팅 사고력을 분석하였으며, 팀 프로젝트 제작 과정에서 컴퓨팅 사고력이 어떻게 변화해 가는지를 분석하였다. 이를 통해 학생들의 스크래치 프로그래밍 수업 방향에 대한 제언 및 학생들의 자기주도적인 학습을 보조할 수 있도록 학생 스스로의 형성평가로서 코드 분석의 활용 방법의 가능성을 탐색하였다.

본 연구의 연구결과를 요약하면, 첫째 대부분의 초등학생들의 컴퓨팅 사고력의 개념 수준은 개발자 수준인 것으로 나타났다. 그러나 기초 단계인 학생들과 소수의 마스터 단계 학생들이 나타난 것으로 볼 때, 교수학습과정에서 학생들의 개인차를 고려한 수업이 구성될 필요가 있다. 초보자와 마스터 단계의 학생이 한 수업에서 같은 과제의 프로그래밍 수업을 하기 보다는 기초와 심화의 과제가 수업 중에 포함되도록 한다면 학습자의 수준에 맞는 학습자 중심의 교육이 이루어질 것이다. 따라

서 학습자의 수준을 고려하여 프로그래밍 수업의 교육 내용 및 활동을 계획하는 것을 제안한다.

둘째, 컴퓨팅 사고력의 요소 중에서 논리와 추상화가 평균이 가장 낮은 것으로 나타났는데, 이는 특히 소프트웨어 교육의 핵심인 논리와 추상화에 대한 개념 교육이 강화될 필요가 있다는 근거가 된다. 초등학생의 수준에서 논리와 추상화는 어려운 개념일 수 있으나 이를 학생들의 인지 수준에 맞도록 구체적이고 단계적으로 접근하여 교수를 강화하여야 하겠다.

셋째, 프로젝트의 유형별로 분석한 결과 애니메이션과 게임 유형 간 컴퓨팅 사고력의 차이가 났다. 이는 작품 유형별 컴퓨팅 사고력의 차이를 밝힌 Moreno-León 외(2017)와 김수환 외(2015)의 연구 결과와 일치한다[8][12]. Moreno-León 외(2017)의 연구에서도 스크래치 프로그래밍 수업을 할 때 애니메이션으로 시작하여, 미술과 음악 프로젝트, 그 다음으로 이야기, 가장 최종적으로 게임 유형의 프로젝트를 제작하도록 제안하고 있다[12]. 이는 게임을 만들 때 학생들이 더 높은 수준의 컴퓨팅 사고력의 개념을 활용한다는 것을 의미한다고 볼 수 있다.

따라서 애니메이션 형태의 프로젝트를 제작하는 것은 초기 프로그래밍 단계에서 접근하고, 프로젝트 제작 경험이 쌓이면 게임 형태의 프로젝트를 제작하기를 제안한다. 즉, 프로그래밍의 경험이 적은 학생들을 대상으로 초기에는 애니메이션을 만들고 보다 더 많은 논리적 사고력과, 객체 간의 상호작용을 고려해야 하는 컴퓨팅 사고력을 요구하는 게임과 같은 형태로 발전해 갈 수 있도록 하는 교육과정의 고려가 필요할 것이다.

넷째, 학생들의 성별에 따라 컴퓨팅 사고력에서 차이가 났다. 이는 여러 선행연구들에서도 SW교육에서 남학생이 여학생보다 더 높은 성취도를 보이는 것으로 보고되고 있다[9][16][19]. 따라서 학생들의 성별에 따라 학습코스를 다르게 제시하는 것을 고려해 볼 수 있다. 그러나 컴퓨팅 사고력의 세부요소에서 차이를 살펴보면, 자료의 표현과 동기화에서만 성차가 유의미한 것으로 나타났기 때문에 이 영역에서 여학생들을 위한 보충학습 및 개별학습을 제안한다. 더불어 성차에 의한 편견을 갖기 보다는 성별에 따른 특징을 파악하고, 이를 보완하기 위한 수업 전략을 통해 동등한 수업의 경험을 제공하는 것이 중요할 것이다.

다섯째, 프로젝트 제작 과정 중에 컴퓨팅 사고력을 분석해 본 결과 전체적으로 점수가 향상되는 것으로 나타났고, 각 요소별로 상이한 변화 양상을 보였다. 이 결과를 통해 프로젝트 제작 과정 중에 교수자의 적절한 개입과 학생 스스로의 코드분석을 통해 자신의 컴퓨팅 사고력 개념의 취약점을 깨닫고 이를 보완하여 프로젝트를 완성하도록 하는 전략이 필요할 것으로 사료된다. 궁극적으로 컴퓨팅 사고력의 향상을 목적으로 하는 프로그래밍 교육에서 학습에 대한 평가와 피드백은 중요한 학습의 한 과정이다. 평가는 결과에만 초점을 맞출 것이 아니라, 학습을 위한 평가(assessment for learning)가 되어야 하며, 더 나아가 학습의 전 과정에서 학습자가 능동적으로 전 과정에 참여하여 학습으로서의 평가(assessment as learning)가 되어야 할 것이다[2][6].

