Journal of The Korean Association of Information Education Vol. 23, No. 5, October 2019, pp. 415-425

http://dx.doi.org/10.14352/jkaie.2019.23.5.415

컴퓨팅사고 프레임워크 기반 초등 소프트웨어교육 경험 분석

이정민* · 이명화

이화여자대학교

요약

본 연구에서는 초등학생의 소프트웨어교육 경험을 Brennan과 Resnick(2012)이 제안한 컴퓨팅사고 프레임워크를 토대로 개념, 수행, 관점 측면에서 체계적으로 분석하고자 하였다. 이를 위하여 서울시 소재 A 초등학교 5학년 전체 학생을 대상으로 1학기 동안 스크래치를 활용한 소프트웨어교육을 실시하고, 소프트웨어교육 경험에 대한 인터뷰를 실시하였다. 인터뷰에는 총 27명의 학생이 자발적으로 참여하였으며, 그 내용을 분석한 결과는 다음과 같다. 첫째, 초등 소프트웨어교육에서 학습자들은 직접 스크래치 프로젝트를 수행하며 컴퓨팅사고를 위한 기본적인 개념을 학습하는 것을 확인할 수 있었다. 둘째, 초등 소프트웨어교육에서 학습자들은 스크래치 프로젝트를 구현하는 과정에서 컴퓨팅사고의 실행이 일어나는 것을 확인할 수 있었다. 셋째, 초등 소프트웨어교육에서 스크래치 프로젝트를 통해 학습자들의 컴퓨팅사고 관점에서 창의적인 표현성과 상호작용적인 연결성의 변화가 있었음을 확인할 수 있었다. 본 연구의 결과는 초등학생의 소프트웨어교육 경험을 질적으로 분석하고 이를 통해 컴퓨팅사고력 증진을 위한 소프트웨어교육 방향을 제시하였다는 점에서 의의가 있다.

키워드 : 초등교육, 소프트웨어교육, 컴퓨팅사고력, 경험, 교육용 프로그래밍 언어

Analyzing Elementary Student Experience on Software Education: Based on Computational Thinking Framework

Jeongmin Lee · Myunghwa Lee

Ewha Womans University

ABSTRACT

The purpose of this study is to analyse of elementary student experience in software education based on computational thinking framework. A total of 27 students (5th grade) were interviewed who took software education during 4 months in A elementary school which located in Seoul. The findings revealed were as follows: First, the elementary learners were able to perform scratch projects and learn basic concepts for computing thinking. elementary students' studied basic concepts for computational thinking by the process of carrying out their Scratch project. Second, elementary learners were able to confirm the execution of computing accidents in the process of implementing scratch projects. Third, elementary students had change in creative expression and interactive connectivity in terms of learners' computing thinking. The result of this study is meaningful in that it analyzes the educational experience of elementary school students qualitatively and suggests the direction of software education for enhancing computing thinking ability.

Keywords : Elementary Education, Software Education, Computational Thinking, Experience, Educational Programming Language

교신저자: 이정민(이화여자대학교)
논문투고 : 2019-07-23
논문심사 : 2019-08-14
심사완료 : 2019-09-04

1. 서론

급격하게 변화하는 새로운 시대의 도래로 인하여 복잡한 문제를 효과적으로 해결하는 컴퓨팅사고력(computational thinking)의 중요성이 강조됨에 따라, 컴퓨팅사고력을 향상시키기 위한 교수학습방법에 대한 다양한 연구가 수행되고 있다. 소프트웨어교육은 다수의 선행연구에서 컴퓨팅사고력을 향상시키는 방안으로 제안되면서[1], 우리나라는 2018학년도부터 초·중등교육에서 소프트웨어교육을 전면 도입하여 공교육의 혁신을 추진하고 있다[2]. 그러나 2018년 소프트웨어교육 연구학교 실태 조사결과에 따르면 초등 소프트웨어교육을 위한 교수학습방법은 미비한 실정으로[3], 실질적으로 초등 소프트웨어교육은 핵심 교수활동과 학습활동을 포함하는 전체적인 구성 체계를 갖춘 구체적인 소프트웨어교육 방향이 제시되어야 할 시점에 있다고 볼 수 있다.

한편, 초등 소프트웨어교육은 건전한 정보윤리의식을 바탕으로 알고리즘과 프로그래밍을 체험하여 실생활의 다양한 문제를 해결하는 능력인 컴퓨팅사고력을 향상시키는 것을 목표로 한다[2]. 즉, 초등 소프트웨어교육은 학습자들의 컴퓨팅사고력을 계발하는 측면에서 설계되어야 할 필요가 있다. 이러한 컴퓨팅사고력을 향상시키기 위해서는 컴퓨터과학의 기본적인 개념과 원리를 바탕으로 문제를 해결하고 시스템을 설계하고 인간의 행동을 이해하는 것이라는 측면에서[4], 학습자들이 학습과정에서 문제를 해결하기 위해 어떤 설계를 바탕으로 어떤 방법으로 해결하며, 이 과정에서 생겨나는 문제 상황에 대하여 어떻게 반응하고, 대처하는지 등 다양한 측면에서 살펴볼 필요가 있다.

컴퓨팅사고 프레임워크(computational thinking framework)는 컴퓨팅사고를 개념, 수행, 관점의 세 가지 차원으로 구분하여 체계적으로 분석하는 모형으로 Brennan과 Resnick(2012)이 제안하였다[5]. 이러한 컴퓨팅사고 프레임워크는 컴퓨팅사고력을 하나의 과정이라는 측면에 초점을 두고 평가하는 대표적인 모형이다. 따라서 본 연구에서는 초등학생의 소프트웨어교육 경험을 해당 컴퓨팅사고 프레임워크를 토대로 체계적으로 분석하여, 이를 바탕으로 초등학생의 컴퓨팅사고력을 향상시키는 양질의 소프트웨어교육을 개발하기 위한 기초자료를 제공하고자 하였다. 이를 위한 연구문제는 다음과 같다.

