컴퓨터교육학회논문지 2020년 제 23 권 제 3 호
https://doi.org/10.32431/kace.2020.23.3.005

스크래치를 이용한 문제해결 프로그래밍에서 CT-TDPS 학습 모형의 효과성 연구

A Study on the Effectiveness of CT-TDPS Learning Model in Problem Solving Programming using Scratch

김영직† · 김성식††
Young-Jik Kim† · Seong-Sik Kim††

요 약

21세기 미래 인재에게 필요한 핵심 역량으로 컴퓨팅 사고력이 주목받고 있다. 국내외적으로 컴퓨팅 사고력 향상을 위한 소프트웨어 교육이 한창이다. 그중에서 문제해결 프로그래밍 교육은 컴퓨팅 사고력 향상에 도움이 된다. CT-TDPS 학습 모형은 복잡한 문제들을 모듈화하는 분해, 추상화 사고 과정과 이를 구현하는 반복적·점증적 프로그래밍 방식인 애자일(Agile) 개발 방식을 따른다. 본 연구에서는 스크래치를 이용한 문제해결 프로그래밍 교육에 CT-TDPS 학습 모형을 적용하여 컴퓨팅 사고력 향상을 확인하고자 하였다. 연구 결과, CT-TDPS 학습 모형을 적용한 문제해결 프로그래밍 교육에서 컴퓨팅 사고력의 하위 요인인 컴퓨팅 개념, 컴퓨팅 수행, 컴퓨팅 관점에서 모두 향상이 되었음을 확인할 수 있었다. 그리고, Dr.Scratch 자동 평가 결과에 대한 t 검정 결과 실험집단에서 유의한 차이가 있음을 확인하였다.

주제어: 컴퓨팅 사고력, CT-TDPS, 문제해결 프로그래밍 학습 모형, 테스트주도개발, TDD, Dr.Scratch

ABSTRACT

Computational Thinking(CT) is drawing attention as a core competency required for future talent in the 21st century. Software education for improving CT ability at home and abroad is in full swing. Among them, problem-solving programming education helps to improve CT ability. The CT-TDPS learning model follows the decomposition, abstraction thinking process, which modularizes complex problems, and the Agile development method, which is an iterative and incremental programming method to implement it. In this study, we tried to confirm the improvement of CT ability by applying CT-TDPS learning model to problem solving programming education using Scratch. As a result of the study, it was confirmed that in the problem solving programming education using the CT-TDPS learning model, it improved in all aspects of computing concept, computing performance, and computing perspective, which are sub-factors of CT ability. In addition, it was confirmed that there was a significant difference in the experimental group as a result of the t-test on the Dr.Scratch automatic evaluation result.

Keywords: Computational Thinking, CT-TDPS, Problem Solving Programming Learning Model, Test Driven Development, TDD, Dr.Scratch

1. 서론

2015년, 세계경제포럼(WEF)에서는 인류 사회가 4차 산업혁명을 통해 대변혁을 맞이할 것이라 예견하면서 4차 산업혁명 시대를 맞이할 준비가 필요함을 강조하였다[1].

특히, 일자리 분야에서 많은 변화가 일어날 것을 예측하였는데, 미래 인재에게 필요한 핵심 역량으로 컴퓨팅 사고력을 들 수 있다. 컴퓨팅 사고력은 21세기에 누구나 가져야 하는 보편적 능력인 동시에[2], 미래 인재에게 필요한 핵심 역량 중 하나이다[3].

컴퓨팅 사고력은 일반적으로 컴퓨터를 이용하여 문제를 해결하는 사고 능력으로 정의 내릴 수 있다[4][5].

세계적으로도 학생들의 컴퓨팅 사고력 역량을 높이기

† 정 회 원: 한국교원대학교 컴퓨터교육과 박사수료
†† 종신회원: 한국교원대학교 컴퓨터교육과 교수(교신저자)

논문접수: 2020년 03월 19일, 심사완료: 2020년 04월 24일, 게재확정: 2020년 04월 28일

위한 소프트웨어교육이 활발하다. 국내에서도 창의적 문제해결 능력 및 논리적 사고력을 신장시키기 위하여 소프트웨어교육이 시행되고 있는데 특히, 문제해결과정을 통해 컴퓨팅 사고력을 높이려는 문제해결 프로그래밍을 강조하고 있다[6].

