Journal of The Korea Society of Computer and Information
Vol. 20 No. 11, pp. 191-197, November 2015

www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2015.20.11.191

The Improvement Effectiveness of Computational Thinking through Scratch Education

Soo-Bum Shin *

Abstract

Recently, it has been activated the software education or coding education for the improvement of the Computational Thinking (CT) ability at home and abroad. Also the CT has influence on courses of Computer Science in the college levels. It has been introduced and the number of cases of using it to general K12 education has increased. However, the research on the software education's influence on the CT was still lacking. So In this paper, we proposed this study has been conducted on how Scratch education in the elementary school level influenced the ability of the CT. And we proposed software education can improve the ability of CT.

First, we provided the theoretical base of the software education and evaluation process through analysis of computational thinking ability. A core analysis content of the CT is broader than algorithmic thinking and can be achieved without using computer. It includes abstract, algorithmic, logical, and measurable thinking. Second, we made efforts to improve the characteristics of the software education with categorization. Finally, we have managed the software education using Picoboard with Scratch and flowchart within 15 weeks based on these theocratical research. An examination of the effectiveness was committed to understand, analyze, and develop strategies of problem solving. It is designed as a strategy of problem solving before and after the software lesson. The result of the software education has improved authentically in all areas without the need to design a strategy for problem solving.

▸ Keyword : Software Education, Computational Thinking

I. Introduction

현대사회에서 소프트웨어가 모든 산업에서 중요한 요소가 되었다. 이에 전 세계의 주요 선진국인 미국, 영국, 프랑스와 중국, 인도, 이스라엘 등의 나라에서는 소프트웨어 교육을 강화하여 소프트웨어 산업 인력 육성을 강화하고 있다[1].

이에 국내에서도 국가경쟁력 강화와 소프트웨어 산업 인력을 확대하기 위해 초중등 교육에서도 소프트웨어 교육을 추진하고 있다. 하지만 실제 소프트웨어 교육을 수행한 후에 교육성과를 측정하고 교육과정을 수정 보완하는 연구는 미흡한 수준이다. 그리고 지금까지 실시하고 있는 초등학교에서의 소프트웨어 교육은 정규교육과정 구조 속에서 이루어지는 것이 아니라 비교과활동, 재량활동, 취미특기 활동의 형태로 이루어지고 있다. 그런데 2018년부터는 초등학교에서도 정규교육과정에서 프로그래밍교육이 시작되고 있기 때문에 교육과정의 구성과 평가체제에 더욱 노력해야 한다.

특히 컴퓨팅 사고력이라는 키워드가 미래세대가 갖추어야 할 중요한 역량으로 평가 받기 시작하면서 정보교과 또는 소프트웨어교육 영역에서도 주요한 목적으로 자리 잡고 있다. 그런데 컴퓨팅 사고력은 "컴퓨터"라는 용어가 사용되었지만 직접적으로 컴퓨터시스템의 사용과는 거리가 있는 용어로 알려지고 있다. 또한 자료수집 등의 컴퓨팅 사고력의 구체 내용은 일반적인 교과에서도 지향할 수 있는 내용이다[2]. 따라서 소프트웨어 교육에서는 컴퓨팅 사고력이라는 키워드와 소프트웨어 교육을 연계시키는 지속적 노력이 필요하다. 즉 소프트웨어 교육이

• First Author: Soo-Bum Shin, Corresponding Author: Soo-Bum Shin
*Soo-Bum Shin (ssb@gjue.ac.kr), Dept. of Computer Education, Gonju Natl University
• Received: 2015. 08. 24, Revised: 2015. 09. 16, Accepted: 2015. 11. 14.
• This work was supported by GJUE Univ of 2015 Research Grant.

컴퓨팅 사고력에 어떻게 영향을 미칠 수 있는지에 대한 연구가 좀 더 확대되어야 할 것이다.

이에 본 연구에서는 스크래치 언어 기반의 소프트웨어 교육 과정을 구성하고 교수학습 활동을 전개하고 교육효과를 분석하고자 한다. 그러기 위해 컴퓨팅 사고력 개념에 대해 좀 더 명확하게 분석하고 소프트웨어 교육의 다양한 방식에 대해 선행연구를 분석하고 실제 교육과정을 운영한 결과를 분석하고자 한다.

II. CT analysis for Software Education

1. 컴퓨팅 사고 개념

컴퓨팅 사고라는 용어는 Carnegie Mellon 대학의 Jeannette M. Wing 교수가 처음으로 사용하였다. 그 후 마이크로소프트의 지원으로 그 개념이 광범위하게 확산하게 되었다. 그는 이 개념이 단지 컴퓨터과학자들을 위한 것이 아니라 모든 사람을 위한 것이라고 제시하고 있다[3].