연구문제 1. 컴퓨팅사고 프레임워크의 개념영역에서 초등학생의 소프트웨어교육 경험은 어떠한가?

연구문제 2. 컴퓨팅사고 프레임워크의 수행영역에서 초등학생의 소프트웨어교육 경험은 어떠한가?

연구문제 3. 컴퓨팅시고 프레임워크의 관점영역에서 초등학생의 소프트웨어교육 경험은 어떠한가?

2. 이론적 배경

2.1 초등 소프트웨어교육과 컴퓨팅사고력

소프트웨어교육이란 학습자들의 의사소통과 지식, 데이터, 상호작용 등의 기본 개념과 정보를 추출하여 프로그램 언어를 구현하는 능력을 개발하는 다차원적의 교육으로[6], 다수의 선행연구를 통해 컴퓨팅사고력을 향상시키는 교수학습방법으로 제안되고 있다[1].

소프트웨어교육은 컴퓨터를 사용하지 않고 다양한 활동을 통해 컴퓨터과학의 원리나 알고리즘을 학습하는 언플러그드 형식의 교육과 교육용 프로그래밍언어를 활용하는 프로그래밍 교육, 소프트웨어를 기반으로 하드웨어를 구현하는 피지컬 컴퓨팅 교육 등을 포함한다.

특히 교육용 프로그래밍 언어 기반 학습은 프로그래밍을 통해 직접 스스로 소프트웨어를 만들어 보면서 컴퓨팅사고를 위한 기본적인 컴퓨팅 개념을 접하고 프로그램을 구현하는 과정에서 추상화와 모듈화, 테스팅, 디버깅, 점진적 증분, 재사용 등 컴퓨팅 수행을 경험하며[7], 컴퓨팅사고력 함양에 있어 효과적인 교육방법으로 보고되고 있다[8].

한편, 우리나라는 초·중등학생들의 컴퓨팅사고력 향상을 위해 2015 표준 초·중등 교육과정 개정을 통해 2018학년도부터 초·중등교육에서 소프트웨어교육을 의무화하였다. 초등교육과정에서의 소프트웨어교육은 컴퓨터 과학적 지식과 기술의 탐구와 더불어 실생활의 문제 해결을 위해 새로운 지식과 기술을 창출하고, 이를 통합적으로 적용하는 컴퓨팅사고력과 협력적문제해결, 정보문화소양을 함양하는 것을 목표로 실과교과에 17시간 이상 포함하였다[2]. 구체적으로는 실과교과의 기술시스템과 기술 활용 영역을 통해 소프트웨어의 이해, 절차적 문제해결, 프로그래밍 요소와 구조, 발명과 문제해결, 개인 정보와 지식 재산 보호, 로봇의 기능과 구조를

학습할 수 있도록 지원한다(<Table 1> 참조).

2.2 컴퓨팅사고 프레임워크 기반 평가

컴퓨팅사고력은 1980년대부터 지금까지 많은 교육자들과 연구자들에 의해 지속적으로 연구되어 왔으나 컴퓨팅사고력의 정의에 대한 논의 가 여전히 진행 중이다[9]. 학자들마다 컴퓨팅사고력의 정의를 제안하고 있으나, 일반적으로 컴퓨터 과학의 기본 개념을 바탕으로 특정 문제를 해결하기 위한 체계를 설계하고 이 과정을 통해 인간의 행동을 이해하는 방법을 제공하는 일종의 분석적 사고로써 컴퓨팅사고력을 정의하는 데 동의한다[4].

한편, 21세기 학습자가 갖추어야 하는 학습역량으로 컴퓨팅사고력이 강조됨에 따라 이를 계발하기 위한 다양한 교수학습방법에 대한 연구가 수행되고 있다. 특히, 컴퓨팅사고력을 측정하기 위한 루브릭, 자기보고식 설문지, 블록기반 학습 프로그램 평가 시스템 등 다양한 평가도구들이 개발하여[10], 학습자들의 컴퓨팅사고력을 분석하고 학습의 효과성을 검증하는 연구들이 활발하게 진행되고 있다.

한편, Brennan과 Resnick(2012)은 양적인 평가도구가 가지는 평가하지 못하거나 학생들의 단편적인 지식이나 결과 중심의 평가에 대한 한계점을 지적하며, 컴퓨팅사고력 프레임워크를 제안하였다[5].

컴퓨팅사고 프레임워크는 개념(시퀀스, 반복문, 병렬, 이벤트, 조건, 연산자, 데이터), 실행(점증적 증분, 테스트와 디버깅, 재사용 및 재활용, 추상화 및 모듈화), 그리고 관점(표현성, 연결성, 반문)으로 구성된다(<Table 2> 참조).