교육부(2015)는 2015년 개정 정보 교육과정에서 소프트웨어교육을 강화하면서 컴퓨팅 사고력 향상을 교육 목표로 정하고, 내용적인 면에서 문제해결 프로그래밍 교육을 강조하였다.

실제로 문제해결 프로그래밍 교육을 통해서 컴퓨팅 사고력이 향상되었음을 입증하는 연구들을 보면[7][8][9], 문제해결 프로그래밍 교육이 컴퓨팅 사고력 향상에 직접적인 영향을 미치는 것임을 알 수 있다.

프로그래밍 교육이 컴퓨팅 사고력 발달에 유효한 방법임에는 틀림이 없음에도 불구하고, 프로그래밍 학습에 적지 않은 어려움을 호소하고 있는 것이 현실이다. 이를 극복하기 위해 다양한 프로그래밍 교수 학습 모형이 등장했는데[10][11][12][13], 이러한 교수 학습 모형의 대부분은 분석-설계-구현-테스트라는 일반적인 소프트웨어 개발 방식인 폭포수(Waterfall) 모형의 프로그래밍 개발 절차를 따르고 있다.

폭포수 모형에서는 테스트가 마지막 단계(Test Last)에서 이뤄진다. 이는 선행 단계에서 문제가 있거나 변경이 발생했을 때 이를 해결하는데 상당한 시간과 노력이 소요된다는 단점이 있다. 최근에는 개발 주기를 빠르게 반복하면서 반복적·점증적으로 개발하는 방식을 선호한다[14][15][16].

CT-TDPS(CT based Test Driven Problem Solving) 학습 모형은 복잡한 문제들을 모듈화해서 반복적·점증적으로 프로그래밍하는 애자일(Agile) 개발 방식을 따른다. 본 연구에서는 스크래치를 이용한 문제해결 프로그래밍 교육에 CT-TDPS 학습 모형을 적용하여 효과성을 살펴보고자 하였다.

2. 이론적 배경

2.1 컴퓨팅 사고력 스킬 역량

컴퓨팅 사고력은 문제를 해결하는 사고 과정으로 정의할 수 있으며, 추상화와 자동화 과정을 거친다. 본 연구에서는 문제해결 스킬(Problem Solving Skills, PSS)관점에서 컴퓨팅 사고력의 추상화 스킬과 자동화 스킬을 정의하였다.

PSS는 일반적 문제해결스킬(generic PSS)과 전문분야 문제해결스킬(discipline-specific PSS)로 나뉠 수 있다. generic PSS는 실생활의 모든 분야에 사용될 수 있는 스킬이고 수학, 화학, 물리 등 discipline-specific PSS와는 구별된다[17].

Unruh & Rosenbloom(1989)은 모든 분야에서의 문제해결에 적용될 수 있는 추상화를 일반적 추상화(generic abstraction)라고 하였고, 특정 분야의 지식(domain specific knowledge)이 필요한 추상화와 구별된다고 하였다[18].

일반적 문제해결스킬에 사용되는 추상화를 일반적 추상화 스킬이라고 할 수 있으며, 전문분야인 컴퓨터 프로그래밍 문제해결스킬에 사용되는 추상화를 컴퓨팅 추상화 스킬이라 할 수 있다. 또, 컴퓨터 프로그래밍 문제해결에서의 프로그래밍 능력을 컴퓨팅 자동화 스킬이라고 할 수 있다.

표 1. 일반적 문제해결스킬과 컴퓨터 프로그래밍 문제해결스킬

문제해결스킬 (Problem solving Skills)		
일반적 문제해결스킬 (generic Problem Solving Skills)	컴퓨터 프로그래밍 문제해결스킬 (Computer Programming Problem Solving Skills)	
일반적 추상화 스킬	컴퓨팅 추상화 스킬	컴퓨팅 자동화 스킬

<표 1>에서 문제해결스킬이 일반적 추상화 스킬, 컴퓨팅 추상화 스킬, 컴퓨팅 자동화 스킬로 구분되어짐을 알 수 있다.

