JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 21, No. 5, pp. 865-873, May 2020

Check for updates

컴퓨팅 사고력 기반 문제해결 중심의 교수학습 모델 개발 연구

김 효 숙[1] · 윤 보 현[2*]
[1]목원대학교 지능정보융합학과 박사과정
[2]목원대학교 소프트웨어교양학부 교수

A Study on Teaching-Learning Model Development for Problem Solving based on Computational Thinking

Hyo-Sook Kim[1] · Bo-Hyun Yun[2*]

[1]Ph. D Student, Department of Intelligent Information Convergence, Mokwon University, Daejeon 35349, Korea

[2]Professor, Department of College of Software Liberal Arts, Mokwon University, Daejeon 35349, Korea

[요 약]

현재 대학의 비전공자 대상 소프트웨어 교육을 살펴보면 교수자의 코드를 따라 하는 프로그래밍 위주의 교육이 이루어지고 있어 컴퓨팅 사고력 향상에는 어려움이 있다. 소프트웨어 교육은 컴퓨팅 파워를 이용한 문제해결 과정에서 사고하는 능력을 키우는 것이고, 이교육의 최종 산출물은 창의적 문제해결력이다. 이에 본 연구에서는 문제를 발견하고 분석한 후 구조화하여 해결할 수 있는 컴퓨팅 사고력 기반 문제해결 중심의 교수학습 모델을 제안하고 전문가 검토를 통해 모델의 타당도를 검증하였다. 모델의 검증 결과 긍정적인 평가 결과를 보였다 특히 'Copy'와 'Modify' 단계 구성 , 프로그래밍 학습에의 도움, CT 증진에의 도움에서 높은 평점을 보여 향후 학습에 적용 시 문제해결 중심의 컴퓨팅 사고력 향상에 긍정적인 효과가 있을 것으로 기대된다.

[Abstract]

Currently software education in college for non-major students are focused on programming, specifically by copying instructor's codes which ultimately stumbles students from gaining Computational Thinking skills. The purpose of software education is to gain thinking skills in problem-solving process by utilizing computing powers. Moreover, the very objective of this education is in creative problem-solving skills. Thus, this research proposes a Teaching-Learning Model focused on Problem-Solving based on Computing Thinking skills which involves problem identifying, analyzing, systemizing, and solving. Furthermore, the model has been reviewed by and has received its validity from respective professionals. The verification results of th model showed positive evaluation results. In particular, 'Copy' and 'Modify' stages configuration of the model, help in programming learning and improvement in Computational Thinking showed high ratings. It is expected that positive effects will show on Computational Thinking improvement based on Problem-Solving when applied in future learning.

색인어 : 컴퓨팅 사고력, 분해, 추상화, 패턴인식, 문제해결
Key word : Computational Thinking, Decomposition, Abstraction, Pattern Recognition, Problem-Solving

http://dx.doi.org/10.9728/dcs.2020.21.5.865

Received 26 March 2020; **Revised** 15 May 2020
Accepted 25 May 2020

***Corresponding Author; Bo-Hyun Yun**

Tel: +※ 개인정보 표시제한
E-mail: kkamti@mokwon.ac.kr

Ⅰ. 서 론

인공지능 기반의 지능 정보화 사회에 접어들면서 컴퓨팅은 문제해결의 가장 강력한 도구로 급부상 하고 있다. 이에 따라 컴퓨팅의 기본적인 개념과 원리를 기반으로 문제를 효율적으로 해결할 수 있는 사고능력, 즉 컴퓨팅 사고력을 기본 소양으로 요구하고 있다.

Seymour Papert에 의해 처음 소개된 컴퓨팅 사고력(Computational Thinking)은 카네기 멜론 대학교 컴퓨터 과학과 교수인 Jeannette M. Wing에 의해 학문 및 일상생활의 모든 문제에 널리 사용되어야 한다고 하였다[1][2][3]. Wing은 컴퓨팅 사고력은 컴퓨터 과학자뿐만이 아니라 누구나 배워서 활용할 수 있는 보편적인 사고라고 제시하고 있다. 따라서 읽기, 쓰기, 셈하기와 함께 모든 사람들이 기본적으로 갖춰야 하는 분석 역량 항목에 컴퓨팅 사고력을 추가해야 한다고 하였다[3][4].

컴퓨팅 사고력은 문제 해결을 위해 사고하는 과정 전반으로 정의될 수 있다. 그리고 이때 해결을 위한 도구로 컴퓨터를 이용하여 효과적으로 해결할 수 있는 능력이 필요한 것이다. 즉, 문제 해결을 위한 사고 과정을 이해하고 컴퓨팅 파워를 이용하여 문제해결력을 높이는 것이 컴퓨팅 사고력을 향상시키는 것이고 이것이 소프트웨어 교육을 하는 이유이며 목적이다[5].

그리하여 전 세계적으로 소프트웨어 교육을 실시하고 있고 우리나라에서도 2015 개정 교육과정을 통해 초· 중등에서 소프트웨어 교육을 강화하고 있다[6]. 대학 또한 교양 교육으로 비전공자 대상 소프트웨어 교육을 하여 채택· 확대해 가고 있다.

