Journal of The Korea Society of Computer and Information
Vol. 24 No. 9, pp. 171-178, September 2019

www.ksci.re.kr
https://doi.org/10.9708/jksci.2019.24.09.171

Education Course Model based on AP CSP For Improvement of Computational Thinking

EunYoung Cheon*

Abstract

Computational Thinking is one of the biggest issues in the era of the Fourth Industrial Revolution. It is a core literacy required not only for SW major but also for all students including them. It is not a simple computer software education, but a coding education based on Computational Thinking, and it should be able to solve the problems in everyday life and to express the process and solutions. However, in the case of students who lack background knowledge on SW and programming languages for development, it is hard to know how to algorithmize problems and express them using computer devices. In this study, we proposed a education course model to improve the students' thinking skills and to express them effectively. In addition, we confirmed whether the non-major students who learned through this education course model can express various problems related to the major field by integrating them with computing accidents and improve the problem solving ability.

▸Keyword: Collaboration, AP CSP, Computational Thingking, SW Education, Coding Education

I. Introduction

정보화 혁명 시대에서 소프트웨어 교육은 컴퓨터관련 분야를 전공하는 학생들의 범주로 한정되어 왔으며 이시기의 비전공자를 대상으로 한 SW교육은 일명 정보화 교육으로 일컬어졌다. 시대적 패러다임의 변화로 2016년 '4차 산업혁명' 시대가 도래 했으며 빅데이터, 사물인터넷, 인공지능, 가상환경 등을 일상으로 누구나 접하고 경험할 수 있게 되었다. 다시 말하면 기술이 가지고 있는 개별적 학문 요소들이 아닌 문화, 예술, IT, 기술 등의 다양한 학문과 전문영역간의 요소들을 융합하여 새로운 가치창출을 이뤄낼 수 있게 된 것이다[1].

따라서 이러한 4차 산업 혁명시대에서는 누구나 자신의 생각을 자유롭게 인터넷, 모바일 등의 다양한 매체를 통하여 의사를 전달 할 수 있고, 표현하는 방법을 학습할 필요가 있다. 4차 산업혁명 시대의 핵심 화두 중 하나인 'Computational Thinking' 즉 컴퓨팅사고는 컴퓨터 프로그래밍이 가지는 문법이나 표현방법 보다는 소프트웨어를 이용하여 현실세계에서 접할 수 있는 여러 가지의 문제를 어떻게 해결할 수 있는지 그 방법을 익히고 자신의 사고를 표현할 수 있는 것을 말한다[2].

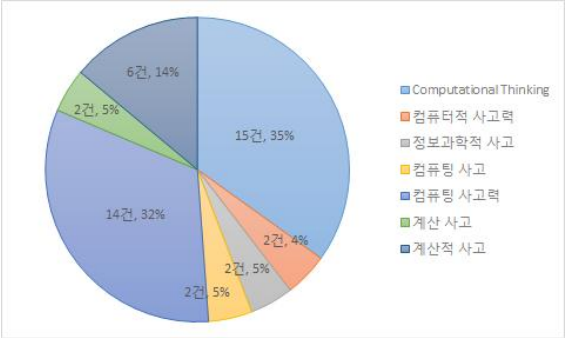

Fig. 1. A Case Study of using Translated Computational Thinking in 43 Domestic Papers

현재 컴퓨팅프로그래밍을 학습할 수 있는 이공계열 학생뿐만 아니라, 인문계열 및 예체능 계열 등 대학의 전 학과 학생들에게 코딩교육을 강의할 수 있는 다양한 매체와 프로그램들이 개발되어 출시되고 있고[3], 또한 국내에서도 여러 대학에서 교양교육 과정으로 이러한 코딩교육들을 실시하고 있는 추세이

• First Author: EunYoung Cheon, Corresponding Author: EunYoung Cheon
*EunYoung Cheon (eycheon@cnu.ac.kr), Dept. of Computer Science & Engineering Chungnam National University
• Received: 2019. 08. 14, Revised: 2019. 09. 18, Accepted: 2019. 09. 18.

며[4-7], 그림 1과 같이 컴퓨팅사고에 관련한 논문 및 연구의 발표가 활발히 진행되고 있다[3].

컴퓨팅사고는 4차 산업혁명이 도래한 현 시기에 대학의 SW 전공자뿐만 아니라 이들을 포함한 모든 학생들에게 요구되는 핵심적인 소양이다. 단순한 컴퓨터 소프트웨어 교육을 실행하는 것이 아닌 컴퓨팅사고를 기반으로 한 코딩교육을 실시하고, 이것을 이용하여 일상생활에서 발생하는 문제들을 해결하고 그 과정과 해법들을 표현할 수 있어야 한다.

