Computational Thinking 역량에 대한 학습자 평가를 위한 스크래치 코드 분석

김수환†

요 약

본 논문의 목적은 Computational Thinking(CT) 교육에서의 학습자 평가를 위한 방안으로 학습자가 제작한 프로젝트의 코드를 분석하는 방법을 제안한다. 최근 초중등 SW교육에서는 블록형 프로그래밍 도구를 활용한 교육이 이루지고 있으므로, 학생들의 산출물인 스크래치 프로젝트의 코드를 분석하여 CT 개념 습득과 학습자의 수준을 평가하는 방안을 모색하였다. 초보학습자 45명의 프로젝트를 분석한 결과 초보학습자들의 잘못된 습관에 대한 패턴이 나타났으며 CT 개념 학습을 코드의 패턴에 의해 평가할 수 있다는 것을 검증하였다. 학습자의 수준이 높을수록 논리적 사고, 동기화, 플로우 제어, 데이터 표현의 요소 점수가 높게 나타났다. 본 연구의 결과는 초중등 SW 교육에서 CT 개념 학습에 대한 평가를 위해 활용할 수 있다.

주제어 : SW 교육, 컴퓨팅 사고력, 평가, 코드 분석

Analysis of Scratch code for Student Assessment about Computational Thinking Capability

Soohwan Kim†

ABSTRACT

The purpose of this research is to suggest the method of code analysis for evaluating learners' projects in computational thinking(CT) education. Recently, block programming tools are applied to K-12 SW education, this study considered the assessment method for evaluating students' levels and learning about CT concepts through analyzing codes of the Scratch projects that students created. As a result from the analysis of 45 projects of novices, it showed the bad coding patterns of novices and verified that it is possible to evaluate students' learning about CT concepts through the analysis of their codes. The higher learner's level, the greater scores of logical thinking, synchronization, flow control, and data representation. This result is able to apply to student assessment of CT concepts in K-12 SW education.

Keywords : SW Education, Computational Thinking, Assessment, Code Analysis

†종신회원 : 총신대학교 조교수
논문접수: 2015년 7월 4일, 심사완료: 2015년 9월 7일, 게재확정: 2015년 9월 9일

1. 서론

미래 산업에서 SW의 중요성이 강조되면서 세계 여러나라에서는 미래 SW인재 양성을 위해 컴퓨팅 교과를 신설하여 초중등 교육에 적용하고 있다[1][2][3]. 최근 우리나라에서도 SW인재 양성을 위해 SW 교육과정을 신설하고 2018년부터 초중등 교육현장에 적용할 예정이다[4][5].

초중등 교육현장에 SW교육과정이 성공적으로 정착하기 위해서는 교육목표부터 평가에 이르기까지 일괄된 형태를 갖추어야 하며, 목표에 부합하는 교육방법 및 교육평가 방법이 개발되어야 한다. 2015년에 교육부에서 발표한 SW교육 운영지침을 살펴보면 SW교육의 목표를 컴퓨팅 사고력(Computational Thinking)의 증진으로 설정하였다[4][5].

컴퓨팅 사고력은 컴퓨팅을 통해 문제를 해결하는 과정에서 사용되는데, 컴퓨팅 사고력을 평가하기 위해서는 먼저 컴퓨팅 사고력의 구성요소를 정립해야 한다. Wing(2008)[6]은 컴퓨팅 사고력의 구성요소를 추상화와 자동화라고 제시하였고, CSTA & ISTE[7]는 컴퓨팅 사고력의 문제해결 요소로 자료수집, 자료분석, 자료표현, 문제분해, 추상화, 알고리즘과 절차, 자동화, 시뮬레이션, 병렬화를 제시하였다. Code.org에서는 문제해결요소를 추상화, 알고리즘, 문제분해, 패턴인식의 4가지로 구분하고 있다[8]. 영국의 CAS에서는 컴퓨팅 사고력의 요소를 문제분해, 패턴 인식, 추상화, 패턴 일반화, 알고리즘 디자인의 5가지로 제시한다[2].

이상에서 나타난 컴퓨팅 사고력의 하위요소는 문제해결 과정에서 사용되는 기능과 사고력과 관련된 것이다. SW 교육지침에 제시된 교육내용을 살펴보면 생활과 소프트웨어, 알고리즘과 프로그래밍, 컴퓨팅과 문제해결의 영역으로 구분되어 있다[5]. 세가지 영역중에서 컴퓨팅 사고력을 중점적으로 길러줄 수 있는 영역은 알고리즘과 프로그래밍 영역이며, 이 영역에서 다양한 문제해결과정을 알고리즘과 프로그래밍으로 경험하게 되어있다. K-12 교육에서 알고리즘과 프로그래밍은 교육용프로그래밍 언어를 사용하여 학습하는데, MIT 미디어랩에서 만든 스크래치가 많이 사용되고 있다. 스크래치팀에서는 창의컴퓨팅 가이드북을 통해 컴퓨팅 사고력을 개념, 실행, 관점으로 구분하고 각 요소를 평가하기 위한 방안을 제시하였다[9][10].