이러한 컴퓨팅사고력 프레임워크는 컴퓨팅사고력을 하나의 과정으로써 과정 중심의 컴퓨팅사고력을 살펴보기 위해 다양한 연구에서 활용되고 있다. 그러나 대부분의 연구는 컴퓨팅사고의 개념부분에 국한되는 경우가 많으며, 이를 질적으로 심도 있게 살펴본 연구는 미비한 실정이다[11]. 따라서

<Table 2> Computational Thinking Framework

Key dimension	Components
concept	sequences, loops, parallelism, events, conditionals, operators, data
practice	incremental and iterative, testing and debugging, reusing and remixing, abstracting and modularizing
perspective	expressing, connecting, questioning

<Table 1> The Contents of Software Education

Core Concept		Contents
Technology system	Communication	Understanding of the basic software concept, procedural problem-solving, the elements and structures of program.
Utilizing Technology	Innovation	Invention and problem-solving, the personal information and knowledge property, functions and structures of robotics

본 연구는 컴퓨팅사고력 프레임워크를 토대로 초등학생들의 소프트웨어교육 경험을 총체적으로 살펴보고자 하였다.

3. 연구방법

3.1 연구 참여자

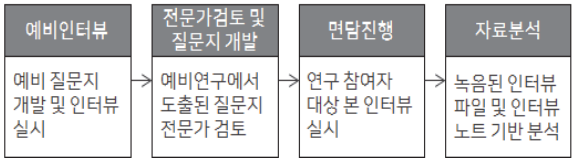
본 연구는 소프트웨어 선도학교인 서울시 소재 A 초등학교의 재학 중인 5학년 전체 학생들을 대상으로 수행되었다. 연구 참여자 선발에 앞서, 연구 참여자들의 연구윤리를 고려하여 연구자의 연구목적을 설명하였다. 본 연구에 참여한 연구 참여자들은 자발적으로 인터뷰에 참여한 학습자들이었으며, 최종적으로 선발된 연구 참여자는 남학생 17명, 여학생 10명으로 총 27명으로 구성되었다(<Table 3> 참조).

<Table 3> Participants

Class	no.	Gender	Class	no.	Gender
A	Student 1	M	B	Student 8	M
	Student 2	M		Student 9	M
	Student 3	M		Student 10	M
	Student 4	M		Student 11	F
	Student 5	M		Student 12	F
	Student 6	F		Student 13	F
	Student 7	F	D	Student 22	M
C	Student 14	M		Student 23	M
	Student 15	M		Student 24	M
	Student 16	M		Student 25	F
	Student 17	M		Student 26	F
	Student 18	M		Student 27	F
	Student 19	M	Total	Male	17
	Student 20	F		Female	10
	Student 21	F		Total	27

3.2 연구절차

본 연구의 참여자들은 한 학기 동안 컴퓨터공학을 전공한 1명의 컴퓨터 교사를 통해 교육용 프로그래밍언어 기반 학습 프로그램인 스크래치(Scratch)를 활용한 소프트웨어교육을 학습하였다. 컴퓨터교사는 소프트웨어교육을 다룰 때, 컴퓨팅사고 프레임워크에서 강조하고 있는 개념, 실행, 관점 등을 포함하였다. 아울러, 모든 연구 참여자들은 국내에서 개발된 교육용 프로그래밍언어 기반 학습 프로그램인 엔트리(Entry)를 학습한 경험이 있으며, 다음에 따라 수행되었다(Fig. 1. 참조). 첫째, 선행연구와 컴퓨팅사고력 프레임워크에 기초하여 개발한 예비 문항을 토대로 예비 연구 참여자들을 대상으로 예비 인터뷰를 실시하여 인터뷰 질문지 초안을 개발하였다.



(Fig. 1) Research procedures

둘째, 최종 질문지 개발을 위해 질적연구 전문가와의 도출된 인터뷰 질문지 초안에 대한 타당성을 검토하였다(<Table 2> 참조).

<Table 2> the examples of interview questions

	the examples of interview question
Concept	1. What blocks did you use to develop your Scratch project? 2. Why did you decide to use each blocks? and do you know the useful of each blocks?
Practices	1. Can you explain the process of your Scratch project development 2. During the development, how did you solve the problems if there were any errors on your Scratch project.
Perspective	Could you tell me what you felt such as your feelings, good or bad experiences and so on since you started learning Scratch?

셋째, 소프트웨어교육이 진행되는 마지막 날, 컴퓨터교사의 협조를 받아 소프트웨어교육에 참여한 모든 연구 참여자들을 중 희망하는 참여자들을 대상으로 인터뷰를 실시하였다. 본 인터뷰는 제시한 질문에 대한 연구 참여자들의 반응에 따라, 추가적으로 다른 경험들을 이야기해 달라고 요청하는 방식으로 진행하였다. 마지막으로, 본 인터뷰의 내용은 녹음기와 주요 내용을 기록한 노트를 토대로 분석하였다.

3.3 자료분석방법

본 연구에서는 근거이론에서 기본적 분석방법인 질문하기와 비교하기[12]를 통하여 연구대상자가 어떤 경험을 하고 있는지, 그 경험의 의미는 무엇인지, 언제, 어디서, 얼마나, 어떻게, 왜, 어떤 결과를 가지고 행동하는지에 대하여 분석하였다. 이와 함께 Guba와 Lincoln(1981)[13]이 제시한 진실성(true), 적합성(applicability), 일관성(consistency), 중립성(neutrality)에 근거하여 타당도를 검증하였다.

4. 연구결과

4.1 컴퓨팅사고 개념

컴퓨팅사고 개념에서는 학습자가 소프트웨어교육에서 컴퓨팅사고를 위한 기본적인 개념을 학습하는 과정과 이해도를 살펴보고자 하였다. 이를 위해, 학습자들이 수행하였던 스크래치 프로젝트에 초점을 두고, 어떤 블록을 활용하였는지, 왜 블록을 사용하였는지 등의 질문을 통해 그 과정에서 활용한 컴퓨팅사고를 위한 개념에 대하여 이야기 하도록 유도하였다. 그 결과, 학습자들의 컴퓨팅사고를 위한 개념을 다음과 같은 스크래치 프로젝트를 구현하는 과정에서 습득하는 것을 확인할 수 있었다.