그러나 소프트웨어 교육이 문제 해결을 위한 사고 과정의 시간 없이 프로그래밍 언어 교육에만 맞춰지고 있는 문제점이 나타나 소프트웨어 교육을 '힘듦', '어려움', '모르겠음'으로 표현하거나[7], 소프트웨어 교육의 내용과 예제를 어렵게 생각하거나 자신의 전공과 연관성이 없다고 소프트웨어 교육에 대해 부정적으로 인식하였다[8].

이에 본 연구는 선행 연구를 통해 컴퓨팅 사고력의 핵심 요소들을 종합적으로 제시하고 그러한 요소들이 문제 해결 단계 학습 활동과 유기적으로 연계되어 어떻게 나타나는지를 분석하였다. 또한 비전공자 교양 SW 수업을 위한 컴퓨팅 사고력 기반 문제해결 중심의 교수학습 모델을 개발하였다.

Ⅱ. 관련 연구

2-1 컴퓨팅 사고력

Wing(2008)은 컴퓨팅 사고력을 2개의 A, 즉 추상화(Abstraction)와 자동화(Automation)로 구분하였고, 여기서 말하는 추상화는 문제 해결을 위해 사고하는 과정 전반을 의미하고, 자동화는 추상화 과정을 통해 만들어진 해결 모델을 컴퓨팅 시스템으로 시뮬레이션 실시하는 것을 말한다[9]. 또한 국제교육기술협회(ISTE)와 컴퓨터과학교사회(CSTA)에서는 컴퓨팅 사고력의 조작적 정의를 내리면서 하위 학습요소 9가지를 제시하였다[10].

Scratch를 개발한 MIT 미디어랩의 Brennan & Resnick은 Scratch관점에서 CT의 구성 요소를 인지적, 심동적, 정의적 측면의 3가지로 제시하였다[11]. 영국의 CS교육과정의 Key Stage3(ks3)에서는 CT의 구성 요소를 분해 , 패턴인식, 추상화, 알고리즘 4가지로 규정하였다[12]. Google(2016)에서도 CT의 구성요소를 분해, 패턴인식, 추상화, 알고리즘 설계로 구성하여 영국 KS 3의 CT 구성요소와 거의 동일하게 설정하였다[13].

우리나라 교육부와 한국교육학술정보원(KERIS)(2015)은 ISTE&CSTA(2011)의 개념을 인용하여 CT의 구성요소를 자료수집, 자료 분석, 구조화, 추상화(분해, 모델링, 알고리즘), 자동화(코딩, 시뮬레이션), 일반화로 설정하였다. 컴퓨팅 사고력은 컴퓨터 과학의 원리와 개념을 바탕으로 문제를 해결하는 지속적이고 체계적인 사고의 과정이며, 각기 다른 속성으로 구성된 요소들은 컴퓨터 과학적 문제해결과정에서 유기적이고 순환적으로 작용한다[14]. 대표적 선행 연구에서 나타낸 CT의 구성요소를 표 1에서와 같이 정리하였다.

표 1. 컴퓨팅 사고력 구성요소
Table 1. Computational Thinking Components

Researcher	Wing (2006)	ISET& CSTA(2011) (2016)	Brennan & Resnick (2012)	England Key Stage 3 (2016)	Google (2016)	Ministry of Education· KERIS (2015)
Computational Thinking Components	· Problem expression · Problem Override · Recursive thinking · Decomposition · Abstraction · Pattern Recognition · Modularization · Algorithm · Programming	· Data Collection · Data analysis · Data Representation · Problem Decomposition · Abstraction · Algorithms and Procedures · Automation · Simulation · Parallelization	· Concept · Training · Perspective	· Decomposition · Pattern Recognition · Abstraction · Algorithm	· Decomposition · Pattern Recognition · Abstraction · Algorithm Design	· Collecting data · Data analysis · Structured · Abstraction : · Decomposition, · Modeling, Algorithm· · ·Automatic : Coding, · Simulation · Generalization

2-2 SW 교육 교수학습 방법

SW 교육은 기술이나 기능의 습득이 아닌 컴퓨터를 이용한 문제해결 절차를 통한 컴퓨팅 사고력 함양에 중점을 둔 교육이다. 즉 창의력, 문제해결력 증진에 목표가 있고 이런 목표는 교육의 효과를 즉각적으로 확인하기 어렵다[15]. 이런 특성으로 비전공자 학습자들의 어려움은 다양한 형태로 나타나고 있다.

최정원과 이영준은 컴퓨팅 사고력의 요소 중 순차, 반복, 조건, 이벤트를 학습하는 과정에서 나타난 초보 학습자들의 오류를 분석하였다[16]. 김수환은 비전공자의 컴퓨팅 사고력 교육에서 학생들은 코딩 프로그램을 다루기 위해 기본적으로 알아야 하는 용어와 원리에 대해 가장 어려워하고, 다음으로 새로운 생각을 하고 구현하는 과정, 구현을 위한 명령어 사용에 대한 고민이 있다고 했다[7]. 오경선, 안성진은 비전공자를 위한 소프트웨어 교육에서 단계별 사고 과정과 코딩까지 모든 과정을 학생들이 어려워한다는 연구결과를 제시하였다[17].