이와 같은 개념이 의미하는 것은 컴퓨팅 사고가 컴퓨터를 사용하는데 관련된 개념을 의미하기 보다는 컴퓨터시스템이 정보를 처리하는 방식과 같이 사고하는 능력을 일컫는 용어라고 할 수 있다. 실제 컴퓨팅 사고라는 용어는 수학교육, 일반교육학, 교육공학, 소프트웨어교육 등에서 광범위하게 사용되고 있다. 그 중에서 국내에서는 ACM의 하부조직의 컴퓨터 교사 학회인 CSTA와 ISTE 학회가 연계한 컴퓨팅 사고 개념을 많이 인용하고 있다. 이 단체의 핵심 내용은 자료수집, 분석, 표현, 분해와 추상화, 알고리즘, 자동화, 시뮬레이션, 병렬화 등으로 제시하고 있다[1].

또한 네브라스카대학의 컴퓨팅 사고에 대한 워크숍 자료를 요약하면 다음 표 [1]과 같다[4].

Table 1. CT Core Concept of Nebraska Univ.

CT types	Detail Explanation
Abstract thinking	Creating and using different levels of abstraction to understand problems by creating relevant models of the real world
Algorithmic thinking	Finding method with the most efficient and effective to solve a problem.
Logical thinking	uses logic to formulate and discard solutions.
Scalable thinking	Decompose a large problem into smaller problems that are easier to model and compose complex solutions from simpler algorithms and components.

추상, 확장, 병렬의 개념은 컴퓨터과학에서의 중요한 개념이지만 수집, 분석, 논리 등의 개념은 타 교과에서도 그대로 적용할 수 있는 개념이다.

즉 컴퓨팅 사고는 컴퓨터가 문제를 처리하는 방식으로 일상세계의 문제를 해결하고자 하는 방식이지 프로그래밍 과정 자체를 의미하거나 소프트웨어 교육에 한정적으로 사용되는 개념은 아니다.

2. 컴퓨팅 사고력을 위한 소프트웨어교육 내용

전통적으로 컴퓨터공학에서 알고리즘은 중요한 영역이며 프로그래밍 교육에서도 알고리즘적 사고는 중요한 역량으로 평가받고 있다. 그런데 알고리즘이라는 용어는 컴퓨터공학 뿐만 아니라 형식과학(Formal Science)에서 광범위하게 사용하고 있다[5].

알고리즘은 어떤 특정 문제를 위한 명령어들의 유한 집합이라고 정의를 내릴 수 있다. 또한 컴퓨터과학에서의 알고리즘은 입출력, 명확성, 유한성과 효과성을 갖추어야 한다[6].

알고리즘적 사고는 보다 구체적이다. 특히 컴퓨터과학과 관련된 알고리즘은 명확한 알고리즘 요소도 제시되고 있다. 반면에 컴퓨팅 사고력은 알고리즘적 사고보다 광범위하고 일반적인 진술이다.

그리고 ISTE와 CSTA에서 제안한 컴퓨팅 사고력 범위 안에 알고리즘 및 프로시저가 포함되어 있기 때문에 컴퓨팅 사고력이 알고리즘적 사고력을 포함하고 있다고 할 수 있다.

그리고 컴퓨팅 사고력이 확산되면서 전통적인 컴퓨터과학개론이나 프로그래밍의 강의에서도 알고리즘 역량 강화 이상의 내용을 구성하기 위해 기존의 교육내용의 변화를 꾀하고 있다[7].

이와 같은 관점에서 소프트웨어교육이 컴퓨팅 사고력 향상을 도모하기 위해서는 알고리즘적 사고 및 코딩 활동을 뛰어넘는 교육과정을 구성할 필요가 있다. 즉 자료를 분석하는 방법, 문제를 해석하고 분류 또는 작은 단위로 나누는 원리 등의 활동을 포함하는 내용까지 포괄하는 소프트웨어 교육과정이 요구된다.

III. A type of Software Education

알고리즘적 사고력을 뛰어넘는 컴퓨팅 사고력 향상을 위해서 교육과정을 구성할 경우 컴퓨터 또는 교육용 프로그래밍 언어가 필수적으로 사용될 수도 있지만 전혀 사용하지 않고도 이루어질 수 있다. 컴퓨팅 사고력이 프로그래밍을 뛰어 넘는 자료수집 및 분석과 자료의 분리 까지 포함하기 때문에 컴퓨팅과 프로그래밍을 생략한 채 교육내용을 구성할 수 있다. 네브라스카 대학에서의 컴퓨팅 사고 개념 워크숍에서는 컴퓨팅 사고를 가르치기 위한 유형으로 컴퓨터 사용과 학습자 액티비티로 구분하여 제시하고 있다[4]. 이에 본 연구에서 컴퓨팅 사고력 향상을 위한 교육과정을 구성하고 전후 테스트를 위해 소프트웨어 교육 유형을 분류하여 제시해 보고자 한다. 그리고 본 연구에서도 컴퓨터를 사용하지 않는 언플러그드 즉 학습자 활동 중심과 물리적인 컴퓨터를 사용하는 유형으로 나누고 컴퓨터를 사용하는 유형중에서도 다시 몇 가지로 세분화하여 제시하고자 한다.