4.1.1 시퀀스

소프트웨어교육에서 학습자들은 특정 순서에 따라 배열하는 시퀀스[5]에 대한 개념을 다음의 과정을 통해 학습하는 것을 확인할 수 있었다.

상어가 물고기 잡아먹는 것 있는데 상어가 물고기를 잡아먹으면 물고기가 다시 나타나고 나타

가고 했어요. 똑같은 동작을 계속할 수 있어서 좋아요 무한반복하기로 안 하면은 몇 번만 하고 나서 멈추니까… (C반, 학생 15, 남)

난수를 이용해가지고 한번 만들어 봤는데 무한반복하기를 하면요. 만약에 클릭되었을 때 무한반복하기 해가지고 10만큼 움직이게 했으면요. 그런데 그게 굴러가다가 없어지면 안 되니까 어떨 때는 무한을 쓰기도 하고 어떨 때는 10번 반복하기를 쓰기도 하는데 제가 요즘 자주 쓰는 게 무한반복하기 같아가지고… (C반, 학생 19, 남)

이와 같이 학습자들은 반복하는 구문을 학습하는 과정에서 시퀀스에 대한 개념을 습득하는 것을 확인할 수 있었다.

4.1.2 반복문

소프트웨어교육에서 학습자들은 프로그램에서 어떤 조건이 만족되고 있는 동안 또는 종료 조건이 성립할 때까지 반복 실행되는 명령의 집합인 반복문[5]에 대한 개념을 다음의 과정을 통해 학습하는 것을 확인할 수 있었다.

무한 반복하기 한번 반복하기 이런 걸 숫자를 쳐 가지고 반복하는 숫자를 정해서 원하는 만큼 반복하기 해서 그런 걸 프로그래밍 했어요. (C반, 학생 14, 남)

반복되는 부분이라면 몇 초 동안 몇 번 움직이기나 벽에 튕겨나가기 같은 그런 기본적인 코드나 시작버튼을 클릭했을 때 같은 그런 코드를 많이 이용했었어요. 사용해보니 신기했죠. (D반, 학생 27, 여)

이와 같이 학습자들은 프로젝트에서 요구되는 반복하는 구문을 구현하는 과정에서 반복문에 대한 개념을 습득하는 것을 확인할 수 있었다.

4.1.3 병렬 연산

소프트웨어교육에서 학습자들은 크고 복잡한 문제를 작게 나눠 동시에 병렬적으로 해결하는 병렬 연산에 대한 개념을[5] 다음의 과정을 통해 학습하는 것을 확인할 수 있었다.

아! 캐릭터 클릭해서 하는 것 같은 것 만들 때… 상어가 물고기 잡을 때, 상어는 잡아먹을 때로 하고 물고기는 다른 거 만들어서 조건을 다르게 해줬어요. 상어가 잡아야하니까 제 마우스를 따라다니게 하고 물고기는 피해야하니까… (A반, 학생 2, 남)

상어 코딩 같은 건 물고기가 난수에 따라서 움직이고 상어가 물고기를 닿으면 잡아먹어야 하는 데 그게 어떤 색이랑 닿으면 이런 조건 해 가지고 해야 됐어요. 예를 들면 물고기 색이 노란색인 상어의 색이 닿았을 때 물고기는 숨기고 상어는 다시 입을 닫아야 하는 그런… (B반, 학생 13, 여)

이와 같이 학습자들은 프로젝트에 등장하는 여러 캐릭터들의 역할과 상황을 구현하는 과정을 통해 병렬 연산에 대한 개념을 습득하는 것을 확인할 수 있었다.

4.1.4 이벤트

소프트웨어교육에서 학습자들은 프로그램의 동작을 감지하고 처리하는 이벤트[5]에 대한 개념을 다음의 과정을 통해 학습하는 것을 확인할 수 있었다.

상어 코딩 같은 건 물고기가 난수에 따라서 움직이고 상어가 물고기를 닿으면 잡아먹어야 하는 데 그게 되게 어떤 색이 어떤 색이랑 닿으면 이런 조건 해 가지고 해야 됐어요. 예를 들면 물고기 색이 노란색인 상어의 색이 닿았을 때 물고기는 숨기고 상어는 다시 입을 닫아야 하는 그런… (B반, 학생 13, 여)

이와 같이 학습자들은 프로젝트에 등장하는 여러 캐릭터들의 역할과 상황을 구현하는 과정을 통해 이벤트에 대한 개념을 습득하는 것을 확인할 수 있었다.

4.1.5 조건문

소프트웨어교육에서 학습자들은 주어진 조건이 참이냐 거짓이냐에 따라 다른 명령을 처리하도록 하는 조건

문[5]에 대한 개념을 다음의 과정을 통해 학습하는 것을 확인할 수 있었다.

> 아! 캐릭터 클릭해서 하는 것 같은 것 만들 때… 상어가 물고기 잡을 때, 상어는 잡아먹을 때로 하고 물고기는 다른 거 만들어서 조건을 다르게 해줬어요. 상어가 잡아야하니까 제 마우스를 따라다니게 하고 물고기는 피해야하니까… (A반, 학생 2, 남)

> 필요한 조건만 사용하고 조건이 사용한 적이 저는 꽤 있는 거 같아요. 제가 집에서 게임도 만들어보고 그런데 너무 프로그램이 너무 복잡하니까요. 조건을 사용해서 넣어봤더니 조건을 사용해봤더니 확 줄어서 갖고 그때부터 프로그램을 그렇게 많이 사용하지 않고 조건을 사용해갖고 단숨에 프로그램을 만들었었어요. 무엇에게 무엇이 닿았는가 아니면 마우스 포인터에 닿았는가 아니면 스페이스바를 클릭했는가. 그런 것 가지고… (C반, 학생 18, 남)

이와 같이 학습자들은 프로젝트에 등장하는 여러 캐릭터들의 역할과 상황을 구현하는 과정을 통해 조건문에 대한 개념을 습득하는 것을 확인할 수 있었다.