대학에서 비전공자 대상 소프트웨어 교육은 프로그래밍에 대한 기본 지식이나 전공과목과의 연계성이 낮기 때문에 소프트웨어 교육의 효과와 다양한 학문 분야의 적용에 대한 정보 전달이 반드시 필요하고 이를 위한 적절한 교수학습 방법이 필요하다[18]. 그러나 박성희는 대학에서의 SW 교육에 관한 구체적인 교수학습 방법이 거의 없다고 언급하였다[19].

한국교육학술정보원은 SW 교육을 위해 표 2와 같이 5가지 교수학습 모델을 제안하고 있다[20].

표 2. 한국교육학술정보원 SW교육 모델
Table 2. KERIS SW Education Model

Education Model	Learning Procedure	Teaching Method
DMM	• Demonstration • Modeling • Making	Direct Instruction
UMC	• Use • Modify • reCreate	Discovery Learning
DDD	• Discovery • Design • Development	Inquiry Learning
NDIS	• Needs • Design • Implementation • Share	Project Based Learning
DPAA	• Decomposition • Pattern Recognition • Abstraction • Algorithm • Programming	Problem Based Learning

시연중심(DMM) 모델은 직접교수 모형을 바탕으로 프로그래밍 언어의 문법, 실습 중심의 명령어 등을 지도할 때 유용하다. 재구성 중심(UMC) 모델은 발견학습 모형에서 사례의 수정과 재구성을 이용한 학습 전략으로 놀이를 통한 수정 활동과 학습자의 능동적인 재구성 활동을 통해 컴퓨팅 사고를 이끄는 데 유용하다. 개발 중심(DDD) 모델은 소프트웨어공학적으로 소프트웨어 개발의 전 과정을 이해하는 모델로, 학습자가 개발의 과정을 주도하는 모델이다. 디자인 중심(NDIS) 모델은 디자인 사고 과정을 따르는 모델로 교사는 조력자 역할을 수행하며, 교사 학생간의 활발한 피드백을 통해 의미 있는 학습 결과를 도출할 수 있도록 하는 학생 중심의 프로젝트 기반 모델이다. CT 요소 중심 모델인 DPAA(P) 모델은 학습자 스스로 문제를 파악하고 분석을 실행하는 과정을 통해 문제해결역량을 배양하는 모델로 컴퓨팅 사고의 핵심적인 내용과 요소를 제시한 모델이다.

하지만, 표 2에서 제시한 SW교육 교수학습 모델 5가지는 2018년 이전까지 시행된 소프트웨어교육 운영지침[6]과 이를 기반으로 개발된 학생용 교재의 내용을 참고하여 교수학습모델을 개발하였다. 따라서 모든 수업 환경에 일반화하는데 한계를 가진다[20].

따라서 비전공자를 위한 소프트웨어 교육의 효과를 높이기 위해서는 표 2에서 제시한 SW 교육 모델을 바탕으로 초기상태의 문제에서 목표상태의 문제 해결에 이르게 하는 단계별 사고 과정을 끌어내어 컴퓨팅 파워를 이용한 다양한 학문 분야의 문제해결 과정을 경험하게 하여 컴퓨팅 사고력을 함양 하는 교수학습 방법이 필요하다.

Ⅲ. 제안하는 교수학습 모델

3-1 교수학습 과정

본 연구에서 제안하는 교수학습 모델은 스크래치 블록형 프로그래밍 언어 기반 CMTCSR 모델로 그림 1과 같이 3단계로 구성하였다.

이 모형은 한국교육학술정보원에서 제시한 SW교육 교수학습 모델 5가지 중 시연중심모델(DMM, Demonstration Modeling Making)과 CT 요소 중심모델인 DPAA(Decomposition, Pattern Recognition, Abstraction, Algorithm) 모델을 혼합한 모델로서 기존 5가지 모델에는 없는 학습 절차 Summary와 Review를 포함 시킨 CMTCSR 교수모델을 설계하였다.

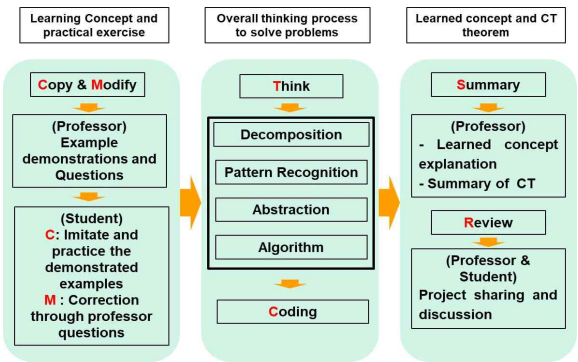

그림 1. CMTCSR 교수학습과정
Fig. 1. CMTCSR Teaching & Learning Course

제안하는 모델의 교수학습 과정은 그림 1과 같이 무작정 따라해보기(Copy) → 수정해보기(Modify) → 컴퓨터 과학자처럼 생각하기(Think) → 코딩하기(Coding) → 정리하기(Summary) → 점검하기(Review) 과정으로 구성하였다.