따라서 본 연구에서는 비전공계열 학부생들을 대상으로 컴퓨팅사고를 기반으로 한 코딩 교육과정 모델을 제안한다. 또한 이 교육과정 모델을 통하여 학습한 비전공 학생들이 전공영역과 관련된 여러 가지 문제들을 컴퓨팅사고와 융합하여 표현할 수 있는지 여부 및 문제해결 향상력을 확인하는데 그 목적을 둔다.

이 논문의 구성은 다음과 같다. 2장에서는 컴퓨팅사고에 대한 정의에 대하여 설명하고, 국/내외 교육현황을 기술한다. 3장에서는 컴퓨팅사고를 기반으로 한 교육과정 모델을 제안하고 4장에서는 제안한 교육과정 모델의 운영 결과에 대해 설명하고, 효율성을 분석한다. 마지막으로 5장에서는 결론을 기술한다.

II. Related works

1. Computational Thinking

학자	Computational Thinking 정의
David Moursund	Saymour Papert가 그의 저서 "Mindstorms"에서 언급하였던 절차적 사고와 연관되어 있는 개념이다. 절차적 사고는 개발(developing) - 표현(representing) - 시험(testing) - 디버깅(debugging)의 순서들로 구성되어 있고, 효과적인 절차를 만들기 위해서는 컴퓨터 등의 특별한 장치로 수행 가능한 형태인 구체적인 단계별 명령이 필요하다.
Peter Lee	CT는 인간 지능을 확대하여 실제적인 적용을 할 수 있는 인간지능의 메커니즘에 관한 연구이다. 다시 말해 인간의 정신적인능력의 복잡도를 관리하거나 일을 자동적으로 처리하도록 하는 추상화 도구를 통해 확장하는 것이라 할 수 있다.
Bill Wulf	과학은 물리적인 대상(physical object)에 관련된 것이고, CT는 어떠한 문제를 해결하는 과정과 그 과정의 진행을 가능하게 하는 추상적인 현상들에 초점을 맞추고 있다.
Don Abrahamson	CT는 형식지를 설명하고, 암묵지를 객관화하고, 이러한 지식을 컴퓨팅적인 형태로 퍼뜨리고, 이러한 활동에서 발생한 결과물을 관리하는 컴퓨터 관련 기호 체계의 사용이다.
Gerald Sussman	CT는 일을 처리하는 정확한 방법을 공식화하는 방법이다. 다시 말해 특정한 문제를 효율적으로 처리하기 위해 그 문제를 철저하게 분석하고 해결하기 위한 엄밀한 절차를 만드는 과정이 CT인 것이다.
Edward Fox	문제 해결을 목적으로 눈에 보이지 않는 추상화된 개념을 다루거나 조작하는 것이 CT의 핵심이다. CT는 "인간이 세상에 접근하고, 과정들을 생각하고, 디지털로 표현된 것들을 다룰 때 하는 일들" 이라고 표현할 수 있다.
Robert Constable	CT는 특정한 기술이나 사고 과정들의 집합이 아니라 개방적이고 점점 발전하는 기술의 역동적인 본질에 관련된 개념이다. 현재까지 나타난 CT를 대략적으로 기술한다면 "지적인 과정을 자동화하는 것" 과 "정보 처리 과정을 연구하는 것" 정도가 될 것이다. CT가 특별하게 받아들여지는 이유는 우리가 생각하는 "컴퓨팅 사고력" 이 컴퓨터라는 도구를 통해 실행될 수 있다는 점이고, 이러한 점은 "컴퓨터가 파트너 및 동지가 될 수 있다." 는 발견을 가져왔다는 것이다.

Fig. 2. Scholars Definition of Computational Thinking

Wing은 컴퓨팅사고는 21세기를 살아가는 모든 사람에게 기본적으로 필요한 기술이며 문제 해결, 시스템 설계, 인간 행동의 이해를 위한 사고로 정의하였다. 또한 이것을 재귀적 사고, 추상적 사고, 선행적 사고, 절차적 사고, 논리적 사고, 동시적 사고, 분석적 사고, 전략적 사고 등 8가지의 항목으로 분류하였다[2][8].

학자들은 2009년, 2010년 두 차례의 워크숍을 워싱턴에서 개최하고 컴퓨팅사고의 정의에 대하여 논의했는데 이들의 정의는 학자마다 다르며 그림2[9] 와 같다.

컴퓨팅사고는 일상생활과 현실세계에서 직면하는 수많은 문제들을 어떻게 해결할 것인지 절차적으로 설계하고 이것을 컴퓨팅 기기를 이용하여 문제해결 방법을 표현할 수 있는 융복합적인 사고의 표현 방식이며, 4차 산업혁명 시대를 살아가는 미래사회의 인재들이 갖춰야 하는 필수 요소이자 기술인 것이다.