1. 학습자 활동 중심 수업 유형

CSTA에서 제안한 컴퓨팅 사고력 세부 분야에서 자료의 분류와 문제의 세부적으로 나누고 분석하는 활동은 컴퓨팅을 통하지 않고서도 충분히 전개될 수 있는 영역이다. 또한 팀벨은 컴퓨터과학의 비전공자를 위하여 컴퓨터를 사용하지 않고 컴퓨터과학을 가르치기 위해 제안하였다. 그리고 뉴질랜드와 국제학생을 대상으로 교육결과에 대한 반응은 대체로 컴퓨터과학에 대한 흥미도가 상승한 것으로 나타났다[8].

그리고 Steven K.도 역할놀이는 컴퓨터과학에서 최소 15년 이상 교수법으로 채택되어 활용하고 있는 것으로 강조하였다[9]. 특히 그는 객체 지향 프로그래밍은 현대 프로그래밍 기법의 가장 중요한 개념으로 받아들여지며 추상적인 개념이 많이 내포되어 있다. 따라서 학생들이 이에 대한 일종의 역할놀이를 수행할 경우 객체지향 기법에 대한 인지능력을 향상시킬 수 있다고 제시하고 있다.

활동 중심의 의미는 학습자들이 실행순서, 자료의 흐름에 따라서 직접 움직이는 것이 핵심인 수업을 의미한다. 구체적으로는 역할놀이 활동이라고 할 수 있다.

학습자의 활동이 중심이 되는 소프트웨어교육은 다른 어떤 유형보다도 동기유발이 뛰어날 수 있다. 시스템을 조작하지 않고 학습자가 직접 시스템이나 자료가 되어서 움직이면서 실제 프로그램을 모방하는 것이기 때문에 학습자의 흥미를 이끌어낼 수 있다.

또한 교육용 언어 선택을 자유롭게 할 수 있다. 학습자의 활동이 교육용 언어의 실행에 영향을 미치지 못하기 때문이다.

2. 컴퓨팅 활용 수업 유형

컴퓨팅 사고력을 위해 컴퓨터를 활용하고자 할 때 우선적으로 프로그래밍 언어 선택이 필요하다. 로봇을 제어하거나 어떤 부가적인 장비를 사용하는 경우라도 이를 제어하기 위한 도구가 필요하며 그것은 프로그래밍 언어라고 할 수 있다. 물론 부가장비를 이용하지 않고 프로그래밍 언어만을 이용할 수도 있다.

2.1 교육용 언어의 선택과 활용

교육용 언어의 개념이 해외에서 광범위하게 사용되는 것은 아니다. 일반 상용 프로그래밍 언어인 베이직, 자바를 교육 도구로 선택하여 프로그래밍 교육을 수행하게 되면 교육용 언어로 일컫을 수 있다. 그렇지만 초중등 학생이나 컴퓨터공학 전공자 중 초보자나 컴퓨터공학 비전공자를 위해서 특화되어서 제작된 프로그래밍 언어가 있다. 이와 같이 초보 프로그래머를 위해 특화된 프로그래밍 언어 중에서 초등단계에서 사용할 수 있는 언어는 스크래치, 앨리스 등이 미국에서 가장 광범위하게 사용되고 있다. 그리고 비주얼베이직의 교육용 버전이라고 할 수 있는 스몰베이직도 초중등 학생들이 쉽게 사용할 수 있는 도구라고 판단된다. 그렇지만 스크래치는 다음 [그림 1]과 같이 8세 이상의 초보 프로그래머에게 적합한 것으로 제시되고 있다[10]. 또한 오라클 기업은 스크래치가 모바일 컴퓨팅 시대의 가장 중요한 도구로 각광받고 있는 자바를 배우는 가장 초보적인 도구로 추천하였다[11]