4.1.6 연산자

소프트웨어교육에서 학습자들은 프로그램에서 제시되는 수학적인 개념을 구현하는 연산자[5]에 대한 개념을 다음의 과정을 통해 학습하는 것을 확인할 수 있었다.

> 함수 값이나 그런 것을 지정해 줄 수 있어요. 그리고 이름을 지어주니까 변수를 많이 사용하더라도 코딩을 하다보면 헷갈릴 수 있지만 제가 직접 지어준 이름이니까 기억하기 좋아요. (A반, 학생 3, 남)

> 변수 썼을 때 많긴 많은데 점수 같은 것 만들 때 만약에 한 마리 저런 게임(만들던 레이싱게임 화면을 가리킴)에서 한번 도착점까지 갔을 때 몇 점씩 얻고 그 다음에 또 만약에 게임오버가 되면 초기화되거나 그런 식으로… (D반, 학생 14, 남)

이와 같이 학습자들은 변수의 값을 구현하는 과정을 통해 구현하는 과정을 통해 연산자에 대한 개념을 습득하는 것을 확인할 수 있었다.

4.1.7 데이터

소프트웨어교육에서 일종의 프로그램에서 특정 값이나 연산의 결과 값을 의미하는 데이터에 대한 개념을[5] 다음의 과정을 통해 학습하는 것을 확인할 수 있었다.

> 함수 값이나 그런 것을 지정해 줄 수 있어요. 그리고 이름을 지어주니까 변수를 많이 사용하더라도 코딩을 하다보면 헷갈릴 수 있지만 제가 직접 지어준 이름이니까 기억하기 좋아요. (A반, 학생 3, 남)

> 변수 썼을 때 많긴 많은데 점수 같은 것 만들 때 만약에 한 마리 저런 게임(만들던 레이싱게임 화면을 가리킴)에서 한번 도착점까지 갔을 때 몇 점씩 얻고 그 다음에 또 만약에 게임오버가 되면 초기화되거나 그런 식으로… (D반, 학생 14, 남)

> 버블점령에서는 변수나 그런 것을 만들어야 편하더라고요. 그때는 그 변수를 I인가? J로 써야 편하다고 하셔서 I로 썼어요. 변수를 사용하면 우선 어떤 것에 초점을 둬야 하는데 그 변수라는 것으로 큰 것을 압축시키니까 스크립트도 짧아지고 좋은 것 같아요. (D반, 학생 25, 여)

이와 같이 학습자들은 변수의 값을 구현하는 과정을 통해 데이터에 대한 개념을 습득하는 것을 확인할 수 있었다.

4.2 컴퓨팅사고 실행

컴퓨팅사고 실행에서는 학습자가 소프트웨어교육에서 컴퓨팅사고를 위한 기본적인 개념을 토대로 컴퓨팅사고의 실행이 일어나는 과정을 살펴보고자 하였다. 이를 위해, 학습자들의 스크래치 프로젝트 과정과 관련한 질문을 통해 컴퓨팅사고의 실행이 일어나는 과정에 대하여 이야기 하도록 유도하였다. 그 결과, 학습자들의 컴퓨팅사고의 실행은 다음과 같은 스크래치 프로젝트를

구현하는 과정에서 일어났음을 확인할 수 있었다.

4.2.1 점증적 증분

소프트웨어교육에서 학습자들은 프로젝트를 점진적으로 향상시키며 구체화하는 컴퓨팅사고의 실행인 점진적 증분이[5] 다음의 과정을 통해 실행되었음을 확인할 수 있었다.

> 왠지 프로그램을 만들 때에는 한 번에 안 되고… 여러 번 해보면서 다시 버그를 고쳐야지 그게 된다는 걸 알았어요. (A반, 학생 7, 여)

> 코딩을 할 때 '어떤 코딩을 해야 하지?' 그런 걸 정하는 게 프로그래밍 할 때 좀 더 도움이 되는 것 같아요. (B반, 학생 13, 여)

> 그런데 너무 많이 만들다 보면요 기억을요 또 까먹게 되잖아요. 그러니까 중간에 또 한 번 바꿔보고 그래도 안 되면 그때 싹 한번 비워야겠다, 그냥 두세 번, 한번 중간에 한번 바꿔보고 그래도 안 되면 그 프로그램을 아예 지워버리겠다. 그러니까 이런 거 줄여서 말하면 '중간에 다시 한 번 검토해보자'… (D반, 학생 27, 여)

이와 같이 학습자들은 스크래치 프로젝트를 설계하는 과정이나 구현과정에서 발생하는 오류들을 수정하는 과정에서 점진적 증분이 일어났음을 확인할 수 있었다.

4.2.2 테스트와 디버깅

소프트웨어교육에서 학습자들은 프로그램을 테스트하고 발생하는 오류들을 수정하는 테스트와 디버깅이[5] 다음의 과정을 통해 실행되었음을 확인할 수 있었다.