2. Domestic and International Computational Thinking Education Status

2.1. Estonia

에스토니아는 2012년 Tiger Leap Foundation이라는 비영리재단에서 ProgeTiger(Programming Tiger)프로젝트를 시작으로 2013년 HITSA라고 불리는 정보교육연구기관에서 소프트웨어 교육과정을 주도하고 있다[10]. 에스토니아의 SW교육 목적은 단순한 프로그래머의 양성과정이 아닌 우리 주변의 현상을 문제들을 파악하여 이해하고 최신의 기술을 접목하여 이 문제들을 해결할 수 있는 방안을 제시하고 있으며, 학습자의 특성과 흥미에 맞는 단계별 코딩교육 과정의 학습단계를 수행하고 있다.

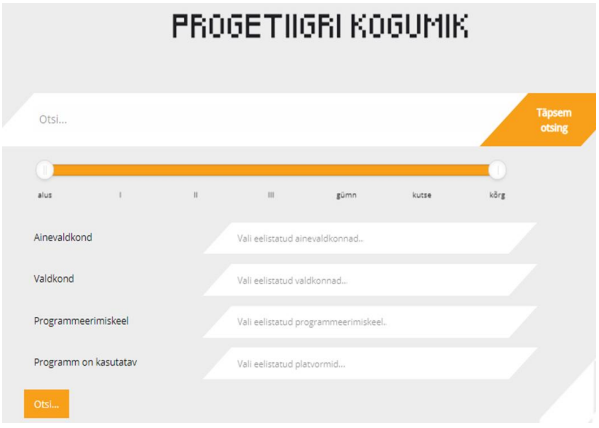

Fig. 3. ProgeTiiger Online Training Tool for Teachers

뿐만 아니라, 그림3[11]과 같이 ProgeTiger programme - #HITSA 사이트의 운영을 통하여 학습자 및 교수자의 흥미영역과 프로그래밍언어의 정도 및 사용수준을 입력하면 이에 맞는 프로그래밍, 로봇 공학, 3D 디자인, 스마트 스팟 및 멀티미디어 영역 등의 코딩용SW를 검색하고 결과를 제공하여 사용할 수 있도록 안내하고 있으며, 해당 SW에 관한 동영상 자습서도 제공하고 있다[12].

2.2 Finland

핀란드는 소프트웨어 교육을 위한 코드클럽(koodikerho)을 구성하여 2016년부터 본격적으로 코딩교육을 필수 교육과정으로 실시하기로 결정했다[13][14]. 또한 프로그래밍을 학습하는 목적은 산업분야에서부터 건강, 예술분야 등 거의 모든 분야에서 자신의 생각을 표현하기 위한 도구로 활용하기 위함으로 밝히고 있으며[15], 표 1[16]과 같이 코딩클래스 과정을 운영하고 있다.

Table 1. Suitable Resources for Coding class[16]

Grades	Programming Resources
1-2 Grade	CS Unplugged, Robottileikki, Computer Science-a-Box:Unplug Your Curriculum, Computer Science for Fun
3-6 Grade	Scratch, Kodu, Alice, App inventor, Logo, Turtle Roy
7-9 Grade	Khan Academy, W3 Schools, Udacity, Codecademy, Code Schol, EDX, Helsingin yliopiston MOOC, MIT OpenCourseWare, Processing, Coursera, Code Avengers, Code Combat, ruby warrior, Dash, Kinestethic Learning Activities

2.3 United States

미국 CSTA는 2011년부터 K-12(Computer Science Standards)를 제시하여 저학년부터 고등교육 전 학년에 컴퓨터 과학교육을 필수로 지정하였다[17]. 컴퓨팅사고를 목표로 학생들의 협업을 통하여 Community global and ethical impacts, Compute practice & programming, Computers and communication devices 등을 수행할 수 있도록 교육한다. 이 과정에서 다음 과 같은 5가지의 Core Discipline을 목표로 가지며, 교육과정은 그림4 와 같다[18].

- Intellectually Important
- Lead to multiple career paths
- Problem solving
- Support & Link other science
- Engage all students

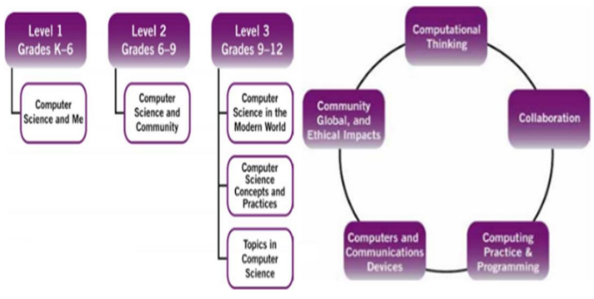

Fig. 4. Organizing Structure for the Computer Science Standards &
Strands in the Computer Science Standards

2.4 South Korea

우리나라에서는 2015년 개정교육과정에서 SW교육을 운영하기로 결정하였고, 소프트웨어 연구•선도학교를 60개(2015년) → 900개(2016년) → 1,200개(2017년) → 1,641개(2018년)로 단계별로 선정하여 SW교육을 필수로 실시하기로 하였다[19][20]. 또한 SW 인재양성을 위한 추진과제의 일환으로 SW 중심대학을 선정하고 체계적인 SW교육 및 인재양성을 위한 저변을 갖추고 있는 추세이다. SW관련 학과들도 기존의 교육 체계를 넘어서 학생들의 프로그래밍 교육들을 강화하여 SW 관련 취업준비를 할 수 있도록 전문성을 갖춘 교육과정들을 준비하고 있다[21].