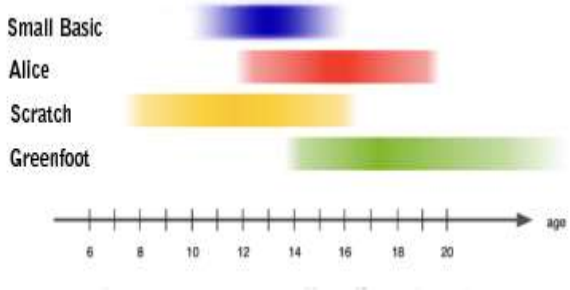

Fig. 1. Appropriate Educational Programming Language by Age

2.2 개발 키트 활용 수업

하드웨어 기술의 발달로 초소형 사이즈 컴퓨터가 발달하고 있다. 초소형 컴퓨터는 일반 사용자 보다는 중간 개발자를 위한 것으로 일반 PC, 센서 장치 등에 연결하여 사물을 조작하는 도구로 활용되고 있다. 가장 기본적인 보드는 스크래치 보드 이며 아두이노와 라즈베리파이는 마이크로프로세서를 가지고 있어서 더욱 복잡한 활동을 연계할 수 있다. 또한 최근에는 인텔사의 에디슨이라는 개발키트가 소개되고 있으며 아두이노와 연계할 수 있는 기능까지 포함하고 있다.

이러한 개발키트는 스크래치, 그린풋, C, 자바 등과 연계하여 외부 사물을 조작할 수 있다.

Mike Richard 외2인(2012)는 대학에서 초보 프로그래머가 쉽게 프로그래밍을 위해서 센서보드를 활용하여 긍정적인 효과를 제시하였다[12]. 그는 학습자들이 센서보드와 스크래치를 이용하여 20분 내에 프로그램을 작성해 냈고 계속적으로 프로그램 작성을 원한 것으로 제시하였다. 또한 Chulakorn Aritajati 외3인(2014)은 고등학생의 여름방학 캠프에서 스크래치와 아두이노를 이용하여 교육을 하였는데 컴퓨터에 대한 자기 효능감이 상승하였으며 프로그래밍이 즐거웠던 것으로 응답하였다[13]. 그들은 특히 연구 설계에서 구성원간의 충분한 이야기를 할 수 있도록 온오프라인 환경을 조성한 것도 교육효과에 긍정적인 영향을 주었다고 하였다.

소프트웨어교육과 개발 킷트의 연계는 추상적인 프로그래밍의 결과를 직접 느끼고 볼 수 있는 장점을 가지고 있다. 그러나 개발 킷트가 특정 언어만을 지원하기 때문에 프로그래밍 언어를 자유롭게 선택할 수는 없다. 그리고 피코보드는 센서 활용 수준, 아두이노와 라즈베리파이는 제어와 데이터 관리를 위해서 선택하여 사용할 수 있으며 지원하는 언어를 고려해서 선택해야 한다.

2.3 로봇 활용 수업

송정범 외(2009)은 피코로봇과 기존 전통적 방식의 소프트웨어 교육의 효과성 차이를 분석하였다[14]. 이를 통하여 로봇을 이용한 소프트웨어 교육의 우수성을 입증하였는데 국내의 정보통신기술 교육과정과 ACM의 교육과정을 근간으로 교육내용을 구성하였다. 프로그래밍에 집중을 하여 조건문과 반복문을 집중적으로 활용하는 교육과정 구성하였다.

이영준 외(2009)은 마인드스톰 NXT-G를 이용하여 알고리즘 교육 효과의 우수성을 분석하였다[15]. 이 연구에서는 로봇을 이용하여 알고리즘 교육 프로그램을 구성하고 실제 적용하여 그 효과성을 보여 주고 있다. 하지만 전체 컴퓨터과학에서 제시하고 있는 알고리즘 영역 중에서 다소 임의적으로 교육내용을 선정하였다.

Barry Fagin(2002)은 컴퓨터 과학 개념을 가르치기 위하여 레고 마인드스톰과 Ada 프로그래밍 언어를 사용하였다[16]. Ada는 소프트웨어 엔지니어링의 개념을 많이 포괄하고 있어 선택을 하였으며 학생들은 Ada 코딩을 통하여 로봇을 시뮬레이션 하면서 순차 흐름, 반복, 입출력, 배열, 파일 처리 등을 학습 하였다. 그리고 그 효율성을 입증하였다.

소프트웨어교육에서 로봇의 연계는 다른 유형보다 오버헤드가 높은 수준이라고 할 수 있다. 그것은 로봇조립과 제어에 많은 시간이 투입될 수밖에 없으며 제어를 위한 프로그래밍 언어도 가장 많은 제약 조건을 가지고 있다. 따라서 소프트웨어교육에서 로봇의 활용은 하나에 집중하기 보다는 로봇의 조립과 제어도 비중 있게 설정할 필요가 있을 때 필요하다고 할 수 있다.