컴퓨팅 사고력의 기능이나 관점의 경우는 실기평가나 포트폴리오법, 설문 등을 통해 평가할 수 있고, 개념의 경우는 학생들이 제작한 프로젝트의 코드를 분석하는 방법으로 평가할 수 있다[10][11][12]. 따라서 본 연구에서는 컴퓨팅 사고력의 개념을 평가하기 위한 방법으로 스크래치 코드를 분석하는 방법을 탐색, 분석하고 시사점을 도출하였다.

2. 관련연구

2.1 SW 교육과 평가

2015년 교육부에서 발표한 SW 교육 운영지침에 의하면 SW교육의 목표를 ‘컴퓨팅 사고력을 가진 창의.융합 인재’로 설정하고 있다. 컴퓨팅 사고력이란 컴퓨팅의 기본적인 개념과 원리를 기반으로 문제를 효율적으로 해결한 수 있는 사고능력이다[4][5].

이러한 구성요소를 토대로 SW 교육에서의 평가 내용 중에서 알고리즘과 프로그래밍의 영역에서는 다음과 같이 제시하고 있다. ‘초등학교에서는 체험중심의 활동으로 문제해결과정과 알고리즘, 프로그래밍을 체험해 보고 컴퓨팅 사고를 이해하고 있는지를 평가한다. 중학교에서는 다양한 문제를 이해.분석하고 구조화하여 알고리즘으로 설계할 수 있는지와 프로그래밍으로 구현하고 분석할 수 있는지를 평가한다’로 되어 있다[5]. 일반적인 교육에서의 평가도구는 <표 1>과 같이 여러 가지 방법이 있으며, 이런 도구를 통해 지식, 기능, 태도의 항목을 평가한다[13].

초중등교육에서의 목표는 컴퓨팅 사고를 이해하고 있는지에 대한 평가와 프로그래밍 구현 및 분석 능력에 대한 평가로 제시된다. SW교육의 경우 실제 프로그래밍이 이루어지나 높은 수준의 알고리즘 구현보다는 자신의 아이디어를 프로그래밍으로 표현하는 것에 초점이 맞추어져 있다. 따라서 학생들의 산출물 자체가 평가의 대상이

되어야 한다. 즉, 수행평가의 방법 중에서 학습자들이 제작한 프로그램을 직접 평가에 활용하는 방법이 필요하다[14].

<표 1> 평가도구의 분류(CES, 2010)

시험 (tests)	산출물/프로젝트 (products/ projects)	수행 (performances)	과정기술 (process skills)
• 에세이 • 선다형 • 연결형 • 단답형 • 정오형	• 광고 • 인공물 • 녹음한 자서전 • 배너 • 청사진 • 책 후기 • 브로셔 • 책 • 게시판 • 만화 • 사례연구 • 그림 • 전시 등	• 활동 • 안내 • 찬가 • 춤 • 회의 • 토론 • 설명 • 패션쇼 • 체험학습 • 소개 • 직업면담 • 뉴스리포트 등	• 일화기록 • 관찰체크리스트 • 개념도 • 간략한 면담 • 상호작용 분석 • 구두질문 • 관찰 • 질문 생산도 등

2.2 프로젝트 코드 분석

스크래치의 경우 8개의 카테고리로 명령어 블록이 분류되어 있는데, 각 블록들의 사용횟수를 통해 코드 분석이 가능하다. Brennan & Resnick[10]의 연구에 의하면 프로젝트 분석법은 컴퓨팅 사고력의 개념을 평가하는데 활용할 수 있으며, 프로그램의 소스 코드를 분석하여 평가하는 방법은 SW 공학에서도 사용하고 있는 방법이다[15].

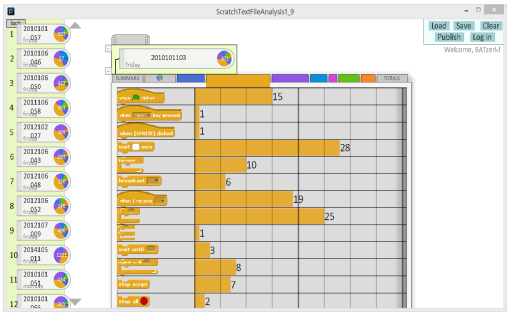
<표 2> CT 교육 평가방법(Brennan & Resnick, 2012)

평가방법	개념	실습	관점
프로젝트 분석 (코드분석)	개념이 포함된 블록의 사용을 평가	N/A	N/A (코멘트 같은 다른 웹사이트의 데이터 분석시 가능)
작품 인터뷰	제한된 프로젝트에서의 개념에 대한 이해 평가	기억의 제한이 있지만 학생들의 디자인 경험에 기반해서 가능함	일부가능, 직접 물어보기 어려움
디자인 시나리오	제시된 프로젝트에 대해 개념적인 이해의 수준 평가	제시된 프로젝트의 새로운 상황에서 실시간 반응을 평가	일부가능, 직접 물어보기 어려움

<표 2>에서 나타난 것처럼 프로젝트 소스 분석 방법은 평가방법이 될 수 있지만 실제 학습자의 기능이나 태도를 평가하기는 어렵다. 그러나 학습자의 개념에 대한 이해를 평가하는 방법으로 사용이 가능하다[11][12][16][17][18]. 따라서 본 연구에서는 학생들의 스크래치 프로젝트 코드 분석을 통해 컴퓨팅 사고력의 개념을 이해하고 사용할 수 있는지에 대한 부분에 대한 평가 방법을 제안한다.