수업 초반 1단계 '무작정 따라해보기(Copy)'와 '수정해보기(Modify)' 에서 스크래치 각 명령 블록의 개념과 사용방법을 실습해본다. 교수자는 각 블록의 개념적 설명을 하는 것이 아니라, 학습자 스스로 블록을 탐색할 시간을 준다. 그런 후 교수자의 시연을 학습자는 그대로 따라 실습을 하며, 이런 과정에서 학습자가 자연스럽게 각 블록의 개념과 사용방법을 인지하도록 한다. '수정해보기(Modify)' 에서는 교수자가 제시한 도전과제를 수정해 본다. '무작정 따라해보기(Copy)'와 '수정해보기(Modify)'는 수업 초반에 반복적으로 이루어지는 활동으로 알고리즘을 작성하는 반복 훈련을 통해 컴퓨팅 사고를 기를 수 있다.

수업 중반 2단계 '컴퓨터 과학자처럼 생각하기(Think)' 에서는 '무작정 따라해보기(Copy)'와 '수정해보기(Modify)'에서 익힌 프로그래밍의 문법구조와 명령어를 능숙하고 효과적으로 사용하기 위해 팀별 협동학습으로 다양한 학문 분야의 문제에서 CT 요소를 찾고 표현하여 컴퓨팅 사고 과정이 능숙해지도록 실습해본다. '코딩하기(Coding)' 에서는 '컴퓨터 과학자처럼 생각하기(Think)' 에서의 산출물을 바탕으로 코딩을 한다.

수업 후반 3단계 '정리하기(Summary)' 에서는 이전 1, 2단계에서 실습으로 경험한 프로그래밍의 문법구조와 CT 요소에 대해 교수자 설명을 통해 개념화하는 단계이다. 이 단계가 표 2의 제시된 모델들과 다른 점으로 실습을 하면서 배운(Learning by practical exercise) 프로그래밍 개념과 원리를 학생 스스로 경험에 의해 지식을 개념화·구조화하도록 정리해 주는 단계이다. '검토하기(Review)'에서는 교수자가 완성된 스크립트의 일부를 은닉하여 제시하면 학습자는 은닉된 부분을 완성시켜보거나, 프로그램의 동작과정을 제시하여 프로젝트를 만들어 본다. 이 과정에서 완성된 산출물을 공유하고 산출과정과 제작 의도를 토론 할 수 있도록 한다.

3-2 CMTCSR 교수학습 과정을 적용한 문제해결 예시

비전공자 교양 SW 교육에 수학교육 전공과 연관성 있는 주제 '정다각형 그리기'와 '변수를 이용한 예각 및 둔각 삼각형 그리기 프로그램'을 적용한 예시를 표 3에서 학습 절차에 따라 학습 내용을 제시하였다.

1) Copy와 Modify (1단계)

스크래치 각 명령 블록의 개념과 사용방법을 실습해보기 위해 문제와 도전과제를 제시한다.

(1) 문제 제시

펜 블록과 반복 명령을 이용하여 정다각형 도형을 그려보자. 정다각형을 그리기 위해서는 내각과 외각의 성질을 알아야 한다. 외각을 이용한 정다각형 도형을 구현해보자. 더 나아가 원을 그리기 위해 움직임과 회전하기를 몇 번 반복해야 할까요? 별은 어떻게 그릴 수 있는가요?

표 3. CMTCSR 모형을 적용한 수학교육전공 다각형 그리기 프로그램 예시

Table 3. Polygon drawing program example of Mathematics Education Department using CMTCSR model

Learning Content	Learning Procedure	Teaching Method	Output Product
Drawing regular polygons using Pen Blocks and Repeat	Copy	Direct Instruction	Copy Script Code
	Modify		Modify Script Code
Drawing acute and obtuse triangles using Variable	Think	Problem Based Learning	Mind Map Table Picture Pictogram Diagram Natural Language Flow Chart Pseudocode
	Coding		Executable Script Code
Pen Blocks Repeat Exterior angle Interior angle Variable Acute angle Obtuse angle	Summary	Direct Instruction	
Blinding test	Review	Direct Instruction	Blinding Test Code Project

(2) Copy 단계 산출물

교수자의 설명과 시연에 따라 학습자는 그림 2와 그림 3의 Copy Script Code를 작성해본다.

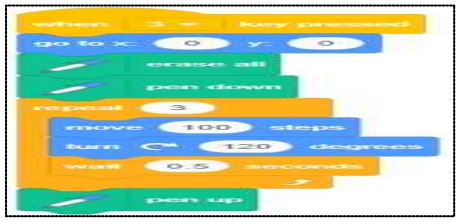

그림 2. 반복문을 이용한 삼각형 따라하기 예제

Fig. 2. Example of following a triangle using a loop

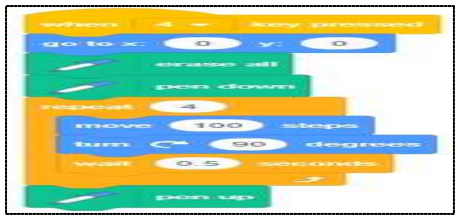

그림 3. 반복문을 이용한 사각형 따라하기 예제

Fig. 3. Example of following a square using a Repeat

(3) 도전과제 제시

도전과제 1 : 삼각형 • 사각형 스크립트를 수정하여 오각형, 육각형을 그리기 위해서는 어떤 블록을 수정해야 할까요?