3. Necessity of research to improve Computational thinking

[22]-[24]들은 소프트웨어교육을 통하여 긍정적인측면과 효과를 확인하고, 코딩교육을 통하여 문제해결능력을 향상할 수 있음을 시사하는 연구들이다. [5][7][25]에서는 컴퓨팅사고력 향상을 위하여 본인의 전공과 소프트웨어의 융합에 대한 학생들의 니즈가 높다는 것 뿐 만아니라 SW교육도구들을 활용하여 문제해결능력을 향상코자 하는 의지가 있음을 보여주는 연구이다.

앞 절의 국내외 코딩교육 현황 사례들처럼 소프트웨어를 이용한 코딩 교육은 단순히 SW인력 양성을 목적을 위해 필요한 것이 아니라, 4차 산업혁명 시대의 미래컴퓨터와 인간이 공존하기 위함이며 또한 학습자들의 분석 및 문제해결능력을 키우고, 창의력과 다양한 분야의 융합적 사고능력 증진을 위하여 반드시 필요한 교육이다. 즉, SW 교육의 도입은 SW중심사회를 통한 성장 동력을 마련하기 위한 발판이 될 수 있음을 뜻하는 것이다.

최근 몇 년 사이 컴퓨터프로그래밍을 학습하지 않은 비이공계 학생들에 대한 컴퓨팅사고력향상을 위한 연구와 문제해결능력을 증진할 수 있는 수요 및 교육방법들이 최근 몇 년 사이에 적극적으로 늘어나고 있으나, 이에 비하여 아직은 이들에게 SW코딩교육을 도입하지 않은 대학들이 더 많은 상황이며 체계적인 기준 및 준거를 가진 교육과정 및 교수학습방법들이 더 필요한 상황이다.

III. The Proposed Research and Operation

이 연구에서는 컴퓨팅프로그래밍 과정에 대한 학습의 경험이 없는 비이공계열의 학생들을 대상으로 컴퓨팅사고를 기반으로 문제해결능력의 향상 및 표현이 가능한 교육과정 모델을 제안한다. 비이공계 학생들이 소속되어 있는 전공 관련분야 또는 일상생활에서 관심 있는 분야에 대한 문제를 선정하고, 이 문제들을 해결할 수 있는 방안을 모색하여 알고리즘으로 표현할 수 있도록 수업내용을 구성한다.

수업과정을 구성하기 위한 프레임워크는 고등교육과정 수준의 준거를 갖기 위해 AP CSP(Advanced Placement Computer Science Principle)를 참조하여 구성한다. AP CSP 커리큘럼 프레임워크는 Computational Thinking Practices와 Big Ideas의 7가

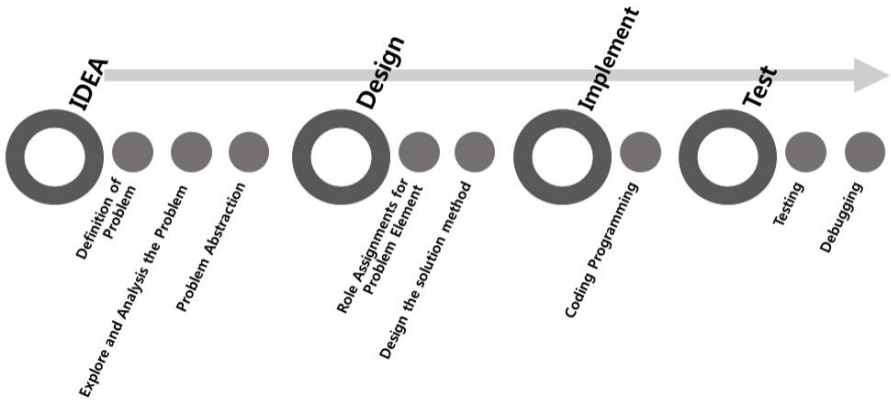

Fig. 6. Software Production Process

지 Concept Outline 으로 표2와 같이 구성되며[26][27], 평가 가능한 교과전공분야들은 그림 5와 같다[28]. Computational Thinking Practices의 P1~P7은 P1~P7의 항목으로 표현하며, Big Ideas의 7가지 요소는 B.I.1~B.I.7으로 줄여 표현한다.