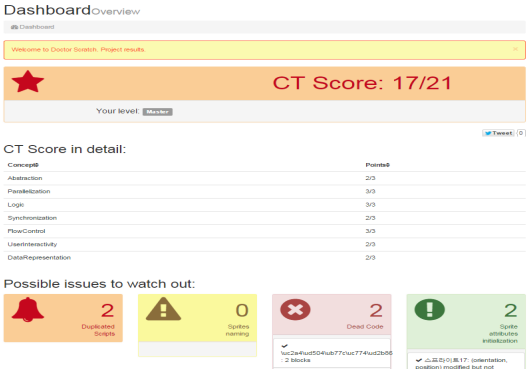
교육용프로그래밍 언어의 코드를 분석할 수 있는 도구에는 Scrape, Drscratch 등이 있다[19][20].

<그림 1>의 Scrape는 스크래치 코드 분석툴로 각 블록의 횟수와 저장횟수, 스프라이트 수, 블록수, 스택수 등의 분석이 가능하다[19].



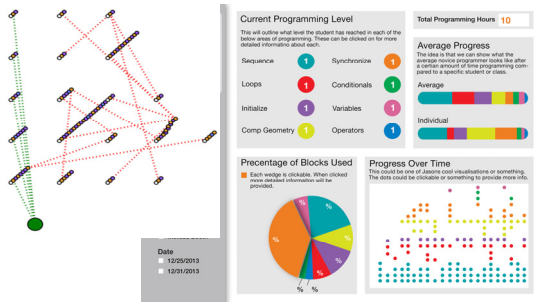
<그림 1> Scrape 분석 화면

Drscratch도 <그림 2>와 같이 스크래치 코드 분석툴이며 Scrape의 기능을 발전시켜 각 블록의 횟수를 분류하여 데이터 표현, 논리적 사고, 사용자 상호작용, 플로우 제어, 추상화, 병렬화, 동기화의 요소를 사용하는데, 그 정의는 Brennan & Resnick[10]의 연구와 유사하다. 또한 각 점수를 합산한 등급을 보여주고 코드 복사, 스프라이트 이름변경, 죽은 코드, 스프라이트 속성을 보여준다[20].



<그림 2> Drscratch 분석 화면

Martin & Fields[21]이 개발한 도구도 스크래치를 분석하는 툴로 Scrape 와 유사한 기능을 가지고 있으나 <그림 3>과 같이 스프라이트들 간의 방송 주고받기에 대한 흐름을 한눈에 볼 수 있는 기능을 제시하고 있다.



<그림 3> Fun 툴 분석 화면

2.3 코드 분석의 알고리즘

스크래치는 이벤트 기반 언어이고 스프라이트 중심으로 코드가 이루어져 있으므로 스크래치 프로젝트를 분석해 보면 <그림 4>와 같이 스프라이트 하위에 각 명령어를 가지고 있는 구조로 되어 있다.

```
<sprite name="스프라이트4">
<stack xpos="15" ypos="46">
<block category="Events" name="whenIReceive"
type="Hat">
<param type="label">game1</param>
</block>
<block category="Control" name="wait:elapsed:from:"
type="Stack">
<param type="number">15</param>
</block>
<block category="Looks" name="show"
type="Stack"/>
<block category="Data" name="showVariable:"
type="Stack">
<param type="label">점수</param>
</block>
<block category="Control" name="doUntil"
type="Stack">
<param type="label">
```

<그림 4> 스크래치 소스 코드 구조

Scrape는 스크래치 1.4버전의 코드만 분석이 가능하며, 각 블록의 빈도수와 블록수, 스택수만을 측정할 수 있다. 즉, 단순하게 각 블록을 사용했느냐에 따라 학습자가 그 개념을 이해하고 사용했다는 가설 하에 다음과 같은 목적을 가진다[12][18].

- 얼마나 많은 프로그램들에서 반복을 사용되는가?
- 각 프로그램에 반복이 얼마나 사용되는가?
- 어떤 수준의 중첩이 프로그램에서 사용되는가?

Hairball에서는 스크래치 소스 코드 분석을 통해 다음과 같은 평가요소를 제안했다[12].

- 어떤 프로그램이 방송하기와 방송받기의 오류를 포함하고 있는가?
- 어떤 프로그램이 방송하기/방송받기 이벤트의 결과로 무한 루프가 발생하는가?
- 어떤 프로그램이 시작상태를 초기화하지 못하는가?
- 어떤 프로그램이 복잡한 애니메이션(시간, 모양 변화, 동작, 반복 등이 포함된)을 제대로 구현하지 못하는가?

Drscratch는 이전 프로그램인 Hairball에서 발견한 요소인 초기상황, 말하기/사운드 동기화, 방송 주고받기, 복잡한 애니메이션의 요소를 발전시켜 <표 3>과 같은 컴퓨팅 사고력의 개념을 재구성하였다.