IV. Study Design and Verification

1. 실험집단의 선정

본 연구는 초등학교 5-6학년 16명의 학생을 대상으로 이루어졌다. 총학생수가 65명인 소규모학교였다. 해당 학생들은 정보활용 교육, 소프트웨어 교육 등의 경험이 없는 것으로 사전 조사되었으며 구체적으로 수업전 사전 컴퓨팅 사고력 테스트를 통해 나타났다.

그리고 해당 교육은 3월부터 7월까지 15주차 교육이 이루어졌다. 특히 시골의 초등학교에서 일괄 방과 후 교육을 실시하는 프로그램 중의 하나로 이루어졌기 때문에 학생들의 희망에 의해서 구성된 학생들은 아니었다.

2. 교육과정 구성 및 운영

2.1 구성 전략

첫째, 스크래치 교육용 언어 기반 알고리즘 교육과정을 구성하였다.

프로그래밍 경험이 없는 초등학생들을 대상으로 교육을 하는 것이기 때문에 가장 쉬운 프로그래밍 언어로 평가 받고 있는 스크래치 교육용 언어를 선택하였다. 그리고 스크래치 언어 위주의 교육과정 구성을 탈피하여 생활 속의 소재와 연관된 알고리즘 구성 능력 위주의 교육과정을 구성하였다. 알고리즘 중심 교육과정 구현을 위하여 변수, 조건, 반복 구조의 개념을 집중적으로 구성하였으며 정렬 알고리즘에서는 초보학습자를 위해 정렬 체험 활동도 삽입하여 구성하였다.

이를 통하여 초보 프로그래머들은 스크래치 자체를 배우는 관점보다 스크래치를 통해 기본적인 알고리즘을 학습하는 것을 목적을 두고 교육과정을 구성하였다.

둘째, 피코보드 활용 교육과정을 구성하였다.

스크래치 센서보드를 활용하여 추상화된 공간의 작업이 실제 세계에서 구현되는 모습을 제공하고자 하였다. 전술한 바와 같이 송정범, 이영준, Barry Fagin은 알고리즘 교육을 위해서 피코로봇, 마이드스톰을 활용하였다. 또한 George공대에서는 컴퓨터 실습을 통한 자료구조 학습을 시도하여 성공적인 결과를 제시하였다[17].

그리고 본 연구에서는 초등학생의 인지적 수준과 정보 및 로봇교육 경험 등을 고려하여 스크래치 센서보드를 선택하여 활용하였다.

센서보드의 활용은 구체적 조작 경험을 필요로 하는 초등학생의 프로그래밍 활동에서 주요한 도구로 판단하여 삽입하여 구성하였다. 그리고 주요 선행 연구에서도 도구를 이용하는 프로그래밍 활동은 효율적이었다라고 기술하고 있어서 삽입하였다.

셋째, 순서도 및 자연어 결합 표현 기법을 활용한다.

본 연구에서 스크래치 프로그래밍 언어를 알고리즘을 표현하는 주요 도구로 채택하였지만 그 이외에 자연어와 순서도를 결합한 형태의 표기 방법, 스크래치 보드와 연결하는 방법 등을 통해 학습자의 로직을 표현해내고자 노력했다. 자연어 표기는 해석의 다양성 등으로 여러 가지 논란이 있을 수 있지만 프로그래밍을 쉽게 안내하기 위해서 필요하다고 판단하였다.

그리고 Jean-Paul TremBlay(1989)는 자연어를 이용해서 알고리즘을 표현할 수 있다고 제시하고 있다[18]. 이에 실제 코딩에 활용되지 못하더라도 논리적인 구성 수준을 파악하기 위하여 순서도에 자연어를 이용한 표현할 수 있도록 구성하였다.

이와 같은 전략으로 구성된 교육과정을 [표 2]와 같이 제시하였다.

Table 2. Practiced Detail Curriculum during 15 Weeks

Week	Theme	Core Contents
1week	Introduction to Programming	- Measuring on understanding ability of decision and looping structure in everyday life - Measuring on transformation ability of picture symbols into pseudo codes
2week	Programming and Problem Solving Strategies	- Value of computer programming - Methods and types expressing Algorithms
3week	Scratch and Coordinate Plane(1)	- Basic concepts for x-y coordinate plane - Sprite movement in the coordinate plane
4week	Variable of Scratch	- Sum from 1 to 150 using variables - Basic concepts of Variables - Sum of odd, even using simple condition statements
5~6 weeks	Make of Kawibawibo Game	- use of random function - Basic concepts of variable - Printout Kawibawibo corresponding to 1, 2, 3 of random numbers - Determine winner of Kawibawibo automatically - Give a score to game winner using variable
7week	Scratch and Coordinate Plane(2)	- Sprite movement using a change of x-y coordinate - Implementation spring out of sprite using x-y coordinate plane in the stage
8~11 weeks	Sensor of movement principle and Sensor board of the scratch	- Several goods making use of sensor - example of being used button sensor and slide sensor - Make use of a slide sensor in the scratch - Make use of a button movement in the scratch - Example of light sensor being used in everyday life - Make use of light sensor in the Scratch - Make use of sound sensor in the Scratch
12~14 weeks	Sorting Algorithm	- Introduction concepts of algorithm - Introduction of basic sorting algorithm - Experience selection and bubble sort using students themselves movement and check out how accomplishment - Understanding of List data structure of Scrach - Implementation bubble sort using Scratch - Implementation of selection sort using the Scratch - Comparing performance ability of several algorithms
15week	Finish	- Watch 'life coding' video