Table 2. Element of AP CSP

Computational Thinking Practices	P.1	Connecting Computing
	P.2	Creating Computational Artifacts
	P.3	Abstracting
	P.4	Analyzing Problems and Artifacts
	P.5	Communicating
	P.6	Collaborating
Concept Outline	Big Idea 1	Creativity(B.I.1)
	Big Idea 2	Abstraction(B.I.2)
	Big Idea 3	Data and Information(B.I.3)
	Big Idea 4	Algorithms(B.I.4)
	Big Idea 5	Programming(B.I.5)
	Big Idea 6	The Internet(B.I.6)
	Big Idea 7	Global Impact(B.I.7)

- Art History
- Biology
- Calculus AB
- Calculus BC
- Chemistry
- Chinese Language and Culture
- Comparative Government and Politics
- Computer Science A
- Computer Science Principles
- English Language and Composition
- English Literature and Composition
- Environmental Science
- European History
- French Language and Culture
- German Language and Culture
- Human Geography
- Italian Language and Culture
- Japanese Language and Culture
- Latin
- Macroeconomics
- Microeconomics
- Music Theory
- Physics 1: Algebra-Based
- Physics 2: Algebra-Based
- Physics C: Electricity and Magnetism
- Physics C: Mechanics
- Psychology
- Research
- Seminar
- Spanish Language and Culture
- Spanish Literature and Culture
- Statistics
- Studio Art: 2-D Design
- Studio Art: 3-D Design
- Studio Art: Drawing
- United States Government and Politics
- United States History
- World History

Fig. 5. AP Courses and Exams in AP CSP

그림 6은 컴퓨팅사고를 바탕으로 문제해결능력을 향상할 수 있도록 SW을 제작 프로세스를 정의한 것 이다. 이를 설명하면 다음 과정과 같다.

① IDEA - 문제를 정의하고, 탐색 및 분석을 통하여 추상화 시킨다.

② Design - 추상화시킨 문제의 구체적인 해결과정을 수립하기 위하여, 문제 내부의 문제해결을 수행하기 위한 단위와 역할을 구분 짓고, 이들을 코딩SW를 이용하여 디자인 한다.

③ Implement - ②에서 구분한 단위 및 역할들이 구동될 수 있도록 알고리즘화 하고 이들을 프로그래밍 한다.

④ Test - 완성된 SW의 테스트를 수행하고 디버깅한다.

문제를 탐색하고 이 문제들을 해결하기 위해 알고리즘으로 구성할 수 있어야 하며, 이것을 교육용 블록프로그래밍을 이용하여 표현 가능도록, AP CSP 프레임과 그림 6의 프로세스를 기반으로 한 교육과정 모델을 표4와 같이 제안하며 이것을 설명하면 다음과 같다.

• Week 1 에는 컴퓨팅사고란 무엇인지를 학습하고 소프트웨어를 배워야 하는 이유와 코딩의 개념을 이해한다.

• Week 2~4까지는 전공분야에 관련된 문제들을 탐색하기 위하여 문제의 표현방법을 학습한다. 이 과정에서 데이터 및 정보의 표현방법을 학습하고 문제의 정의 및 표현방법 그리고 이것들을 추상화하고 단계별로 나누어서 분석하는 방법을 이해한다.

• Week 5~7 에서는 분석한 문제의 유닛들을 알고리즘을 표현하기 위한 과정들을 학습한다. 순차, 선택, 반복 및 정렬과 탐색 알고리즘의 개념들을 이해하고 이를 코딩으로 표현할 수 있도록 유도한다.

• Week 8~12에는 교육용 블록 프로그램들의 종류에 대해 알아본다. 이 중 스크래치 프로그램의 사용방법을 숙지하고 스프라이트와 명령블록들을 이용하여 자신의 생각과 사고들을 컴퓨터를 이용하여 표현할 수 있음을 숙지하도록 한다. 뿐만 아니라 알고리즘 단계에서 학습했던 여러 가지의 문제들을 블록코딩으로 프로그래밍 할 수 있도록 강의한다.