<표 3> CT 개념에 대한 개발 수준

CT 개념	베이직	개발자	마스터
데이터 표현	스프라이트 속성 변경	변수를 이용한 연산	리스트를 이용한 연산
논리적 사고	if	if else	논리 연산
유저 상호작용	시작버튼	키입력, 클릭, 묻고 기다리기, 마우스 블록	조건 실행, 비디오, 오디오
플로우 제어	블록들의 순차	반복, 무한반복	까지 반복
추상화	하나 이상의 스프라이트와 스크립트	사용자블록	복제이용
병렬	시작버튼의 2개 사용	키입력의 2개사용, 같은 스프라이트의 클릭 2개 사용	방송 받을 때 2개 사용, 클론 생성, 조건실행 2개사용, 배경바꾸기 2개 사용
동기화	기다리기 블록	방송 보내기, 받기, 모두 멈추기, 모든 스프라이트 멈추기	까지 기다리기, 배경 바뀔 때 방송하기 기다리기

또한 Moreno & Robles [17]은 스크래치 초보자들의 나쁜 습관으로 스프라이트의 이름을 변경하지 않는 것과 함수를 만들지 않고 코드를 반복

해서 사용하는 것이라고 제시하였다. Fields 외 [16]는 스크래치를 통한 미디어와 코딩교육을 실시하고 학습자 코드를 분석하였는데, 불린, 변수, 랜덤 수의 사용이 프로그래밍의 복잡도를 측정하는 요소가 될 수 있으며, 방송하기와 변수 블록이 동기화를 위한 중요블록으로 사용되고 있음을 밝혔다.

따라서 본 연구에서는 Scrape와 Drscratch 두 가지 도구를 사용하여 학생들의 스크래치 프로젝트 코드를 분석하고 컴퓨팅 사고력의 개념을 평가하는 방법을 제안하였다.

3. 연구의 방법 및 절차

3.1 교육내용 및 측정요소

본 연구의 목적은 CT 교육 후 학습자들이 만든 프로젝트 파일을 분석하여 평가하기 위한 방안을 모색하는 것이므로, 먼저 <표 4>와 같이 2014년도 2학기에 수도권 소재 C대학교에서 스크래치를 통한 CT 교육을 실시하고 기말과제로 제출한 스크래치 파일의 소스코드를 분석하였다.

<표 4> 교수・학습 활동 주제

주	활동 주제	CT 요소
1주	스크래치 소개 애니메이션	인터페이스 설명 절차, 반복/ 표현하기
2주	조건에 따른 애니메이션 상호작용 애니메이션	절차, 반복, 분기/ 재사용 & 재구성/ 표현하기
3주	미로게임 바나나 게임	변수, 난수, 객체 통신/ 테스팅 & 디버깅/ 표현하기, 연결하기
4주	음악 프로그램 리스트 활용	데이터 입출력, 불린 로직, 연산자/ 추상화 & 모듈화/ 연결하기, 질문하기
5주	수치계산 프로그램 퀴즈 프로그램	불린 로직, 연산자, 객체 통신/ 추상화 & 모듈화/ 연결하기, 질문하기
6~7주	내 프로젝트 구상하고 구현하기	모든 요소

연구에 참여한 학생들은 모두 컴퓨터 비전공자로 1-4학년 학생들 75명이었으며, 데이터가 제대로 수집된 45명의 프로젝트를 대상으로 코드 분석을 실시하였다. 코드 분석이 실제 학습자의 능력을 측정할 수 있는지 검증하기 위해 self-프로그래밍 능력, self-CT 역량, 실제 실기과제 평가점수와의 관계를 통해 상관분석을 실시하였다. self-프로그래밍 능력은 스스로 프로그래밍을 잘 한다고 인식하는 문항으로 수집하였으며, self-CT는 Brennan(2015)이 제시한 CT 관점을 문항으로 제작하여 설문하였다[18][22]. self-CT 문항은 컴퓨터를 이용하여 새로운 창작물을 만들 수 있다고 생각한다와 같은 문항으로 이루어져 있다.

3.2 분석절차

<표 5>와 같이 분석절차는 3단계로 나누어지며 1차 분석에서는 Scrape 툴을 이용하여 학습자들의 코드에 대한 빈도 분석을 주로 한다.

<표 5> 소스 코드 분석 절차 및 내용

분석절차	분석내용	분석툴
1차분석	초보학습자들의 일반적 코드 특성 - 빈도분석	Scrape
2차분석	논리적사고력과의 관계 - 상관분석, t-test	Drscratch SPSS
3차분석	CT 개념과의 연관성 CT 판별툴의 타당성 - 상관분석, t-test	Drscratch SPSS

2차 분석에서는 논리적사고력과의 관계를 알아보기 위해 상관분석과 t-test를 실시하였다. 논리적사고력과의 관계를 알아보기 위해 <표 6>과 같이 윤일규[23]의 연구에서 제시한 논리적사고력 요소와 관련 명령어와의 관계를 적용하여 각 명령어의 사용빈도를 논리의 점수로 측정하였다.