문제해결 프로그래밍 관점에서의 문제해결은 추상화와 자동화로 구분할 수 있고, 추상화는 일반적 추상화 스킬과 컴퓨팅 추상화 스킬을 포함하며, 자동화는 컴퓨팅 자동화 스킬을 의미한다고 할 수 있다. 즉, 문제해결 프로그래밍에서의 문제가 실생활에서의 일반적인 문제들로 구성될 수 있기 때문에 일반적 추상화 스킬을 포함한 컴퓨팅 사고력 역량의 재정의가 필요하다. 그렇기에 본 연구에서는 <표 2>와 같이 문제해결 프로그래밍에서의 컴퓨팅 사고력 역량을 일반적 추상화 스킬 역량, 컴퓨팅 추상화 스킬 역량, 컴퓨팅 자동화 스킬 역량을 모두 포함하는 것으로 정의하였다.

표 2. 문제해결 프로그래밍에서의 컴퓨팅 사고력 역량

컴퓨팅 사고력 역량		
추상화 스킬 역량		자동화 스킬 역량
일반적 추상화 스킬 역량	컴퓨팅 추상화 스킬 역량	컴퓨팅 자동화 스킬 역량

2.2 컴퓨팅 사고력 스킬 역량 평가 도구

González, León & Robles(2017)는 Dr.Scratch 평가가 프로그래밍 CT(컴퓨팅 사고력) 지필 평가와 유의한 상관을 보인다고 하면서 [그림 1]과 같이 블룸의 텍사노미(taxonomy)와 컴퓨팅 사고력 평가 도구와의 위치 관계를 선정하였다[19]. 해당 연구에서 컴퓨팅 사고력 평가 도구인 CTt는 이해하기, 기억하기 평가에, 베브라스는 분석하기, 적용하기 평가에, Dr.Scratch는 창조하기, 평가하기의 평가에 적합하다고 제시하였다. Dr.Scratch는 컴퓨팅 사고력 스킬을 활용한 결과물인 스크래치 프로그램을 직접적으로 평가한다는 점에서 컴퓨팅 사고력의 추상화 스킬과 자동화 스킬 역량 모두를 평가하는데 적합하다고 할 수 있다.

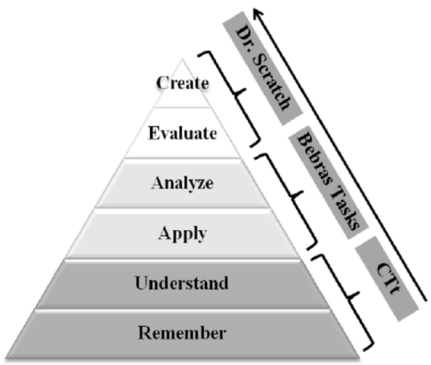

그림 1. Bloom's taxonomy and CT assessment tools[19]

2.3 CT-TDPS 학습 모형

Computational Thinking-based Test Driven Problem Solving(CT-TDPS) 모형은 [그림 2], <표 3>에서와 같이 문제해결과정에서 컴퓨팅 사고력 요소를 기반으로 한 문제해결 프로그래밍 학습 절차에 애자일(Agile) 개발 방식인 테스트주도개발(TDD)과 다양한 프로그래밍 수업 기법(PDT, IPO Chart, Pseudo code)을 적용한 모형으로 문제해결을 위한 테스트 코드를 먼저 작성함으로써 명확한 개발 목표를 설정하고 즉각적인 피드백을 제공해 설계상의 오류나 문제점을 프로그래밍 과정에서 바로 확인할 수 있도록 하여 학생들로 하여금 문제해결에 자신감을 갖게 하고 프로그래밍 학습에 몰입할 수 있도록 한 문제해결 프로그래밍 학습 모형이다[14].

[그림 3]은 CT-TDPS를 적용한 문제해결 프로그래밍의 개발 흐름을 보여준다. 문제정의 및 추론 단계에서 개발할 모듈과 입·출력값을 확인한 후, 모듈 #1부터 점증적으로 실행 코드를 완성한다. 실행 코드 수정 후에는 테스트 코드의 결과를 확인하고 Pass이면 다음 모듈의 테스트 코드와 실행 코드를 작성한다. 중복제거 또는 알고리즘 개선이 있는 경우에는 전체 테스트 코드가 Pass인지 확인한다.