Table 3. Practiced Detail Curriculum based on AP CSP

Semester Schedule	Lecture Plan	Lecture Content	AP CSP Frame	
			CTP	CO
Week 1	Computational Thinking and Software	Introduce about Computational Thinking Concepts of Software and coding	P.1 P.2	B.D.1
		Learning Objectives of Software Coding		
Week 2-4	Finding Problems in Everyday life	Understanding about Data and Information	P.1 P.2 P.3 P.4 P.5 P.6	B.D.1 B.D.2 B.D.3
		Expression about Information and learning of Structure		
		Finding Problems in everyday life		
		Definition of Problem		
		Understanding the Resolution Process of Problem		
		Problem Analysis and Abstraction		
Week 5-7	How to Understanding and Representing Algorithms Methods.	Understanding about Algorithms	P.2 P.3 P.4	B.D.3 B.D.4
		Method of Representation of Algorithms		
		Algorithm design 1 (Learning of Sequential, Select and Loop)		
		Algorithm design 2 (Understanding Sorting and Searching)		
Week 8-12	Coding learning method using Block Program.	Types of Educational Block Programming Languages	P.1 P.2 P.3 P.4	B.D.1 B.D.2 B.D.4 B.D.5
		Scratch Screen Configuration and Usage		
		Types and Functions of Instruction blocks		
		Variable Concepts and Examples		
		Computational Representation of Daily Life Problems Using Scratch		
		Sequential, Select and Loop Implementation of Scratch		
		Sorting and Searching Implementation of Scratch		
Week 13-14	Mini-Project	Collaboration Project : Explore and Represent about Problem on the Major field using Scratch. -Explore and Analysis the Problem -Design the solution method -Programming -Testing and Modify	P.1 P.3 P.4 P.5 P.6	B.D.1 B.D.2 B.D.4 B.D.5
Week 15	End of Term	Lecture Review		

• Week 13~14에는 팀 또는 개인으로 구분해 자신의 관심 또는 전공분야에 관련된 문제들을 탐색하고 해결할 수 있는 융합프로젝트를 진행한다. 이 과정에서 하나의 프로그램을 기획하는 첫 단계부터 이를 완성해 나가는 과정을 거칠 수 있도록 강의를 진행한다. 강의의 마지막은 학생들이 개발한 작품 또는 프로그램에 대한 발표 및 시연과정을 거치고 교수자는 이에 대한 피드백을 전달하여 완성도 높은 프로젝트의 결과물을 산출할 수 있도록 한다.

• Week 15에는 기말 시험 및 리뷰로 강의를 마무리 한다.

IV. Case Study and Result Analysis

1. Case Study

A대학 인문대학(전공분야 : 국어국문학, 철학, 디자인창의학, 언어학과, 고고학) 3~4학년 학생들을 대상으로 약 40시간 교육을 실시한 후 그 결과를 분석 하였다. 단, 이 강의는 1일 3시간 13일간 수업하였고, 중간고사와 기말고사 시험은 따로 치루지 않았으므로 한 학기당 45시간 강의와 동일한 커리큘럼으로 간주하여 진행되었다.

또한 강의의 마지막 단계에서는 학생들이 전공분야 또는 관심 있는 주제를 선택하여 SW코딩과 융합한 미니프로젝트를 수행하였다. 강의에 참석한 학생들의 95%가 개인 프로그램을 완성하였고, 이들 모두 문제해결방법에 관한 작품발표와 시연회를 진행하였다. 아래의 그림7~그림11은 융합프로젝트 결과 화면을 발췌한 것으로 철학과 학생이 개발한 교육용 게임프로그램의 일부 내용이다.

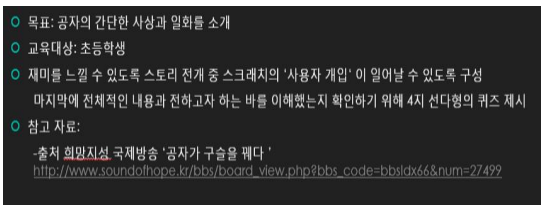

Fig. 7. Program abstract

Fig. 8. After problem analysis, divide by unit

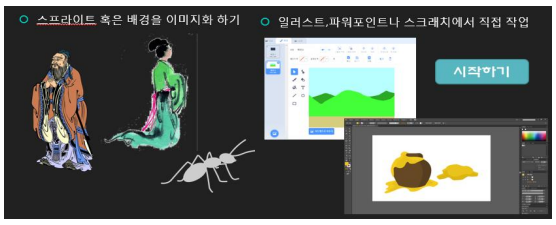

Fig. 9. Design for Program

Fig. 10. Algorithm Block coding

Fig. 11. Complete Program

2. Result Analysis

이 논문에서 제안한 교육모델 수행 이전과 이후의 학생들의 컴퓨팅사고를 기반으로 한 표현과 융합의 문제해결 정도를 비교하기 위하여 설문평가를 시행하였다. 설문평가는 강의의 수강 시작 전과 강의 종료 후에 각각 진행했으며 설문의 기준은 다음 다섯 가지 항목과 같다.