> 컴퓨터 선생님의 도움을 받았어요. 선생님이 빠르게 하셔서 정확히는 못 봤지만, 틀렸던 부분을 고쳤었어요. (A반, 학생 6, 여)

> 처음부터 다시 생각해보고 뭐가 잘못되었는지 보고… 뭐가 잘못됐다고 생각되면 그걸 다른 걸로 어떻게 고칠지 생각해 봤어요. (C반, 학생 21, 여)

> 처음에 했던 게 독수리가 움직이면서 하늘을 날아다니는 거였거든요 제가 할 때 그때 제가 독수리한테 분명히 코딩을 했는데 배경이 움직이는 거예요 그래가지고 놀라가지고 '아! 뭐야' 계속 그랬던 적이 있는데 배경에 코딩을 했더라고요. 그래 가지고 그거를 복사해가지고 독수리한테 붙이고 그 배경의 코딩을 삭제해서 다시 그렇게 문제를 해결한 적도 있었어요. (D반, 학생 22, 남)

> 처음부터 오작동이 났을 거라고 생각하기 때문에 그 프로그램을 삭제하고 아예 새로 만들어요. 똑같은 걸로 중간에 고치다가 오히려 더 꼬일 수 있기 때문에 저는 그냥 지우고 다시 시작해요. (D반, 학생 27, 여)

이와 같이 학습자들은 스크래치 프로젝트를 테스트하고 오류를 수정하는 시행착오를 겪는 과정에서 테스트와 디버깅이 일어났음을 확인할 수 있었다.

4.2.3 재사용 및 재활용

소프트웨어교육에서 학습자들은 필요한 자원을 탐색하는 과정에서 공개 자원의 재사용과 재활용이 다음의 과정을 통해 실행되었음을 확인할 수 있었다.

> 비슷한 것을 다른 곳에서 따오고 그랬던 것이 도움이 된 것 같아요. 스크래치를 보관소에 저장되어 있는 캐릭터나 배경은 적잖아요. (B반, 학생 13, 여)

> 온라인에 들어가서 다른 친구들이 만들어놓은 것 보는 것이 좋았어요. (C반, 학생 14, 남)

> 근데 내가 그거를 보완하기 위해서는 내가 직접 그리거나 그래야 하는데 패드가 있지만 쓰기가 불편하더라고요. 그래서 다른 곳에서 따오고 하게 된 거죠. (D반, 학생 24, 남)

이와 같이 학습자들은 스크래치 프로젝트를 구현하기 위하여 필요한 자원을 탐색하는 과정에서 공개 자원의 재사용과 재활용의 활동이 일어났음을 확인할 수 있었다.

4.2.4 추상화 및 모듈화

소프트웨어교육에서 학습자들은 프로젝트를 분해하고 조합하는 컴퓨팅사고의 실행인 추상화 및 모듈화가 다음의 과정을 통해 실행되었음을 확인할 수 있었다.

> 처음부터 다시 쪼개서 하나하나씩 보면서 다시 붙였어요. (A반, 학생 1, 남)

> 다시 버그를 찾고 다시 검토하고서는 안 되면 진짜 운이 없으면 하나씩 빼고서 다시 조립해보고 뭔가 파트별로 있잖아요? 그… 뭐하면 뭐하기 뭐하면 뭐하기 그게 몇 십 개씩 있잖아요? 그 파트를 하나씩 본 다음에 안 되면 다시 보고 그랬죠. (A반, 학생 7, 여)

> 처음으로까지는 안가고 잘못된 블록을 찾아서 그 블록을 지웠죠. 잘못된 블록을 지우고 선생님이 했던 거를 자세히 보면서 그렇게 했죠. (D반, 학생 9, 남)

> 처음부터 뭐가 잘못 되었는지 다 봐요 너무 길면 어느 정도 떼서 거기까지 해보고 맞는지 그게 되면 그 다음부터 문제가 있는 거고 그게 안 되면 거기서 문제가 있는 거고 그런 식으로 계속… (D반, 학생 13, 여)

이와 같이 학습자들은 스크래치 프로젝트를 구현하며 발견된 오류를 수정하는 과정에서 추상화 및 모듈화가 일어났음을 확인할 수 있었다.

4.3 컴퓨팅사고 관점

컴퓨팅사고 관점에서는 학습자가 스크래치 프로젝트를 통해 변화하는 컴퓨팅사고의 관점을 살펴보고자 하였다. 이를 위해, 스크래치 프로젝트를 실습하는 과정에서 느낀 점, 좋았던 점, 일어났던 일들 등 생각나는 것들과 같은 질문을 통해 자신에 대한 이해와 다른 사람들과의 관계 및 우리 주변의 기술에 대하여 이야기 하도록 유도하였다. 한편, 프로그래머의 자질을 깨닫고 일상생활에서 소프트웨어교육을 활용하는 것과 관련하여 학습자들이 스스로 자신에게 질문하는 것을 의미하는 반문은[5] 한 학기동안 진행한 소프트웨어교육을 통해 형성되기에 다소 어려운 측면이 있다고 사료되어 인터뷰 문항에서 제외하였다.

4.3.1 표현성

> 자기가 마음대로 움직일 수 있어요 예를 들면 저렇게 틀에 갇혀있지 않고 자신이 원하는 대로 블록을 이렇게 만들어서 갖다 놓으면 자신이 원하는 대로 움직일 수도 있고 해서 그런 것들이 좋은 것 같아요. (A반, 학생 6, 여)

> 코딩에는 여러 가지 방법이 있잖아요. 함수 값을 일정하게 정하는 것도 아니고 자신이 넣어보고 틀리면 다시 해보면 되고, 잘 모르겠으면 선생님한테 도움을 요청해도 되니까. 코딩은 제가 잘 할 수 있고, 더 많은 것을 해볼 수 있어서 제가 창의적으로 변하는 것 같아요. (B반, 학생 10, 남)

> 저희가 스크래치 시험을 볼 때 저희가 직접 만드는 것이 괜찮았었어요. 어쨌든 다 좋았어요. 프로그램을 많이 만들어서 창의적으로 된 것 같은 기분이 들었어요. (D반, 학생 13, 여)

이와 같이 스크래치 프로젝트 경험으로 인하여 컴퓨팅사고 관점에서 학습자들이 자신의 표현성이 창의적으로 변화하였다고 인식하고 있음을 발견할 수 있었다.