<표 6> 논리적 사고력 요소와 관련 명령어

논리적사고력요소	관련 명령어
비례논리	동작, 형태, 소리
변인통제	제어, 관찰, 변수
명제논리	조건, 반복

윤일규[23]에 의하면 정보교육의 문제해결과정에서 요구되는 논리적 사고력의 구성요소는 서열화 논리, 명제 논리, 조합 논리, 비례 논리, 상관 논리, 변인 통제 논리이다. 또한 애니메이션에서는 비례논리가 변인통제보다 많이 사용되고, 게임에서는 변인통제가 비례논리보다 많이 사용됨을

밝혔다.

3차 분석에서는 Drscratch에서 제시한 CT 점수의 타당성을 검증하기 위해 상관분석과 t-test를 이용하여 분석한다.

4. 결과 분석

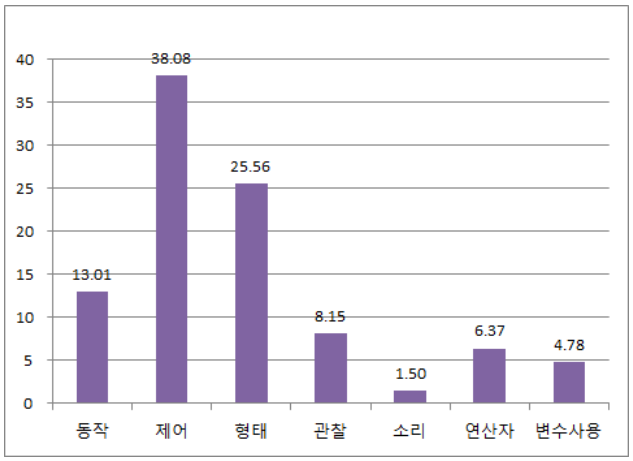
4.1 학습자들의 코드 분석 현황

먼저 실험에 사용된 45개의 프로젝트 코드를 분석한 결과 각 블록의 사용빈도 평균을 살펴보면 <표 7>과 같다.

<표 7> 소스코드의 기초 통계량

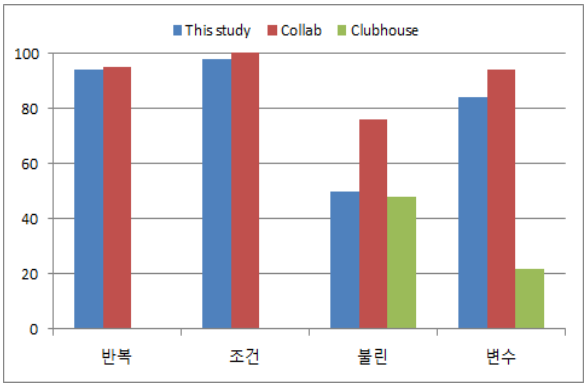
구분	N	최소값	최대값	평균	표준편차
저장	45	1.00	75.00	16.00	14.38
형태	45	1.25	29.00	14.61	6.88
사운드	45	.25	49.00	4.45	7.81
스프라이트	45	3.00	42.00	14.09	7.91
블록수	45	27.00	676.00	269.91	138.40
스택수	45	1.00	133.00	42.29	29.55

<그림 5>와 같이 가장 많이 사용한 블록의 카테고리는 제어블록들이며, 형태, 동작, 관찰의 순서로 나타났다.



<그림 5> 블록 사용 빈도 평균(횟수)

Fields 외[15]는 초보학습자를 대상으로 한 스크래치 Collab 캠프를 실시하고 학습자의 코드를 분석하였다. 이 연구에서는 코드 분석 및 비교를 위해 선행연구인 Clubhouse에서의 코드 분석결과와 비교하였다. <그림 6>과 같이 초보학습자들의 향상을 측정한 블록의 반복, 조건의 사용은 Collab의 학생들과 비슷한 수준이며, 불린과 변수는 Collab의 학생들에 비해서는 낮고 Clubhouse 학생들에 비해서는 높게 나타났다. 물론 각 연구에 참여한 학생들의 수준과 학교급이 달라서 직접비교가 어렵지만 일반적인 특성을 예측해 볼 수 있다.



<그림 6> 주요 블록의 사용빈도 비교

Drscratch에서 제시한 CT 점수와 코드의 레벨이 실제 분석에 사용될 수 있는지 검증하기 위해 self-CT와 self-프로그래밍 능력, 실제 실기점수와의 상관분석을 실시한 결과 <표 8>과 같이 나타났다. Drscratch의 CT 점수는 실기점수와의 유의미한 상관관계가 나타나($p<.05$) 앞으로의 코드 분석에 사용할 수 있는 도구로 분석되었다.

<표 8> 상관관계 분석

		self-CT	self-프로그래밍	Dr-CT	점수
self- CT	Pearson계수	1	.346*	.082	.451**
	p		.029	.613	.003
self-프로그래밍	Pearson계수		1	.124	.386*
	p			.447	.014
Dr-CT	Pearson계수			1	.309*
	p				.031
점수	Pearson계수				1
	p				

*(p<.05), **(p<.01)

Drscracth 의 수준에 대한 분석결과 <표 9>의 결과처럼 45명의 수준이 모두 베이직 이상 개발자와 마스터 수준으로 나타났다.