- 항목 1 - 문제 추상화의 정도
- 항목 2 - 추상화 한 문제의 개별요소 구분 정도
- 항목 3 - 컴퓨팅사고를 기반으로 한 알고리즘 표현 가능 정도
- 항목 4 - 블록 코딩 프로그램 사용능력 정도
- 항목 5 - 전공 또는 관심분야와 SW코딩의 융합 정도

강의시작 전에 실시한 설문평가와 종료시점에서 실시한 설문평가는 동일한 항목으로 조사하였으며, 문제해결능력의 정도는 '표현하기 너무 어렵다'를 1로 '매우 잘 표현할 수 있다'를 5로 코딩하여 진행하였다. 평가의 분석 결과, 수강 이전의 전체 항목평균은 2.57에서 수강이후 4.06으로 약 30% 증가하였다. 강의 전/후 각 항목별 평균값의 차이는 그림12와 같으며, 문항별 증가율은 항목1 10%, 항목2 11%, 항목3 51%, 항목4 35%, 항목5 42%로 각 항목별로 학생들의 표현가능 정도가 모든 항목에서 증가함을 확인할 수 있었다.

이 중 항목1부터 항목3까지는 문제해결을 수행하는데 순차적과정이 필요한 항목들이며, 항목3은 항목1과 항목2가 수반되어야 좀 더 효율적인 결과 표현이 가능한 요소이다. 또한 항목5의 경우 전공분야와 SW코딩의 융합정도를 평가하는 항목이다. 특히 이 두 가지 항목들이 높은 증가율을 보여주는 것은 본 제안 교육모델을 통하여 비전공자 학생들의 분석 및 문제해결능력을 키우고, 창의력과 다양한 분야의 융합적 사고능력 증진이 가능함을 반추한다고 볼 수 있다.

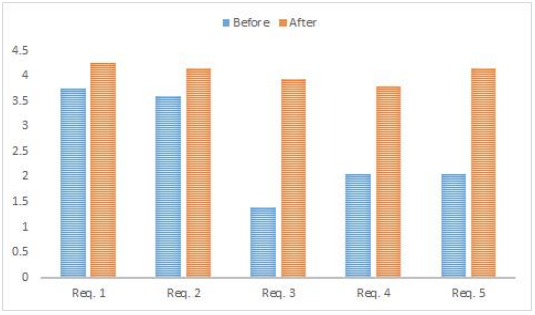

Fig. 12. Average before and after training course

또한 본 제안 모델 수행 이전과 이후의 학생들의 컴퓨팅사고의 문제해결 정도가 유의적인 차이가 있는지를 확인하기 위하여 R프로그램을 이용하여 대응표본 t검정(Paired T-Test)을 수행하였다. 이 때 유의수준은 0.05, t 기각치 2.093 하에서 검정을 진행하였으며 결과는 표 5와 같다.

Table 4. Result Analysis for proposed Model

	t value	p value
Req. 1	-2.032347112	0.056336641
Req. 2	-2.772897981	0.012117511
Req. 3	-10.86025356	0.000000001
Req. 4	-7.314762274	0.000000617
Req. 5	-7.041298293	0.000001057

강의 수강전보다 각 설문문항의 t값은 기각치 2.093 이하로 모두 유의미한 것으로 평가되었으며, 항목1~4의 경우 유의수준 0.05를 기준으로 p-value 가 유의수준보다 작은 것으로 나타난다. 이것 역시 제안하는 교육모델의 기반으로 강의를 수강한 학생들의 컴퓨팅사고력과 문제해결 표현능력정도가 전반적으로 향상되었음을 보여준다.

예외적으로 문제의 추상화의 정도를 평가하는 설문문항 1의 경우 유의수준을 0.05를 초과하는 것으로 나타났으나 해당항목의 강의 전/후 평균값은 증가하였으며, 교육대상이 고등교육과정에 해당하는 학부생인 점과 이들의 전공분야가 인문대학임을 고려하였을 때, 문제 추상화의 정도가 하락되었다고 보기 어렵다.

V. Conclusions

SW중심사회에서 컴퓨팅사고를 기반으로 자신의 전공 및 전문영역에서 표현하는 것은 이제 필수적인 요인으로 간주된다. 그러나 컴퓨팅프로그래밍을 학습하지 않은 비전공자 학생들에게 문제해결 방법을 알고리즘화 하는 방법은 익숙하지 않으며, 이것을 컴퓨터 및 모바일 기기 등을 활용하여 결과물로 표현하기에는 많은 어려움을 겪을 수 있다.

따라서 이 논문에서는 비전공영역과 컴퓨팅사고의 표현방법을 융합하고 신장하기 위하여 교육과정 모델을 제안하였다. 제안하는 교육과정 모델은 AP CSP의 평가기준을 프레임으로 준거를 두었으며, 비이공계 학생들이 소속되어 있는 전공 관련분야 또는 일상생활에서 관심 있는 분야에 대한 문제를 선정하고, 이 문제들을 해결할 수 있는 방안을 모색하여 알고리즘으로 표현할 수 있도록 구성하였다. 제안모델에 대한 효율성 및 컴퓨팅사고력 향상 정도를 검토하기 위해 제안교육과정 모델에 따라 강의를 진행하고, 강의 시작 전/후에 실시한 설문조사를 수행하였다. 수행결과 학생들의 문제파악과 분석능력, 컴퓨팅기술능력 그리고 융합능력이 전체 평균 30%가 더 향상되었음을 확인하였다.