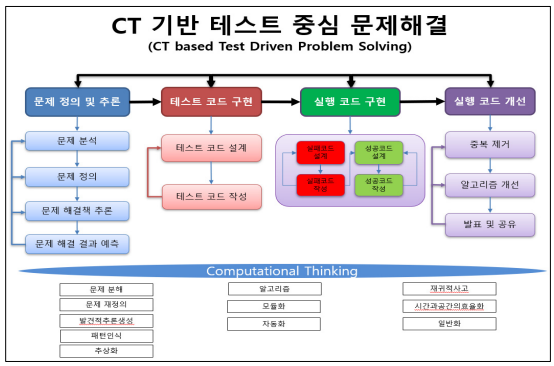

그림 2. 컴퓨팅 사고력 기반 테스트 중심 문제해결 (CT-TDPS) 학습 모형

표 3. CT-TDPS 모형, 문제해결 프로그래밍 학습절차, 테스트주도개발 절차, 수업 기법

단계	CT-TDPS 모형	문제해결 프로그래밍 학습 절차	테스트주도 개발(TDD) 절차	수업 기법
1 단계	·문제 정의 및 추론	·문제 분석 ·문제 정의 ·해결책 고안 ·아이디어 구상		·PDT ·IPO Chart ·Pseudo code
2 단계	·테스트 코드 구현	·해결책 실천 ·구현	·Write a test that fails(실패)	
3 단계	·실행 코드 구현		·Make the code work(성공)	
4 단계	·실행 코드 개선	·평가	·Eliminate redundancy (리팩토링)	

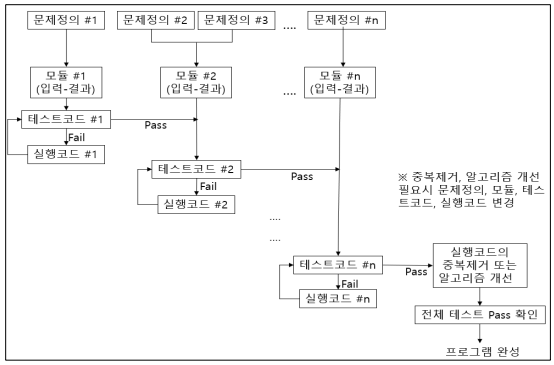

그림 3. CT-TDPS를 적용한 문제해결 프로그래밍 개발 흐름

3. 연구 방법 및 절차

3.1 연구 대상

본 연구의 연구 대상은 <표 4>와 같이 K 교육대학교의 컴퓨터과학 비전공 1학년 학생들로 구성하였다. 연구에서 실험집단은 27명, 비교집단은 27명으로 구성된다. 실험집단과 비교집단 간의 수업 비교를 통해서 제안된 학습 모형의 효과성을 검증한다.

표 4. 연구 대상

집단	심화 학과	학생수
실험집단	사회교육과	27
비교집단	국어교육과	27
총 학생수		54

3.2 연구 설계

CT-TDPS 학습 모형의 효과성 검증에서는 학습자의 컴퓨팅 사고력에 미치는 영향을 분석하기 위해 실험집단과 비교집단 전후검사 설계를 사용하였으며, 구체적인 연구의 실험설계는 <표 5>와 같다.

표 5. 연구의 실험설계

실험집단	O_1	X_1	O_2
비교집단	O_3	X_2	O_4

O_1, O_3 : 사전검사 (컴퓨팅사고력, 문제해결성향)
O_2, O_4 : 사후검사 (컴퓨팅사고력, 문제해결성향)
X_1 : CT-TDPS 모형을 적용한 문제해결 프로그래밍 교육
X_2 : 문제중심학습(PBL) 기반의 문제해결 프로그래밍 교육

3.3 연구 도구

본 연구에서는 학습 모형의 효과를 검증하기 위하여 컴퓨팅 사고력, 문제해결성향 검사 도구를 선정하였다. 컴퓨팅 사고력 검사는 자기보고식 평가 설문지를 실험 사전, 사후에 실시하였고, 학생들의 프로그래밍 산출물을 이용하여 Dr.Scratch 자동 평가를 실시하였다.

3.3.1 컴퓨팅 사고력 자기 보고식 설문지 검사

컴퓨팅 사고력 평가를 위해 Brennan & Resnick(2012)이 개발한 컴퓨팅 사고 프레임워크를 기반으로 최형신 외(2015)가 개발한 스크래치 수업 후의 컴퓨팅 사고력 자기보고식 평가 설문지를 사용하였다[20]. 본 연구에서의 검사지 신뢰도 Cronbach's α는 0.960이다.