도전과제 2 : 원(360도)을 그리기 위해서는 움직임과 회전하기를 몇 번 반복할까요?

도전과제3 : 별을 그리기 위해서는 무엇을 수정해야 할까요?

(4) Modify 단계 산출물

학습자는 도전과제에 따른 그림 4와 그림 5, 그림 6의 Modify Script Code를 스스로 작성해본다.

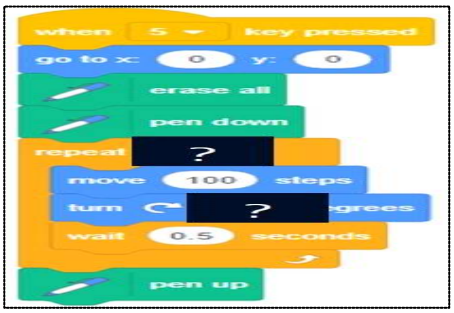

그림 4. 오각형 및 육각형 그리기 위한 블록 수정하기

Fig. 4. Modifying blocks for drawing pentagons and hexagons

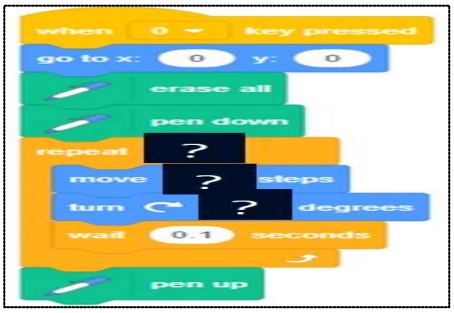

그림 5. 원 그리기 움직임과 회전하기 수정

Fig. 5. Modify circle drawing movement and rotation

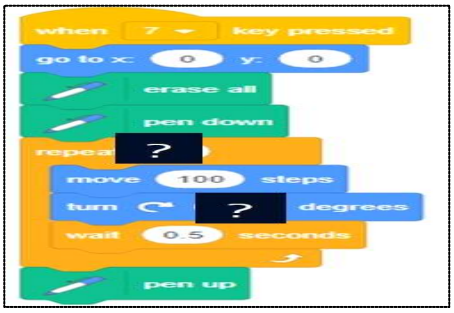

그림 6. 별 그리기 수정

Fig. 6. Star drawing correction

2) Think와 Coding (2단계)

팀별 협동학습을 통해 다양한 학문 분야의 문제에서 CT 요소를 찾고 산출물을 만들어 프로그래밍 하여 컴퓨팅 사고 과정에 능숙해지도록 실습한다.

(1) 문제 제시

삼각형을 각의 크기에 따라 분류했을 때, 0도 보다 크고 직각보다 작은 각을 예각이라 하고 세 각이 모두 예각인 삼각형을 예각삼각형이라고 한다. 크기가 직각보다 크고 180도 보다 작은 각을 둔각이라고 하고 한 각이 둔각이면 둔각삼각형이라고 한다. 변수를 이용하여 예각 및 둔각 삼각형 그리기 프로그램을 작성해 보자.

(2) Think 단계 산출물

CT 요소의 하나인 분해를 마인드 맵을 이용해 표현해 보거나, 분해된 작은 문제 사이의 유사성 또는 패턴을 표 4와 같이 표로 표현해 본다.

표 4. 표를 이용한 패턴 인식 표현

Table 4. Pattern recognition expression using table

Action of vertex Sprite	Operation according to conditions
Mouse Click	Broadcast (Save x, y coordinates) and wait
If the mouse click is 3	Broadcast (Draw a shap)
when I receive (Save x, y coordinates)	Store in vertex coordinate value variable

스프라이트 동작 과정에 따른 알고리즘을 그림 7과 그림 8, 그림 9와 같이 순서도로 표현해 본다.