<표 9> Drscratch 분석 학습자 수준

레벨	명	DrCT 점수	
		평균	표준편차
개발자	13	12.77	1.59
마스터	32	16.45	1.06

4.2 초보학습자들의 코드 특성

초보학습자들이 보이는 특성은 추상화 개념이 형성되지 않은 경우 모든 블록을 순서대로 연결하는 <그림 7>과 같은 형태가 나타난다.

<그림 7> 초보학습자 코드유형 A

실험에 사용된 모든 프로젝트의 스택당 평균 블록수는 9.76개인데, <그림 7>의 경우 스택당 블록수가 25.25개이다. 이 프로젝트는 스프라이트 7개, 블록수 202개, 스택수 8개, 스택당 블록수 25.25개 이다. 스택당 블록수가 많은 경우는 초보학습자들의 전형적인 패턴이라고 볼 수 있다.

또한 초보학습자들의 전형적인 패턴은 병렬을 잘못 사용하는 경우이다. <그림 8>과 같이 총블록수가 302개인데, 스택당 블록수는 3.21개에 불과하다. 코드를 살펴보면 시작버튼 클릭되었을 때 형태가 변경되는 것을 다 흩어놓은 상황이다. 이 프로젝트는 스프라이트 수 29개, 블록수 302개, 스택수 94개, 스택당 블록수 3.21개이다.

<그림 8> 초보학습자 코드유형 B

개발자와 마스터의 분석결과, <표 10>과 같이 카이제곱 값은 5.702이고 자유도가 1일 때, 유의확률은 0.017이므로 유의미한 통계차이가 있음을 알 수 있다. 즉, 레벨의 구분에 따라 스프라이트의 이름을 변경하는 것에 차이가 있다는 것이 나타났다. 이 결과는 Moreno & Robles[17]의 결과와 일치하는 것으로 나타났다.

<표 10> 스프라이트 이름 변경 분석

구 분		변경	무변경	계	X^2 (df)
레벨	개발자	5(35.7)	9(64.3)	14(100.0)	5.702* (1)
	마스터	26(72.2)	10(27.8)	36(100.0)	
구 분		31(62.0)	19(38.0)	50(100.0)	

*(p<.05)

4.3 논리적 사고력과의 관계

윤일규[23]의 연구에서 나타난 바와 같이 논리적 사고력의 비례, 변인, 명제 논리에 해당하는 코드 사용의 빈도가 각 점수와 상관관계가 있는지 분석하였다.

<표 11> 상관관계 분석

		비례	변인	명제	self-CT	self-프로그래밍	Dr-CT
비례	Pearson계수	1	**-.772*** **	-.208	.108	-.059	-.256
	p		.000	.151	.513	.723	.094
변인	Pearson계수		1	-.023	-.286	-.004	**.304***
	p			.877	.074	.982	.042
명제	Pearson계수			1	.183	.195	.194
	p				.259	.228	.201
self-CT	Pearson계수				1	**.346***	.022
	p					.029	.898
self-프로그래밍	Pearson계수					1	-.054
	p						.753
Dr-CT	Pearson계수						1
	p						

*(p<.05), ***(p<.001)

분석결과 <표 11>과 같이 Drscratch의 점수와 변인통제 점수와의 약한 상관관계가 나타났다. 이는 변인통제는 제어, 관찰, 변수의 명령어 빈도수이므로 Drscratch의 데이터 표현, 플로우 제어, 논리적 사고, 유저 상호작용의 요소와 관련이 있어서 상관관계가 성립되는 것으로 분석된다.

<표 12>와 같이 학습자의 수준과 논리적 사고

력요소와의 관계에서는 t-test 결과 유의미한 차이는 나타나지 않았다.

<표 12> 논리적 사고력 요소별 학습자 수준

요소	레벨	N	평균	표준편차
비례	개발자	13	42.72	9.29
	마스터	32	38.85	9.27
변인	개발자	13	44.40	8.31
	마스터	32	51.64	7.91
명제	개발자	13	7.65	4.42
	마스터	32	33.93	122.98

4.4 Drscratch 의 CT 개념 점수 분석

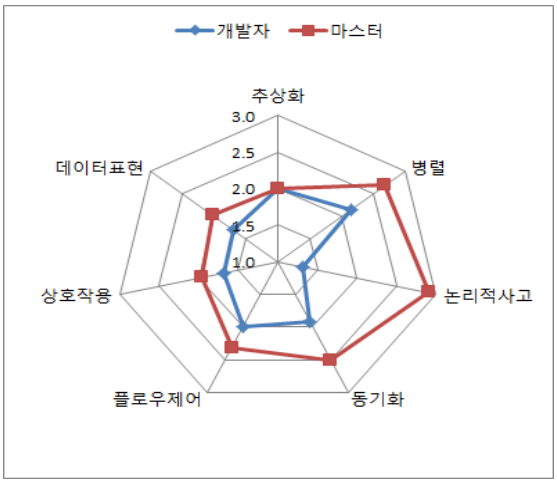
Drscratch의 학습자 수준에 따른 각 점수차이를 분석하기 위해 개발자와 마스터 두 그룹의 차이를 분석하였다. t-test를 실시한 결과 <표 13>과 같이 여러 항목에서 유의미한 차이가 나타났다.