3.3.2 컴퓨팅 사고력 Dr.Scratch 평가 검사

Oluk & Korkmaz(2016)은 초등학생을 대상으로 스크래치 프로그래밍 수업을 실시한 후 컴퓨팅 사고력 스킬을 창의력, 문제해결, 알고리즘적 사고, 협업, 비판적 사고 요소를 갖는 Computational Thinking Level Scale을 이용하여 평가하였는데 Dr.Scratch로 평가한 결과와 매우 높은 상관관계를 가진다는 것을 확인하였다. 학생들의 스크래치를 통한 프로그래밍 스킬 발달은 컴퓨팅 사고력 스킬 향상에 영향을 미치며, 컴퓨팅 사고력 스킬 향상이 스크래치 프로그래밍 능력을 향상시킬 수 있다고 하였다. 그는 Dr.Scratch의 평가 결과와 컴퓨팅 사고력 스킬은 매우 높은 상관관계를 갖는다고 하였다[21].

<표 6>은 Dr.Scratch CT 평가 요소 및 점수이다. 흐름제어, 자료 표현, 추상화, 상호작용, 동기화, 병렬화, 논리적 사고 총 7가지 요소로 구성되며 각 요소 당 해당없음(N/A), 초급, 중급, 상급 수준으로 0~3점이 부여된다. 총점수는 21점 만점이다.

표 6. Dr.Scratch CT 평가 요소 및 점수

CT 평가 요소 (CT dimension)	레벨(Level)			
	N/A	초급	중급	상급
Flow control	0	1	2	3
Data representation	0	1	2	3
Abstraction	0	1	2	3
User interactivity	0	1	2	3
Synchronization	0	1	2	3
Parallelism	0	1	2	3
Logic	0	1	2	3
총점	0 ~ 21			

3.3.3 문제해결성향 검사

학생들의 문제해결성향을 측정하기 위하여 Heppner와 Petersen(1982)이 개발한 Problem Solving Inventory를 박주연(2015)이 번안한 설문지를 사용하였다[7]. 본 연구에서의 검사지 신뢰도 Cronbach's α는 0.852이다.

3.4 연구 절차

연구는 12주간 매주 2차시로 총 24차시 수업시간을 운영하였다(오리엔테이션, 발표, 기말시험 제외). 실험 처치 수업에 들어가기 전에 컴퓨팅 사고력 검사(자기보고식), 문제해결성향 검사를 사전에 실시하여 실험집단과 비교집단의 동질성을 확인하였고, 수업 종료일에 컴퓨팅 사

고력 검사(자기보고식), 문제해결성향 검사를 실시하고, 학생들의 스크래치 산출물을 Dr.Scratch로 평가한 후 두 집단의 차이를 관찰하였다. [그림 4]는 실험집단과 비교집단에 적용한 수업 내용이다. 9~12주에서 실험집단과 비교집단에 각각 CT-TDPS 모형과 문제중심학습(PBL) 모형을 나누어 적용하였다.

주	수업주제	수업내용	수업방법
1	오리엔테이션		강의
2	컴퓨터과학 소개	- 컴퓨터 역사 - 컴퓨터과학 개념 - 컴퓨터과학 교육의 필요성	강의
3	코딩 이해	- 매직펜 오디토리엄연주 - 컴퓨터 프로그래밍 사용하기	강의
4	소프트웨어 중심사회의 이해	- 우리 생활과 소프트웨어 - 4차 산업혁명의 이해 - 소프트웨어 중심사회	온라인 학습(1)
5	코딩 기초 이해하기 (순차, 조건, 반복, 함수, 변수)	- 스크래치 이해 - 스크래치 기본 블록 - 스프라이트 동작 시키기	실습
6	코딩 기초 이해하기 (리스트)	- FizzBuzz 게임 시뮬레이션 하기	실습
7	코딩 기초 이해하기 (재귀)	- 숫자더하기 계산기 만들기	실습
8	2015개정 SW교육과정의 이해	- 소프트웨어 교육 방법 - 2015개정 SW교육과정의 이해	온라인 학습(2)
9	수업 모형 이해	학습모형 기반 프로그래밍 이해	실습
10	실생활 문제해결 실습 #1	학습모형 기반 문제해결 프로그래밍 실습	실습
11	실생활 문제해결 실습 #2	학습모형 기반 문제해결 프로그래밍 실습	실습
12	실생활 문제해결 실습 #3	학습모형 기반 문제해결 프로그래밍 실습	실습
13	컴퓨팅 사고력	- 컴퓨팅 사고력의 역사와 개념 - 컴퓨팅 사고력의 하위요소 및 사례 - 컴퓨팅 사고력 기반 문제해결	온라인 학습(3)
14	실습 발표	평가 설문조사	평가 설문조사
15	기말시험	지필평가	시험

그림 4. 수업 내용

3.5 결과 분석

수업 전후 수집한 컴퓨팅 사고력 자기보고식 설문 결과, Dr.Scratch 평가 결과, 문제해결성향 검사 결과에 대한 분석은 표준 통계 프로그램인 SPSS 20을 사용하였다.