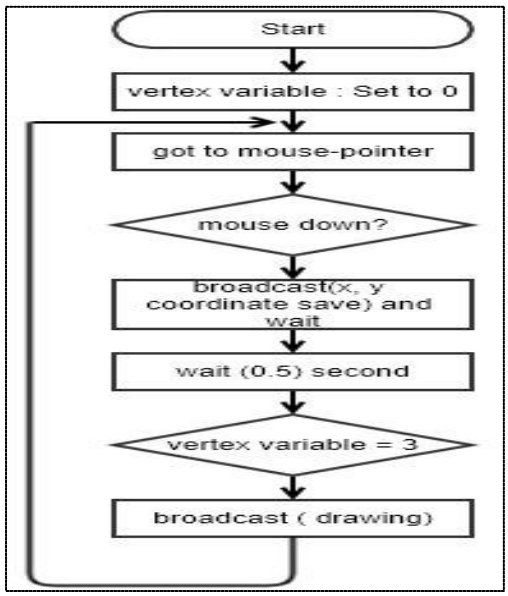

그림 7. 꼭지점 스프라이트 시작 시 순서도

Fig. 7. Flow Chart when start click of vertex Sprite

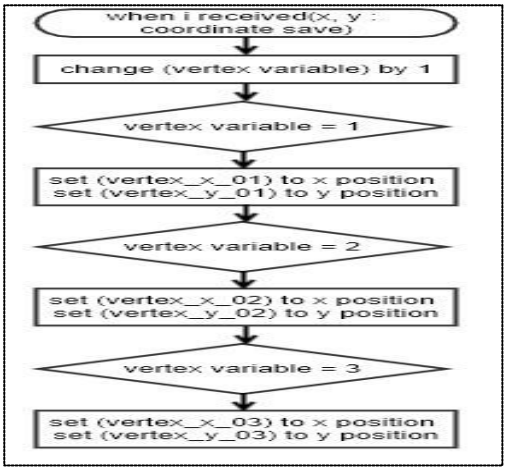

그림 8. 꼭지점 스프라이트의 신호 받았을 때 순서도
Fig. 8. Flow Chart of when I received broadcast of vertex Sprite

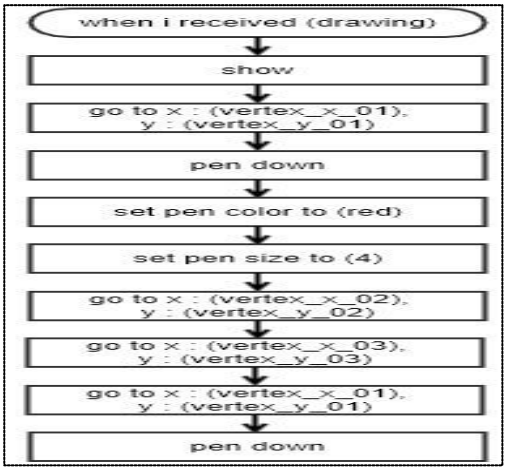

그림 9. 연필 스프라이트의 신호 받았을 때 순서도
Fig. 9. Flow Chart of when I received broadcast of pencil sprite

(3) Coding 단계 산출물

순서도를 보고 실행 가능한 스크립트 코드를 작성해 본다.

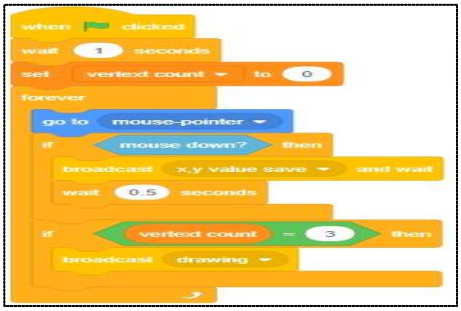

그림 10. 꼭지점 스프라이트 시작 시 실행 코드
Fig. 10. Executable Script Code when start click of vertex Sprite

그림 11. 꼭지점 스프라이트 신호 받았을 때 실행 코드
Fig. 11. Executable Script Code of when I received broadcast of vertex Sprite

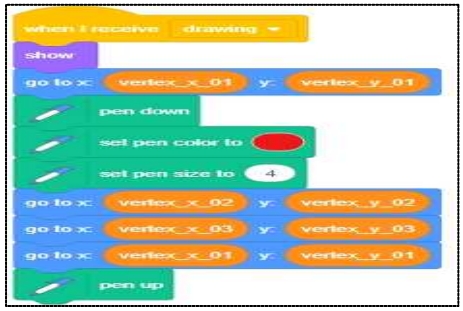

그림 12. 연필 스프라이트 신호 받았을 때 실행 코드
Fig. 12. Executable Script Code of when I received broadcast of pencil Sprite

3) Summary와 Review (3단계)

(1) Summary

1단계와 2단계 실습을 하면서 배운(Learning by practical exercise) 프로그래밍의 문법구조 반복문과 선택문, 변수에 대 교수자 설명을 통해 지식을 개념화한다.

(2) Review 단계의 Blinding Test Code

'검토하기'에서 그림 13의 프로젝트 실행 결과와 일부 은닉된 스크립트 그림 14을 제시하면 학습자는 은닉된 부분을 완성한다.