2.2 교육과정 운영

15주 동안의 교육과정은 초보자를 위해서 매우 쉽게 프로그래밍을 수행할 수 있도록 초기단계에서 문제를 단순화하여 구성하는 일상생활 소재 등을 소개하였다. 그 후로 7주차까지는 기본 로직 구현을 위한 교육과정을 구성하였으며 8주부터는 센서보드와 연계하는 내용으로 구성하였다. 그리고 자료구조 기초를 교육과정으로 구성하여 운영하였다.

그 중에서 핵심적인 내용이라고 할 수 있는 변수와 정렬 차시에 대한 실제 교수학습 활동을 요약하여 다음 [표 3]과 같이 제시하고자 한다.

실제 교육과정 운영은 방과 후 수업의 형태로 주당 2시간 동안 이루어졌으며 학교 행사가 있을 때는 보강이 이루어졌다. 그리고 해당 학생들의 소프트웨어교육 경험은 없었으며 학업성취도도 낮은 수준의 학생들로 구성되었다. 이를 위해 강사와 별도로 보조교사를 운영하여 집중도가 떨어지거나 부진한 학생을 개별지도를 실시하였다.

16명 안팎의 소수그룹으로 학습자가 구성되었지만 열악한 배경지식과 학업성취도가 높지 않은 상태였기 때문에 집중도를 유지하면서 한 학기동안 교육과정을 운영하기 위해서 개별지도, 상담 등의 다양한 노력이 필요하였다.

Table 5. Theme of Key Lesson & Activities Contents of the Teaching and Learning

차시, 주제	교수학습 활동 핵심 내용
4주 스크래치에서 변수	-변수의 기본 개념 배우기 프로그래밍에서 변수의 개념, 즉 단순한 숫자가 아니고 저장장소로서의 변수의 개념에 대해서 학습 -스크래치에서 변수 만들기 스크래치에서 변수를 만드는 방법에 대해서 알아보기. 변수를 만드는 버튼의 위치와 변수에 숫자나 문자 등을 저장하는 방법 알아보기 - 변수를 이용하여 1~150까지의 합 구하기 -조건문과 변수를 이용한 계산하기 간단한 조건문을 이용한 홀수, 짝수의 합 구하기
12주, 13주 정렬 알고리즘 (1)	- 알고리즘의 개념 소개 어떤 문제를 해결하기 위해 명확히 정의된 유한 개의 규칙과 절차의 모임이라는 것에 대하여 설명과 일상생활 속에서의 알고리즘이 적용된 예를 찾는 활동 전개 - 기본적인 정렬 알고리즘의 소개 어떤 순서에 의한 나열이라는 정렬의 개념을 이해하고 정렬 알고리즘에 대해서 학습하고 학교생활 속에서 정렬 데이터를 활용하는 사례 찾기 활동 전개 선택정렬, 버블 정렬을 학생들이 움직여서 체험해보고, 어떠한 절차를 거쳐서 이루어지는지 그룹 토의를 통해 확인 스크래치 블록을 통해 단계별로 선택 정렬을 구현하고 그룹토의를 통해 자신의 블록의 문제점을 파악한 후에 스크래치 블록을 수정 보완하는 활동 전개

2.3 컴퓨팅 사고력 전후 검사지 구성

프로그래밍의 교육 효과를 분석하기 위해 김종혜(2009)의 컴퓨팅 사고력 테스트 평가 문항을 활용하였다[19]. 해당 평가 문항은 컴퓨터를 조작하지 않고 지필평가만을 이용하여 측정할 수 있게 구성되어 있다. 그것은 실제 소프트웨어교육이 CT에 얼마나 영향을 주었는지 확인할 수 있는 방법이라고 할 수 있다. 실제 수업에서는 프로그래밍 언어를 이용하지 않는 경우도 있지만 많은 시간을 스크래치와 보드를 활용하였다. 이와 같은 수업 전후에 컴퓨터 조작과 프로그래밍 실습을 제외한 채 평가가 이루어졌다. 이에 소프트웨어교육이 실제 CT에 얼마만큼의 영향을 미쳤는지 파악할 수 있는 것으로 해석할 수 있다.