4. 연구 결과

4.1 사전 검사

두 집단 간의 동질성 여부를 검증하기 위하여 실험 처치에 앞서 사전 검사를 실시하였다.

표 7. 컴퓨팅 사고력 사전 검사 결과

하위 요인	집단	인원	평균	표준편차	t	p
컴퓨팅 개념	실험	27	10.22	4.643	.052	.959
	비교	27	10.15	5.776		
컴퓨팅 수행	실험	27	6.44	2.636	.801	.309
	비교	27	5.70	2.658		
컴퓨팅 관점	실험	27	5.29	1.857	.483	.631
	비교	27	5.03	2.084		

표 8. 문제해결성향 사전 검사 결과

하위 요인	집단	인원	평균	표준편차	t	p
문제해결 자신감	실험	27	29.78	6.172	-.443	.660
	비교	27	30.56	6.727		
접근-회피 양식	실험	27	45.96	8.671	.276	.338
	비교	27	43.93	6.690		
개인의 통제력	실험	27	15.70	4.348	.635	.415
	비교	27	16.63	3.914		

컴퓨팅 사고력 사전 검사(<표 7>)와 문제해결성향 검사(<표 8>) 결과 모든 하위 요인들에서 비교집단과 실험집단이 유사한 평균값을 가지며, 각 하위 요인별 평균 차이는 유의수준 0.05보다 높게 나타나 실험집단과 비교집단이 동질집단임을 확인하였다.

4.2 사후 검사

실험 처치 후 두 집단 간의 컴퓨팅 사고력, 문제해결성향, Dr.Scratch 차이를 검증하기 위하여 독립표본 t 검정을 실시하였고, 집단 내 차이 비교를 위하여 대응표본 t 검정을 실시하였다.

<표 9>는 실험집단 내에서의 컴퓨팅 사고력 자기보고식 검사의 사전-사후 결과를 보여준다. 컴퓨팅 사고력 하위 요인인 컴퓨팅 개념($p<0.001$), 컴퓨팅 수행($p<0.001$), 컴퓨팅 관점($p<0.001$)에서 유의하게 향상되었음을 확인할 수 있다. <표 10>은 컴퓨팅 사고력 자동 평가 도구인 Dr.Scratch의 집단 간 사후 평가 결과를 보여준다. 실험집단의 CT 점수가 비교집단보다 높은 평균을 보이며 유의한 차이($p<0.001$)가 있음을 확인할 수 있다.

표 9. 실험집단의 컴퓨팅 사고력 사전-사후 검사 결과

하위 요인	집단	인원	평균	표준편차	t	p
컴퓨팅 개념	사후	27	19.44	3.434	8.297	.000***
	사전	27	10.22	4.644		
컴퓨팅 수행	사후	27	8.93	1.980	3.912	.000***
	사전	27	6.44	2.636		
컴퓨팅 관점	사후	27	8.11	1.553	6.043	.000***
	사전	27	5.29	1.857		

표 10. 집단 간 Dr.Scratch 사후 평가 결과

평가	집단	인원	평균	표준편차	t	p
Dr.Scratch CT 점수	실험	27	40.44	.801	10.783	.000***
	비교	27	32.85	3.570		

4.3 검사 해석

실험 결과, 문제해결 프로그래밍 수업에서 CT-TDPS 모형을 적용한 집단이 문제중심학습(PBL) 모형을 적용한 집단보다 컴퓨팅 사고력 향상에 유의미한 차이가 있음을 확인하였다.

실험집단 내에서 컴퓨팅 사고력 자기보고식 평가는 컴퓨팅 개념, 컴퓨팅 수행, 컴퓨팅 관점 하위 요인 모두에서 유의한 향상이 관찰되었다. 또, 집단 간 Dr.Scratch 자동 평가 결과에서 실험집단이 비교집단보다 유의미한 차이가 관찰되었다.