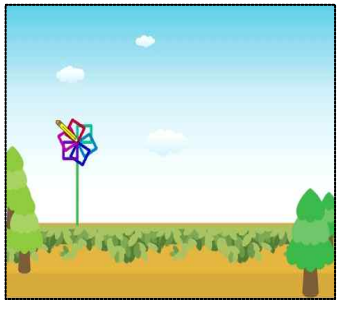

그림 13. 실행 결과
Fig. 13. Result of execution

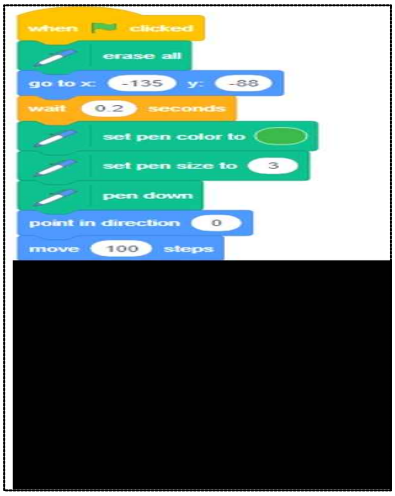

그림 14. 일부 은닉된 테스트 코드
Fig. 14. Blinding Test Code

표 5. CMTCSR 모형 타당도 조사 항목 결과
Table 5. CMTCSR Model Validity Survey Item Result

Number	Validity survey item	M	CVR
1	CMTCSR model learning stage composition is appropriate	4.5	1.00
2	'Copy' and 'Modify' stage composition and process order appropriateness	4.7	1.00
3	Subcomponents of the 'Think' stage and the progression sequence of 'Coding' are appropriate	4.4	1.00
4	'Summary' and 'Review' stage composition and process order appropriateness	4.5	1.00
5	Adequacy of CMTCSR Model Step-by-step Output	4.5	1.00
6	Adequacy of CT connections by CMTCSR model step-by-step	4.3	1.00
7	Adequacy of programming learning in CMTCSR model	4.6	0.80
8	Appropriateness of CT improvement in CMTCSR model	4.6	1.00
Mean		4.5	

Ⅳ. CMTCSR 교수학습 모형 검증

본 모형의 학습 단계 구성, 각 단계별 구성과 진행순서, CT 증진의 도움 등이 타당한지를 검증하기 위하여 타당도 조사를 실시하였다. 타당도 조사에 참여한 교육전문가의 교육 경력은 20년 이상 (1명), 15년 이상(1명), 10년 이상(6명), 5년 이상(2명)이다. 학력은 박사(4명), 석사(4명), 학사(2명)이다.

이들 10인의 컴퓨터교육 분야 전문가를 대상으로 표 5와 같이 CMTCSR 모형의 8가지 항목에 대한 타당도를 조사하였고, 응답은 리커드 5점 척도 '매우 그렇다'(5점), '그렇다'(4점), '보통이다'(3점), '그렇지 않다'(2점), '매우 그렇지 않다'(1점)로 평가하였다.

타당도 분석 방법은 Lawshe(1975)의 내용타당도 비율(CVR ; Content Validity Ratio)을 이용하였다. 응답 전문가 수 10인에 따른 CVR 최솟값은 .62이다.

평가 결과 전체 문항 평점은 4.5로 긍정적인 것으로 나타났다. 특히 'Copy'와 'Modify' 단계 구성과 진행 순서의 적절성(4.7), 프로그래밍 학습에 도움(4.6), CT 향상에 도움(4.6)에서는 높은 평가 결과가 나타났다. 이는 모형에서 제공된 예제를 통하여 'Copy'와 'Modify' 단계에서 문제를 해결하는 과정이 프로그래밍 학습과 컴퓨팅 사고력 향상에 긍정적인 효과가 있을 것으로 해석 할 수 있다.

반면, 'Think'의 하위 구성요소와 'Coding' 단계의 진행 순서 적절성(4.4), CT 연결의 적절성(4.3)에서는 전체 문항 평균보다 낮은 결과가 나타났는데, 이것은 다양한 학문 분야의 문제에서 CT 구성 요소를 찾아내는 연습이 익숙하지 않은 상황에서 CT 구성 요소와의 연관성에 의문점을 갖는 것으로 추측된다. 따라서 'Think' 이전 단계인 'Copy'와 'Modify'를 통해 CT 구성 요소에 익숙해지도록 충분한 예제를 제공할 필요가 있을 것으로 생각된다.

Ⅴ. 결 론

본 연구에서는 프로그래밍에 대한 기본 지식이 없고 전공과목과의 연계성이 낮은 비전공자들의 소프트웨어 교육을 위해 CMTCSR 교수학습 절차에 따른 프로그래밍 실습 과정과 컴퓨팅 사고의 과정을 실습하는 교수학습 모델을 제안하였다.

제안하는 교수학습 모델은 시연중심모델(DMM)과 CT 요소 중심모델인 DPAA(P) 모델을 혼합한 모델이다.

'무작정 따라해보기(Copy)'와 '수정해보기(Modify)' 단계에서 알고리즘 작성의 반복적인 활동으로 학생들로 하여금 문제해결과 프로그래밍에 자신감을 갖게 할 수 있도록 하였다.

'컴퓨터 과학자처럼 생각하기(Think)'와 '코딩하기(Coding)' 단계에서는 다양한 학문 분야의 문제를 통해 컴퓨팅 구성 요소를 찾아내고 표현해보며 컴퓨팅 사고가 능숙해지도록 실습하여 문제해결력을 배양 할 수 있도록 하였다.

이런 단계를 통해 익힌 프로그램의 문법구조와 알고리즘, CT 요소에 대해 '정리하기(Summary)'와 '검토하기(Review)' 단계에서는 경험에 의해 지식을 개념화·구조화하도록 하였다.

본 연구의 초점은 교수자의 시범을 따라만 하는 프로그래밍 기능 습득과 기능 구현에 머무를 수 있는 비전공자 대상 SW 교육에 프로그래밍 학습과 CT 역량을 연결시켜 봄으로써 컴퓨팅 사고력 기반 문제해결 중심의 교수학습 방안을 제시하였다는 것이다.