김종혜(2009)의 연구는 블록 프로그래밍을 가정하지 않고 평가 문항을 구성하여 본 연구에서는 문법 에러와 관련된 평가 문항을 제외하여 [표 4]와 같이 구성하였다[19].

Table 4. Evaluation Area and Criteria of CT

Evaluation Area	Evaluation Criteria
Understanding & Analysis	Elaboration, Sensibility, Reorganization
Searching for Problem Solving Strategies	Fluency, Flexibility, Originality
Design for Problem Solving Strategies	Elaboration, Reorganization
Implementation	Elaboration

그리고 실제 문항은 다음 [표 5]와 같이 구성되었다.

Table 5. Themes of CT Evaluation Question

Item	Theme	Evaluation Area	Allot
1	Library	Understanding & Analysis	5
2	Design by Number	- Understanding & Analysis - Searching for Problem Solving Strategies	5 3
3	Design by Number	- Understanding & Analysis - Searching for Problem Solving Strategies - Design for Problem Solving Strategies	5 3 3
4	Energy	- Understanding & Analysis	5
5	Phone Call	- Searching for Problem Solving Strategies	3

이와 같이 구성된 평가 문항지를 이용하여 3월 학기 초에 측정을 하고 수업을 15주간 진행을 한 후에 같은 수준을 유지한 채 다른 평가 문항지를 이용하여 다시 측정을 실시하여 [표 6]과 같은 결과를 얻어 냈다.

2.4 연구 결과

소프트웨어 교육 초기에는 총 16명의 초등학생이 참여를 했지만 수업결손과 평가 미참여 등으로 최종 전후비교에 참여한 인원은 12명이다.

Table 6. CT Evaluation Result

Area	Significant Probability	Average Gap of Order
Understanding and Analysis	.030	3.67
Research	.020	1.78
Design	.808	0.11

대응표본 t검정(n=12)

이해, 분석과 탐색의 영역은 문제를 분석하고 문제에서 요구하는 것이 무엇인지를 파악하는 역량을 테스트하고 구성 요소들 간의 연관성을 찾아내는 능력을 측정하는 분야이다. 해당 영역에서 초보 프로그래머의 전후비교는 유의미한 결과를 도출하였다.

그럼에도 전후비교 평균차이는 6점 이상의 차이를 보이지 못하였다. 이것은 해당 소프트웨어교육에 참여한 모든 학생은 의무참석이었으며 집중도가 낮은 상황이었기 때문에 나타난 현상으로 파악할 수 있다.

동일한 학생의 사전/사후 검사 문항으로, 사전 검사지에 응답한 답안이 보다 정교해지고, 정확한 표현을 사용한 것을 파악할 수 있다.

하지만 설계 영역은 위의 표와 같이 유의미한 평균차이를 나타내지 못하였다. 즉 설계는 로직을 유의미하며 효율적으로 구성하는 영역으로서 이해와 분석 그리고 탐색보다 근본적인 역량이 요구된다.

본 연구의 대상이었던 초등학교 학생들은 사전 소프트웨어 교육 경험이 없어서 프로젝트 기반 수업 등의 학습자가 직접 로직을 설계하고 결과물을 도출해 내는 활동이 미흡한 상태였다.

3계 영역 중에서 변화도가 가장 낮게 나타난 것은 이와 같은 원인으로 발생한 것으로 판단된다. 즉 3개 영역 중에서 학습자의 역량이 가장 많이 요구되고 난이도가 있는 영역이 설계 영역이었는데 평균차이가 낮게 나타난 점은 학생들이 가장 어려워했던 영역으로 해석할 수 있다.

V. Conclusion

본 연구는 프로그래밍 교육의 결과가 컴퓨팅 사고력 향상에 미치는 효과를 분석하기 위해 교육내용 구성, 교수학습 활동 수행, 전후 평가와 분석의 내용을 수행한 것이다.

프로그래밍 교육에 대한 다양한 연구가 나타나고 있음에도 불구하고 실제 교육 후에 컴퓨팅 사고력의 진전 수준을 측정하는 연구는 미흡하다. 특히 소프트웨어교육이라는 타이틀로 초중등학교 정규교육과정에 삽입되는 등의 교육정책이 추진되고 있는 상황에서 이와 같은 수준의 연구는 필요하다고 판단된다.