4.3.2 연결성

> 다른 친구랑 하면은 아마도 제가 가지고 있는 것과 다른 친구가 할 수 있는 스크래치 기술이 있으니까 다른 친구랑 하면은 좀 더 다양한 코딩을 할 수 있을 것 같아요. 저는 할 수 없는 코딩을 다른 친구는 그런 거를 알아서 할 수 있잖아요. 그런 거를 알 수 있을 것 같아요. (A반, 학생 2, 남)

> 혼자 하는 것은 제 생각만 반영되는 건데, 친구들과 함께 있으면 의견을 모아가지고 좀 더 좋은 것을 만들거나 더 좋은 아이디어를 이렇게 만들 수 있기 때문에 친구들하고 협동하면 더 좋을 것 같다고 생각해요. (B반, 학생 12, 여)

선생님이 저보고 좀 잘 아니까 애들 도와달라고 하는데 도와주면 저도 기분 좋고 다른 애들도 고맙다고 했어요. 스크래치 할 때 어려울 때도 있잖아요. 애들이 계속 똑같은 걸 물어보니까 귀찮을 때도 있긴 한데 그래도 친구들을 도와주면 기분이 좋아요. (C반, 학생 20, 여)

내가 만든 프로그램을 공유해서 다른 사람들도 보고 그거에 대한 가끔씩은 들어가 있는 클랜에서 뭔가를 공유하고 그러면 제 작품이 좀 유명해졌거든요. 그러면 사람들이 와서 댓글을 달아주고 그럼 댓글을 보면서 버그 수정 해달라고 하면 버그 수정하고… 공유를 하면 내가 모르고 상대방이 아는 버그를 수정해주면 좋죠. 그런 것을 알려주면 도움이 많이 되죠. (D반, 학생 25, 여)

이와 같이 스크래치 프로젝트 경험으로 인하여 컴퓨팅사고 관점에서 학습자들의 연결성이 상호작용적으로 변화하였음을 발견할 수 있었다.

5. 결론 및 논의

본 연구는 초등 소프트웨어교육에서 컴퓨팅사고력 향상을 위한 시사점을 도출하기 위하여 초등학생의 소프트웨어교육 경험을 Brennan과 Resnick(2012)의 컴퓨팅사고 프레임워크를 토대로 체계적으로 살펴보고자 하였다. 본 연구의 결과가 시사하는 바에 대한 논의와 연구의 결론은 다음과 같다.

첫째, 초등 소프트웨어교육에서 학습자들은 직접 스크래치 프로젝트를 수행하며 컴퓨팅사고를 위한 기본적인 개념을 학습하는 것을 확인할 수 있었다. 다시 말하자면, 조건문과 이벤트, 반복문 등 다양한 컴퓨팅사고의 개념을 이론적인 지식을 토대로 학습하기 보다는 개인의 프로젝트를 구현하는 과정에서 자연스럽게 습득하고 있었다. 특히, 학습자들은 각각의 컴퓨팅사고의 개념에 대하여 특정 프로그램을 개발하는 과정을 예시로 설명하였는데, 이러한 관점에서 초등 소프트웨어교육은 이론중심보다는 문제를 해결하는 과정에서 자연스럽게 학습할 수 있도록 실습중심으로 설계되어야할 필요가 있다.

둘째, 초등 소프트웨어교육에서 학습자들은 스크래치 프로젝트를 구현하는 과정에서 컴퓨팅사고의 실행이 일어나는 것을 확인할 수 있었다. 구체적으로는 프로젝트를 설계하고, 구현하는 과정에서 테스트 및 디버깅을 하며 최종 프로그램을 완성하는 과정을 통해 본인의 프로젝트를 점진적으로 향상시키며, 추상화와 모듈화 등의 컴퓨팅사고의 실행이 일어났다. 이러한 결과는 프로젝트를 설계하고 이를 구현하는 과정에서 프로젝트를 테스트하고 디버깅하는 과정은 실질적인 컴퓨팅사고의 실행을 돕는 필수적인 과정임을 시사한다. 따라서 초등 소프트웨어교육은 설계와 테스트, 오류의 수정의 과정을 거칠 수 있는 과정을 제공할 필요가 있다. 그러나 테스트와 디버깅 과정에서 학습자가 지속적인 어려움을 경험할 경우 소프트웨어교육에 대한 흥미를 잃을 수 있으므로[14], 분석, 설계, 프로토타입 제작, 테스트 및 평가의 일련의 프로세스인 디자인사고(Design Thinking)와 같은 교수·학습 지원전략을 마련하여 학습자의 참여를 지속적으로 독려할 필요가 있다.