향후 본 연구에서 제안하는 CMTCSR 교수학습 모델을 사립대학 비전공자 신입생 교양 소프트웨어 교육에 적용하여 실증적 검증 및 정교화 작업을 거쳐 컴퓨팅 사고력 기반 문제해결 역량이 이루어지는지를 분석하고자 한다. 그러나 제안한 모델의 타당도 검증에서 'Think'의 하위 구성요소, 'Coding' 단계의 진행 순서와 CT 연결의 적절성 문항은 전체 문항 평균보다 낮은 결과가 도출되었기에 후속 연구에서는 CT 구성 요소에 익

숙해질 수 있는 방안을 모색하고, 이를 바탕으로 학습자의 전공 분포도와 프로그래밍 경험 여부 등의 학습자 분석과 평가도구 개발을 통해 효과성을 입증하고자 한다.

참고문헌

[1] Papert, S. , Resnick, M, "Technological Fluency and the Representation of Knowledge", Proposal to the National Science Foundation. MIT Media Laboratory. 1995

[2] Wing, J. M, "Computational Thinking", *Communications of the ACM.* Vol. 49, Issue 3, 19-3. pp.33-35, March 2006

[3] H. S. Choi, I. K. Jeong, H. J. So, "Computational Thinking Framework-based Analysis of After school Scratch Team Project Experiences", *Journal of The Korean Association of Information Education*, Vol. 18, No. 4, pp. 549-558, 2014.

[4] H. S. Choi. "Developing Lessons and Rubrics to Promote Computational Thinking", *Journal of The Korean Association of Information Education*, Vol. 18, No. 1, pp. 57-64, 2014.

[5] S. Y Choi, "A Study on Teaching-learning for Enhancing Computational Thinking Skill in terms of Problem Solving", *The Korean Association Computer Education,* Vol. 9, No. 1, Jan, 2016.

[6] Software Education Operation Guidelines, Ministry of Education, 2015

[7] S. W. Kim, " Analysis of Non-Computer Majors Difficulties in Computational Thinking Education", *The Korean Association Computer Education*, Vol. 19, No. 3, May 2015.

[8] W. S. Kim , "A Study on the Recognition of Freshman on computational Thinking as Essentral Course", *Culture and Convergence*, Vol. 39, No. 6, December 2017.

[9] Wing, J. M. "Computational Thinking and Thinking about computing", *Philosophical Transactions of Royal Society A*, Vol. 366, pp. 3717-3725, July 2008.

[10] ISTE & CSTA, Computational Thinking Leadership Toolkit 1st edition.

[11] Brennan, K. & Resnick, M., "New frameworks for studying and assessing the development of computational thinking", Proceedings of the 2012 annual meeting of the American Educational Research Association, pp. 1-25, 2012.

[12] BBC Bitesize(2016). Introduction to computational thinking. Retrieved from [Internet]. Available: http://www.bbc.co.uk/education/topics/z7tp34j

[13] Google's CT. [Internet]. Available: http://goo.gl/oG0w56

[14] C. H. Lee, " Development of Computational Thinking based Problem Solving Model(CT-PS Model) for Software Education," *The Journal of Korean Practical Arts Education*, Vol. 22, No. 3, pp. 97-117, 2016.

[15] G. J. Park, Y. J. Choi, "Exploratory study on the direction of software education for the non-major undergraduate students", *Journal of Education & Culture*, 2018, Vol. 24, No. 4, pp. 273-292, 2018

[16] J. W. Choi, Y. J. Lee, "The analysis of Learners` difficulties in programming Learning," *The Journal of Korean Association of Computer Education*, Vol. 17, No. 5, pp. 89–98, 2014.

[17] K. S. Oh, S. J. Ahn, "A study on the relationship between difficulty in learning to program and Computational Thinking," *The Journal of Korean Association of Computer Education*, Vol. 18, No. 5, pp. 55 –62, 2015.

[18] H. W. Jung, "A Study on basic software education applying a step-by-step blinded programming practice," *Journal of Digital Convergence,* Vol. 17, No. 3, pp. 25-33, 2019.

[19] S. H. Park, "Study of SW education in university to enhance computational thinking", *Journal of Digital Convergence*, Vol. 14, No. 4, pp 1-10, 2016.

[20] J. Kim et al. 2015 Education policy network training on-site support research : Development of SW education Teaching and Learning Models, Commissioned research CR 2015-35, 2016.

김효숙(Hyo-Sook Kim)

2001년 : 목원대학교 컴퓨터교육과 (공학사)
2007년 : 목원대학교 교육대학원 컴퓨터교육과 (교육학석사)

2008년~현 재: 목원대학교 강사
2019년~현 재 : 유원대학교 강사
2018년~현 재 : 목원대학교 일반대학원 지능정보융합 박사과정
※관심분야 : 소프트웨어교육, 빅데이터분석, 텍스트마이닝, 인공지능

윤보현(Bo-Hyun Yun)

1995년 : 고려대학교 컴퓨터학과 이학석사
1999년 : 고려대학교 컴퓨터학과 이학박사

1999년 ~ 2003년 : 한국전자통신연구원 지식처리팀 선임연구원 및 팀장
2003년 ~ 현 재 : 목원대학교 소프트웨어교양학부 교수
※관심분야 : 소프트웨어교육, 자연어처리, 빅데이터분석, 지식마이닝, 정보검색