따라서 프로젝트의 코드 분석은 형성적 평가, 자기평가의 도구로 활용될 수 있겠다. 평가 결과를 학생과 교수자는 함께 분석하면서 컴퓨팅 사고력 개념 수준을 파악하고, 교수자는 즉각적인 피드백을 제공하고, 학습자는 이를 다음 학습에 적용하여 학습을 개선해 나갈 수 있다. 또한, 학생들은 이 과정에서 스스로 자신의 프로젝트를 분석하고 개선하면서 메타인지 전략을 활용할 수 있을 것이다.

본 연구는 국내에서 초등학생들의 프로젝트 코드를 분석한 실증적인 데이터에 기반한 연구로서 초등교육 현장에서 프로그래밍 교육을 위한 교수방법의 개선 및 평가 방향에 대한 시사점을 제시한다는 점에서 그 의의를 찾을 수 있다.

참고 문헌

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

[2] Browning. S. F.(2017). Using Dr.Scratch as a Formative Feedback Tool to Assess Computational Thinking. Dissertations of Brigham Young University. https://scholarsarchive.byu.edu/etd/6659.

[3] CSTA & ISTE (2011). Operational definition of computational thinking for K-12 Education. http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf.

[4] Davis, R., Kafai, Y., Vasudevan, V., & Lee, E. (2013). The education arcade: Crafting, remixing, and playing with controllers for Scratch games. Proceedings of the 12th International Conference on Interaction Design and Children, 439 - 442. New York: ACM.

[5] Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational thinking in K-12, In Proceedings of the 19th Annual conference on innovation & technology in computer science education, 57-62, Uppsala, Sweden.

[6] Kim, M. S., & Choi. H., S.,(2018). Fostering Primary Pre-service Teachers' Computational Thinking through Self-Assessment. *Journal of The Korean Association of Information Education, 22*(1), 61-70.

[7] Kim, S. H.(2015). Analysis of Scratch code for Student Assessment about Computational Thinking Capability. *The Journal of Korean Association of Computer Education, 18*(5), 25-34.

[8] Kim, S., Lim, S., & Song, S.,(2015). Analysis about user log for development of online SW education Platform in Korea. *In Proceeding of the Korean Association of Computer Education, 19*(2), 63-67.

[9] Lee. J. M., Jung. Y. J., & Park. H. K.(2017). Differences in Computational Thinking, Creativity, and Academic Interest on Elementary SW Education. *Journal of The Korean Association of Information Education, 21*(4).381-380.

[10] Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia, (46)*, 1-23.

[11] Moreno-León, J., Robles, G., & Román-González, M. (2016). Comparing computational thinking development assessment scores with software complexity metrics. In 2016 IEEE global engineering education conference (EDUCON) (pp. 1040-1045). IEEE.

[12] Moreno-León, J., Robles, G., & Román-González, M. (2017). Towards data-driven learning paths to develop computational thinking with Scratch. IEEE Transactions on Emerging Topics in Computing.

[13] Park, J. Y., Kim, J. H., Kim. S. H., Kim. S. H., & Lee, H. S.(2017). Development of evaluation factors for SW education in elementary and secondary schools. *The Journal of Korean Association of Computer Education, 20*(6), 47-59.

[14] Park, S. J.(2018). Analysis of Computational Thinking Level Through the Scratch Project Analyzation. *Journal of The Korea Association of Information Education, 22*(6), 661-669.

[15] Shim, J. K., & Chae, J. M. (2018). Development of On-line Judge System based on Block Programming Environment. *The Journal of Korean Association of Computer Education, 21*(4), 1-10.

[16] Song, J., Paik, S., & Lee, T. (2009). The effect of robot programming learning considered gender differences on female middle school student's flow level and problem solving ability. *The Journal of Korean Association of Computer Education, 12*(1), 45-55.

[17] Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

[18] Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, *Physical and Engineering Sciences, 366*(1881), 3717-3725.

[19] Yu, B., Kim, J., & Lee, W. (2012). Analysis on the relation between programming achievement and problem solving according to gender. *The Journal of Korean Association of Computer Education, 15*(6), 1-10.

[20] Dr.Scratch. http://www.drscratch.org

저자소개

박 주 연

2003 이화여자대학교 초등교육 학사

2005 이화여자대학교 초등교육학 석사

2015 이화여자대학교 교육공학 박사

2005~현재 이화여자대학교 부속 초등학교 교사

관심분야 : 뉴미디어 기반 학습방법, SW교육, 사고력 교육(창의력, 컴퓨팅 사고력)

E-mail : jy3262@hanmail.net