학생들이 작성한 스크래치 산출물을 살펴보면 비교집단은 블록이 하나의 덩어리로 작성된 경향을 보인 반면, 실험집단은 모듈로 구분되는 경향을 보인다. 이는 CT-TDPS 모형에 적용된 테스트주도개발 방식의 특징인 모듈화 설계, 모듈식 구현 성향이 영향을 미친 것으로 해석할 수 있는데, Khanam & Ahsan(2017)은 테스트주도개발의 효과성 연구에서 테스트주도개발의 주요한 특징으로 프로그램 구조가 모듈화되는 경향을 보이며 이를 통해 코드의 이해와 디버깅이 용이하며, 결함을 발견하고 교정하는데 도움이 된다[22]고 한 것과도 일치한다. 또, 애자일 개발 방식인 반복적 프로그래밍 구현이 컴퓨팅 사고력 발달에 유의한 영향을 미친다는 연구 결과[23]를 놓고 볼 때, CT-TDPS 학습 모형에 적용된 반복적·점증적 구현 방식이 컴퓨팅 사고력 향상에 영향을 미친 것으로 해석할 수 있다.

5. 결론 및 제언

본 연구에서는 스크래치를 이용한 문제해결 프로그래밍 교육에서 CT-TDPS 학습 모형이 컴퓨팅 사고력 향상에 효과가 있음을 확인하였다.

CT-TDPS 모형을 적용한 실험집단에서 컴퓨팅 사고력 자기보고식 평가 결과가 향상되었고, Dr.Scratch CT 평가에서 CT-TDPS 모형을 적용한 실험집단이 비교집단보다 높은 CT 점수를 획득하여 컴퓨팅 사고력 향상에 효과가 있음을 확인하였다.

실험집단의 스크래치 산출물은 모듈로 구성되는 경향을 보이는데, 이는 CT-TDPS 모형에 적용된 테스트주도개발 방식의 특징인 모듈화 설계, 모듈식 구현 성향이 영향을 미친 것으로 해석할 수 있으며, 모듈식 프로그래밍이 코드의 이해와 디버깅을 용이하게 하며, 결함을 발견하고 교정하는데 도움이 된다고 봤을 때, 모듈화가 프로그래밍 능력과 컴퓨팅 사고력 발달에 영향을 미친다고 판단된다.

CT-TDPS 모형이 컴퓨팅 사고력을 기반으로 한 문제해결 프로그래밍 학습 절차와 테스트주도개발의 모듈식 개발, 다양한 효과적인 프로그래밍 기법을 적용하였기에 문제해결 프로그래밍 과정에서 컴퓨팅 추상화 스킬과 자동화 스킬 향상에 긍정적인 효과가 있는 것으로 판단할 수 있다. 향후 이러한 효과의 요인을 분석하기 위한 후속 연구가 필요해 보인다.

본 연구는 대학생을 대상으로 실험한 결과로 초중등학생에게 적용하기 위해서는 후속 연구가 필요하다.

참고문헌

[1] Pil-Sung Jang. (2016). 2016 Davos Forum: Our strategy for the upcoming fourth industrial revolution. *SCIENCE & TECHNOLOGY POLICY, 26*(2), 12-15.

[2] Wing, J.M. (2006). Computational Thinking. *CACM Viewpoint*, 49(3), 33-35.

[3] 이재호 · 백승욱 · 이윤조 · 이경화 (2018). 미래인재 역량 정립 연구. **한국창의정보문화연구,** 4(3), 311-320.

[4] 김영직 · 김성식 (2019a). CT-TDPS를 활용한 스도쿠 퍼즐 프로그램 제작. **한국컴퓨터교육학회 학술발표대회논문집,** 24(1), 131-134.

[5] ISTE & CSTA. (2011). Computational Thinking Leadership Toolkit 1st edition. Http://csta.acm.org/Curriculum/sub/CurrFiles/471.11CTLeadershipToolkit-SP-vF.pdf

[6] 교육부 (2015a). **실과(기술·가정)/정보과 교육과정**. 교육부 고시 제2015-74호 [별책 10].

[7] 박주연 (2015). **Scratch 프로그래밍 수업에서 학습자 특성, 학습몰입, 학습효과의 구조적 관계 규명**. 박사학위 논문, 이화여자대학교.

[8] 신수범 (2015). 스크래치 소프트웨어 교육을 통한 컴퓨팅 사고력 향상 효과. **한국컴퓨터정보학회논문지**, 20(11), 191-197.