셋째, 초등 소프트웨어교육에서 스크래치 프로젝트를 통해 학습자들의 컴퓨팅사고 관점에서 창의적인 표현성과 상호작용적인 연결성의 변화가 있었음을 확인할 수 있었다. 이러한 결과는 소프트웨어교육은 학습자들의 컴퓨팅사고력을 향상시킬 뿐만 아니라 창의성과 협업에 있어서도 긍정적인 역할을 한다는 것을 시사한다. 그러나 교육용 프로그래밍 언어 기반 학습은 학습자가 창의적으로 자신만의 언어로 프로그램을 구성하며 창의성을 키우기 적합한 방법이지만, 동료학습자와의 협업에서는 각자의 창의성에 대한 존중이 요구되며 공동 작업이 어렵다는 점에서 한계를 가진다. 따라서 교육용 프로그래밍언어를 활용한 소프트웨어교육을 설계할 경우, 학습자들이 서로 프로젝트를 공유하고 피드백을 제공하여 프로젝트에 대한 성찰하는 과정과 같이 협업에 대한 긍정적인 인식을 제공하는 과정을 포함하여 구성한다면 컴퓨팅사고력을 포함하여 창의성, 협업 모두를 향상 시킬 수 있는 효과적인 소프트웨어교육을 제공할 수 있을 것이다.

본 연구의 제한점을 토대로 다음과 같은 후속 연구를 제안하고자 한다. 첫째, 본 연구에서 활용한 컴퓨팅사고 프레임워크는 해외에서 개발된 것으로 국내 소프트웨어교육 환경을 충분히 반영하지 못할 수 있으므로 국내 교육환경을 고려한 컴퓨팅사고력을 분석할 수 있는 컴

퓨팅사고 프레임워크 개발이 요구된다. 둘째, 본 연구는 컴퓨팅사고 프레임워크를 토대로 반구조화 인터뷰 질문지를 개발하여 진행하였으나, 후속연구에서는 보다 다양한 관점으로 컴퓨팅사고력을 분석하기 위해 산출물을 함께 분석하거나 관찰 등을 함께 사용함으로써 컴퓨팅사고력의 다양한 측면을 심층적으로 분석할 필요가 있다. 셋째, 본 연구에서는 학습자들의 컴퓨팅사고력을 살펴보기 위한 인터뷰를 1학기의 소프트웨어교육이 모두 마친 후 실시하였다. 이는 관점에 변화에 있어서 정확한 측정이 어려웠을 수 있음을 시사하며, 후속연구에서는 전반적인 과정을 참여관찰하며 중간에도 인터뷰를 실시하고 혼합 분석을 통해 총체적으로 살펴본다면 의미있는 결과를 도출할 수 있을 것으로 사료된다.

이와 같은 제한점에도 불구하고, 초등 소프트웨어교육에서 컴퓨팅사고력 향상을 위한 교수·학습 지원전략을 논의함으로써, 향후 초등 소프트웨어교육에서 컴퓨팅사고력 증진을 위한 전략을 수립하기 위한 기초자료를 제공하였다는 점에서 연구의 의의가 있다.

참고문헌

[1] Yoon, I. (2009). An Analysis of Logical thinking in Programing Education Based on Elementary Student Scratch Tasks. *The Journal of Creative Informatics & Computing Education, 3*(1), 1-7.

[2] The Korean ministry of Education (2015). *General and Curriculum in Elementary and Secondary Education*. Seoul: the ministry of Education.

[3] Seo, S. (2018). *Analysis the status and effectiveness of Computer Science Education School in 2018*. Seoul: KERIS.

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

[5] Brennan, K., & Resnick, M. (2012). Using artifact-based interviews to study the development of computational thinking in interactive media design. In *annual American Educational Research Association meeting*, Vancouver, BC, Canada.

[6] Sysło, M. M., & Kwiatkowska, A. B. (2013, February). *Informatics for all high school students. In International conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 43-56). Springer, Berlin, Heidelberg.

[7] Choi, H. (2018). Domestic Literature Review on Computational Thinking Development through Software Programming Education. *Journal of Educational Technology, 34*(3), 734-774.

[8] Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming : What is next for K-12? *Computers in Human Behavior, 41*, 51-61.

[9] Grover, S., & Pea, R. (2013). Computational thinking in K‐12: A review of the state of the field. *Educational researcher, 42*(1), 38-43.

[10] Jeon, S. J. & Han, S. K. (2016). Descriptive Assessment Tool for Computational Thinking Competencies. *The Journal of Creative Informatics & Computing Education, 20*(3), 255-26

[11] Choi, H., Jeong, I., & So, H. (2014). Computational Thinking Framework-based Analysis of Afterschool Scratch Team Project Experiences. *The Journal of Creative Informatics & Computing Education, 18*(4), 549-558.

[12] Strauss, A., & Corbin, J. (1998). *Basics of qualitative research techniques*. Thousand Oaks, CA: Sage publications.

[13] Guba, E. G., & Lincoln, Y. S. (1981). *Effective evaluation: Improving the usefulness of evaluation results through responsive and naturalistic approaches*. CA: Jossey-Bass.

[14] Bandura, A. (1977). Self-efficacy: toward a unifying theory of behavioral change. Psychological review, 84(2), 191-215.

저자소개

이 정 민

2001 이화여자대학교 교육공학과 학사

2003 이화여자대학교 교육공학과 석사

2009 플로리다주립대 교육심리 및 교육공학박사

2009 퍼듀대학교 연구원

현재 이화여자대학교 교육공학과 부교수

관심분야: 창의적 문제해결, 모바일러닝, 소셜러닝

E-mail: jeongmin@ewha.ac.kr

이 명 화

2011 명지대학교 컴퓨터공학과 학사

2015 이화여자대학교 교육공학과 석사

현재 이화여자대학교 교육공학과 박사과정

관심분야: 교수설계, 공학교육, 소프트웨어교육

E-mail: myunghwa523@gmail.com