이에 본 연구에서는 전세계적으로 초보 프로그래머 교육 도구로서 가장 대중적인 스크래치를 활용하여 교육내용을 구성하고 교수학습 활동을 실시하였다. 또한 본 연구에서는 자연어 표기, 순서도 활용, 스크래치 보드 도구 이용을 통하여 알고리즘

교육 형태를 다양하게 실시하였다. 실제 소프트웨어 교육은 15주 동안 이루어졌으며 전후 컴퓨팅 사고력을 측정하였다. 측정 결과 문제의 이해와 분석, 탐색하는 영역에서의 학생 변화는 유의미하였으며 설계 영역에서의 변화는 제대로 나타나지 못하였다.

그런데 본 연구에서 실제 컴퓨팅 사고력의 세부 영역별로 학습자의 역량을 측정하고 변화도를 분석하지 못하였으며 향후 연구에서는 세부 영역별로 변화도를 측정하는 노력이 요구된다.

그리고 향후 연구에서는 교사의 어떤 접근이 컴퓨팅 사고력 향상에 보다 도움이 되는지에 대한 연구, 어떤 평가 문항이 소프트웨어교육의 결과를 보다 효율적으로 나타나게 할 수 있는지에 대한 추가적인 연구가 필요한 상황이다.

REFERENCE

[1] MOE, A Promotion Plan of Cultivation Peoples of Talent for Oriented Society of Software. 2015http://www.moe.go.kr.

[2] Linda Mannila, Valentina Dagiene, Barbara Demo, Natasa Grgurina, Claudio Mirolo, Lennart Rolandsson, Amber Settle, Computational Thinking in K-9 Education. ITiCSE-WGR'14, pp. 1-3, June. 2014.

[3] Jeannette Wing (2006). Computational Thinking. Communications of the ACM . Vol. 49 No 3, pp. 33-35, March. 2006

[4] Google CS4HS Workshop(2013). Compuational Thinking. Jul. http://www.cs.unomaha.edu/~hsiy/CS4HS

[5] Wikipedia(2015). http://www.wikipedia.org

[6] Glenn Brookshear, Computer Science an overview Addison Wesley, 5th Edition. pp. 2-5, 1998.

[7] Heather Bort, Dennis Brylow, CS4Impact: Measuring Computational Thinking Concepts Present in CS4HS Participant Lesson Plans. SIGCSE'13, pp. 427-428, Mar. 2013.

[8] Tim Bell, A low-cost high-impact Computer Science show for family audiences. Australasian Computer Science Conference. Canberra, pp. 10-16, Jan. 2000.

[9] Steven K. Andrianoff, David B. Levine, Role Playing in an Object-Oriented World. SIGCSE'02. Kentucky, USA. p. 121, Feb-Mar. 2002.

[10] Soobum Shin, Selection Strategies of Educational Programming Language. Educational Material 2014-00. KERIS. p. 42, 2014.

[11] Young Developer - Visual Programming Software Tools http://www.oracle.com/technetwork

[12] Mike Richard, Marian Petre, Arosha K. Bandara, Starting with Ubicomp: Using the SenseBoard to Introduce Computing. SIGCSE'12, p. 587, Feb-Mar. 2012.

[13] Chulakorn Aritajati, Mary Beth Rosson, Joslenne Pena, Dana Cinque, Ana Segura. A Socio-Cognitive Analysis of Summer Camp Outcomes and Experiences. SIGCSE'14, Mar. p. 583. 2015

[14] Jeong-Bum Song, Tae-Wuk Lee, The Effect of Programming Education using Pico Cricket on Improving Problem Solving Ability. Journal of Korean Practical Arts Education. Vol.14 No.4, pp. 243-258, 2008.

[15] Lee Yeoung Jun, Lee Eun Kyung, An Algorithm Learning Program with Robot. The Journal of Korean association of computer education . Vol.12 No.1. pp. 42-43, 2008.

[16] Barry Fagin, Laurence Merkle, Quantitative Analysis of the Effects of Robots on Introductory Computer Science Education. JERIC. Vol 2. No 4, p. 2, 2002.

[17] Lauren Rich, Heather Perry, Mark Guzdial. A CS1 Course Designed to Address Interests of Women. SIGCSE'04, Virginia. pp. 190-191, Mar. 2004.

[18] Jean-Paul Tremblay, Richard B. Bunt, Introduction to Computer Science. McGraw-Hill. p 21, 1989.

[19] Kim, JongHye, Secondary Education Program for Problem-solving Ability based on Computational Thinking. A Doctor's Thesis. Korea University Graduate School. pp 221-259, 2009.

Authors

Soo-Bum Shin received the Bachelor's degree in the department of Elementary Education from the Incheon Natl Univ. of Education in 1991. He received the MS degree and the Ph.D. degree in the Department of Computer Education from the KNUE.

Dr. Shin joined the faculty of the Department of Computer Science at Gongju Natl University in 2005. He is currently a Professor in the Department of Computer Education, Gongju Natl University. He is interested in SW education.