<표 13> Drscratch 요소별 t-test 결과

요소	레벨	N	Mean	S.D.	t	p
추상화	개발자	13	2.00	.00	-	-
	마스터	32	2.00	.00		
병렬	개발자	13	2.15	.99	-1.779	.093
	마스터	32	2.69	.69		
논리적사고	개발자	13	1.31	.75	-7.285***	.000
	마스터	32	2.91	.39		
동기화	개발자	13	1.92	.76	-2.978**	.005
	마스터	32	2.50	.51		
플로우제어	개발자	13	2.00	.00	-2.985**	.005
	마스터	32	2.31	.59		
상호작용	개발자	13	1.69	.48	-2.020	.064
	마스터	32	1.97	.18		
데이터표현	개발자	13	1.69	.48	-2.166*	.036
	마스터	32	2.03	.47		
복사	개발자	13	2.54	2.26	1.181	.244
	마스터	32	1.91	1.30		
죽은코드	개발자	13	1.38	1.80	1.388	.172
	마스터	32	.75	1.19		
개체속성	개발자	13	7.08	3.88	-.868	.390
	마스터	32	10.56	14.17		

*(p<.05), **(p<.01), ***(p<.001)

개발자와 마스터의 구분이 각 요소별 점수를 합산하여 구분하기 때문에 요소별로 유의미한 차이가 나타나는 것이지만 논리적 사고, 동기화, 플로우 제어, 데이터 표현 요소에서는 통계적으로 유의미한 차이가 나타났다. 이러한 결과는 Fields 외[16]의 연구결과에서 변수, 불린연산의 사용의 중요성과 방송하기/받기의 사용의 중요성이 코드 분석을 통해서도 나타난 것으로 분석된다.



<그림 9> 학습자 수준에 따른 요소 차이

세부분석을 위해 중요 요소인 변수사용과 방송사용에 대한 수준별 t-test를 실시하였으나 <표 14>와 같이 유의미한 차이는 나타나지 않았다. 마스터 수준의 사용 빈도가 높게 나타났지만 분산이 넓게 분포되어 있어서 통계상으로 유의미한 차이는 나타나지 않았다. 현재 데이터의 분산의 차이가 크게 나타나 추가 데이터의 보강을 통한 검증이 필요하다.

<표 14> 변수 및 방송사용의 차이 분석

요소	레벨	N	평균	표준편차
변수 사용	개발자	13	3.74	3.10
	마스터	32	4.98	3.28
방송 사용	개발자	13	6.35	5.02
	마스터	32	12.33	13.73

5. 결론 및 제언

본 연구에서는 CT 교육에서 학습자 평가를 위한 방안으로 스크래치 코드 분석을 실시하고 평가방안을 모색하였다. CT 교육 후 학습자들이 제작한 프로젝트 파일을 분석한 결과 다음과 같은 결론을 도출하였다.

첫째, 학습자들이 가장 많이 사용하는 명령어 블록은 제어, 형태, 동작, 관찰 순이며 중요한 명령어인 불린연산과 변수의 사용은 국외 다른 연구결과에 비해 중간 수준으로 나타났다.

둘째, 초보학습자들의 잘못된 습관으로는 모든 명령어를 분할하지 않고 그대로 쌓거나, 너무 많은 병렬을 사용하거나, 스프라이트의 이름을 고치지 않는 것이 나타났다.

셋째, 윤일규[22]의 연구에서 제시된 논리적 사고력 요소와 명령어 블록과의 관계를 적용한 분석에서는 유의미한 결과가 나타나지 않았다. 이 부분에 대해서는 다양한 데이터 수집을 통한 분석이 필요하다.

넷째, Drscratch의 CT 개념 요소의 분석에서는 논리적 사고, 동기화, 플로우 제어, 데이터 표현 항목에서 학습자 수준에 따른 유의미한 차이가 나타났다. 학습자 수준이 높을수록 각 요소의 점수가 높게 나타났으므로 소스 코드의 각 항목이 학습자의 수준을 평가하는 요소로 사용할 수 있음을 시사한다.

마지막으로 위의 결과를 종합하면 학습자들이 제작한 스크래치 코드 분석을 통하여 학습자의 수준을 평가하는 것이 가능하며, CT 개념의 하위요소 학습상황을 분석하고 평가하는 방안으로 활용할 수 있다는 것을 알 수 있다.

본 연구의 결과를 일반화하기 위해서는 다음과 같은 부분을 고려해야 한다.

첫째, 본 연구에서 사용한 데이터는 45개의 프로젝트이므로 많은 양의 초보학습자들의 프로젝트 분석을 통한 패턴의 일반화 형태를 검증할 필요가 있다.

둘째, 코드 분석을 통한 CT 개념 학습 평가의 경우, 학습자가 직접 제작하지 않거나 다른 사람의 코드를 그대로 차용하는 경우가 발생하는 등 실제 학습자의 능력에 대한 검증이 이루어지지 않을 수 있으므로 교육현장에서 적용할 때는 세심한 주의가 필요하다.