[9] 한순재 (2018). **PBL 기반 앱 프로그래밍 교육이 특성화 고등학교 학생들의 학습 태도 및 컴퓨팅 사고력에 미치는 영향**. 박사학위 논문, 한국교원대학교.

[10] 배학진 · 이은경 · 이영준 (2009). 문제 중심 학습을 적용한 스크래치 프로그래밍 교수 학습 모형. **컴퓨터교육학회 논문지**, 12(3), 11-22.

[11] 이철현 (2016). 소프트웨어 교육을 위한 컴퓨팅 사고력 기반 문제 해결 모형(CT-PS Model) 개발. **실과교육연구**, 22(3), 97-117.

[12] 전용주 (2017). **새로운 교육과정의 소프트웨어 교육을 위한 컴퓨팅 사고력 기반 창의적 문제해결(CT-CPS) 수업 모형의 개발 및 적용**. 박사학위 논문, 한국교원대학교.

[13] 최숙영 (2016). 문제해결의 관점에서 컴퓨팅 사고력 증

진을 위한 교수학습에 대한 연구. **컴퓨터교육학회 논문지**, 19(1), 53-62.

[14] 김영직 · 김성식 (2019b). 컴퓨팅 사고력 기반 테스트 중심 문제해결 학습 모형 연구. **컴퓨터교육학회 논문지**, 22(6), 43-55.

[15] 신정호 · 박상오 · 이규일 · 전우균 · 조건희 (2014). **TDD 이야기**. 한빛미디어.

[16] Kayongo, P. (2016). *Why do software developers practice test-driven development?*. University of Cape Town.

[17] Klegeris, A., McKeown, S. B., Hurren, H., Spielman, L. J., Stuart, M., & Bahniwal, M. (2017). Dynamics of undergraduate student generic problem-solving skills captured by a campus-wide study. *Higher Education*, 74(5), 877-896.

[18] Unruh, A., & Rosenbloom, P. S. (1989). Abstraction in problem solving and learning. *In Proceedings of the 11th international joint conference on Artificial intelligence-Volume* 1, 681-687.

[19] Román-González, M., Moreno-León, J., & Robles, G. (2017). Complementary tools for computational thinking assessment. *In Proceedings of International Conference on Computational Thinking Education* (CTE 2017), S. C Kong, J Sheldon, and K. Y Li (Eds.). The Education University of Hong Kong, 154-159.

[20] 최형신 · 김기범 (2015). 스크래치 프로그래밍이 예비교사에게 미치는 영향:컴퓨팅 사고 및 블룸의 텍사노미 활용 평가. **정보교육학회논문지**, 19(2), 225-232.

[21] Oluk.A., & Korkmaz.Ö. (2016). Comparing Students' Scratch Skills with Their Computational Thinking Skills in Terms of Different Variables. *International Journal Modern Education and Computer Science*. 8(11), 1-7.

[22] Zeba, Khanam & Mohammed, Najeeb, Ahsan. (2017). Evaluating the Effectiveness of Test Driven Development: Advantages and Pitfalls. *International Journal of Applied Engineering Research, 12*(18), 7705-7716.

[23] Fronza, I., Ioini, N. E., & Corral, L. (2017). Teaching computational thinking using agile software engineering methods: A framework for middle schools. *ACM Transactions on Computing Education (TOCE), 17*(4), 1-28.

김 영 직

1997년 한국교원대학교 컴퓨터교육과 (교육학학사)
2007년 고려대학교 컴퓨터정보통신대학원 소프트웨어공학(공학석사)
2004년~2016년 한국교육개발원 이러닝센터 근무
2014년~현재 한국교원대학교 컴퓨터교육학과 박사과정
2017년~현재 경인교육대학교 시간강사
관심분야: 소프트웨어교육, 소프트웨어공학, 이러닝
E-Mail: capbang93@gmail.com

김 성 식

1977년 고려대학교 경영학과(경영학사)
1978년~1991년 교육부 및 교육정책자문위원회 근무
1988년 미국오리곤주립대학교 대학원 컴퓨터과학과(이학석사)
1992년 고려대학교 컴퓨터학과(이학박사)
1992년~현재 한국교원대학교 컴퓨터교육과 교수
관심분야: 교육용 콘텐츠, 알고리즘, 원격교육
E-Mail: seongkim@knue.ac.kr