셋째, 실질적인 학습자의 CT 개념 평가를 위해서는 초기코드와 최종코드의 차이 분석을 통한 향상 정도를 평가하는 방법을 고려해야 한다.

마지막으로 코드사용을 개념 습득으로만 볼 것이 아니라 프로젝트의 장르와 연관지어 학습 코스를 구성할 수 있고, 학습자의 특성을 보여주는 프로파일 형태의 평가를 제공할 필요가 있다.

참 고 문 헌

[1] National Research Council (2011). *Report of a Workshop of Pedagogical Aspects of Computational Thinking*. Washington, D. C., The National Academies Press.

[2] Nacce & CAS (2014). Computing in the national curriculum: a guide for secondary teachers. 2015. 7. 14일 검색 http://www.computing atschool.org.uk/data/uploads/cas_secondary.pdf

[3] Tucker, A. et al. (2006). *A Model Curriculum for K-12 Computer Science*: New York, CSTA.

[4] 김경훈 외 (2015). 정보과 교육과정 시안 개발 연구. 한국교육과정평가원. 연구보고 CRC 2015-17

[5] 김영애 외 (2015). 소프트웨어 교육 운영 지침 개발 연구. 한국교육학술정보원.연구보고 CR 2015-3

[6] Wing, J. M. (2008). Computational thinking and thinking about computing, *Philosophical Transactions of The Royal Society, 366*, 3717-3725.

[7] CSTA & ISTE (2011). Operational definition of computational thinking for K-12 Education. 2015. 7. 14일 검색 http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

[8] Code.org (2014). Computational Thinking. 2015. 7. 14일 검색 http://learn.code.org/unplugged/unplug2.pdf.

[9] Brennan, K., Chung, M., & Balch, C. (2015), *Creative Computing*, 2015. 7.14일 검색 http://scratched.gse.harvard.edu/guide/

[10] Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Paper presented at annual American Educational Research Association meeting, Vancouver, BC, Canada*.

[11] Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010). Towards the Automatic Recognition of Computational Thinking for Adaptive Visual Language Learning. I*n Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC '10). IEEE Computer Society, Washington, DC, USA,* 59-66.

[12] Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013). Hairball: lint-inspired static analysis of scratch projects. *In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). ACM, New York, NY, USA,* 215-220.

[13] Coalition of Essential Schools (2010). Overview of alternative assessment approaches. 2015. 7. 14일 검색 http://www.essentialschools.org/resources/115.html

[14] 최형신 (2014). Computational Thinking 역량 계발을 위한 수업 설계 및 평가 루브릭 개발. **한국정보교육학회논문지, 18(1)**, 57-64.

[15] Kirkov, R., & Agre, G. (2010). *Source Code Analysis - An Overview*. BULGARIAN ACADEMY OF SCIENCES CYBERNETICS AND INFORMATION TECHNOLOGIES, 10(2), 60-77.

[16] Fields, D. A., Kafai, Y. B., Strommer, A., Wolf, E., & Seiner, B. (2014). INTERACTIVE STORYTELLING FOR PROMOTING CREATIVE EXPRESSION IN MEDIA AND CODING IN YOUTH ONLINE COLLABORATIVES IN SCRATCH. *Constructionism 2014 : Constructionism 2014 International Conference*

[17] Moreno, J. & Robles, G. (2014). Automatic etection of Bad programming Habits in Scratch: A Preliminary study. *Frontiers in Education Conference (FIE), 2014 IEEE*, 1-4.

[18] Burke, Q., & Kafai, Y. B. (2012). The writers' workshop for youth programmers: digital storytelling with scratch in middle school classrooms. *In Proceedings of the 43rd ACM technical symposium on Computer Science Education, SIGCSE '12,* 433-438.

[19] Scrape. 2015. 7. 14일 검색 http://happyanalyzing.com/downloads/scrapeLocal

[20] Drscratch. 2015. 7. 14일 검색 http://drscratch. programamos.es/

[21] Martin, T., & Fields, D. (2014). Macro data for micro learning: Developing FUN! for Automated Assessment of Computational Thinking in Scratch. *Poster presented at the 2014 Cyberlearning Summit, Madison, WI.*

[22] 김수환 (2015). Computational Thinking 교육에서 나타난 컴퓨터 비전공 학습자들의 어려움 분석. **한국컴퓨터교육학회 논문지, 18(3)**, 49-57.

[23] 윤일규 (2010). **정보 교육의 문제해결과정에서 요구되는 논리적 사고력의 구성요소**. 석사학위 논문, 고려대학교 대학원, 서울.

김 수 환

1999 인천교육대학교 (교육학학사)
2006 경인교육대학교 컴퓨터교육과(교육학석사)
2011 고려대학교 컴퓨터교육과(이학박사)
2013~2014 경인교육대학교 겸임교수
2014~현재 총신대학교 교양교직과 조교수
관심분야: 컴퓨터교육, EPL, 컴퓨터적사고, CSCL, Computational Literacy
E-Mail: skim@csu.ac.kr