이 연구에서는 학생들의 자가평가를 통하여 교육모델 적용 이전과 이후의 효율성을 검토하였다. 그러나 연구의 완성도 및 컴퓨팅사고의 진전 수준을 정량적으로 측정하기 위하여 교육과정 마지막 단계에서 수행하는 융합프로젝트를 기준으로 한 객관적인 평가도구에 관한 추가적인 연구를 수행할 예정이다.

REFERENCES

[1] Schwab, K. "The Fourth Industrial Revolution by Klaus Schwab." Geneva: World Economic Forum. 2016.

[2] Wing, Jeannette M. "Computational thinking." Communi cations of the ACM, VOL.49, No.3, pp.33-35, 2006.

[3] Lim, Yongtaek. “Analysis on abroad e-Learning for SW education.” Diss. The Graduate School Seoul National University, 2017.

[4] Nah, JeongEun. "Analysis of Computational Thinking Learning Effect through Learner Observation" Korean Journal of General Education, Vol.11, No.3, 2017.

[5] Kim Wan Seop. "A Study on the Recognition of Freshman on Computational Thinking as Essential Course.", Culture and Convergence, Vol.39, No. 6, pp.141-170, 2017.

[6] Su-Young Pi. "A Study on Coding Education of Non-Computer Majors for IT Convergence Education.", Journal of Digital Convergence, Vol.14, No.10, pp.1-8, 2016.

[7] Sung-Hee Jin, Soobong Shin, “Case Study and Needs Analysis on Convergence Education in Engineering Collegespp.”, 2013 Journal of Engineering Education Research, Vol. 16, No. 6, pp.29~37, November 2013.

[8] Wing,J.M., “Computational Thinking and Thinking About Computing.” Philosophical Transactions of the Royal Society, 366, pp.3717-3725, 2008.

[9] Kyungkyu Kim, Jongyun Lee, “Analysis of the Effectiveness of Computational Thinking-Based Programming Learning.” The Journal of Korean Association of Computer Education, 19(1), pp.27-39, Jan, 2016.

[10] Peets. M., “The development of ProgeTiger programme”. Retrieved from http://twinspace.etwinning.net/files/collabspace/2/62/562/562/files/a7426d.pdf, 2014.

[11] Retrieved from http://progetiiger.ee/?q=

[12] “Innovation Centre in HITSA. Programming at Schools and Hobby Clubs.“ Retrieved from http://www.innovatsioonikeskus.ee/en/programming-schools-and-hobby-clubs.

[13] “Koodikerho opettaa suomalaiset lapset ohjelmoimaan” By LUMA Center. LUMA-sanomat. January 15, 2015.

[14] Seungki Shin, Youngkwon Bae, “Review of Software Education based on the Coding in Finland” Journal of The Korean Association of Information Education, Vol. 19, No. 1, pp.127-138, March 2015.

[15] Tiina Heinilä. “Future will be built by those who know how tocode” SITRA, July 11, 2014.

[16] Koodi2016. “Koodi2016-ensiapua ohjelmoinnin opetta miseen peruskoulussa“ Retrieved from https://s3-eu-west-1.amazonaws.com/koodi2016/Koodi2016_LR.pdf

[17] Alves, Nathalia Da Cruz, Christiane Gresse Von Wangenheim, and Jean CR Hauck. "Approaches to Assess Computational Thinking Competences Based on Code Analysis in K-12 Education: A Systematic Mapping Study." Informatics in Education 18.1 2019.

[18] CSTA K-12 Computer Science Standards: Revised, 2011

[19] “Spread of excellent software education models by expanding leading schools”, Ministry of Science and Technology, 2018.

[20] “Science and Technology ICT Policy and Technology Trends” Vol.56 2015,

[21] Jaewon Choi, Ho Lee, Jungmin Kim, Juho Song, “A Comparative Analysis of Curriculums for Software-related Departments based on Topic Modeling” The Journal of Society for e-Business Studies, Vol.22, No.4, pp.193-214, November 2017.

[22] Hwang, Yohan, Kongju Mun, and Yunebae Park. "Study of Perception on Programming and Computational Thinking and Attitude toward Science Learning of High School Students through Software Inquiry Activity: Focus on using Scratch and physical computing materials." Journal of the Korean Association for Science Education, Vol.36, No.2 , pp.325-335, 2016

[23] Kim, Seong-Won, and Youngjun Lee. "Development of a Software Education Curriculum for Secondary Schools." Journal of The Korea Society of Computer and Information, Vol.21, No.8, pp.127-141, 2016

[24] Park, SunJu. "Analysis of Computational Thinking Level Through the Scratch Project Analyzation.", Journal of The Korean Association of Information Education Vol. 22, No. 6, pp. 661-669, 2018,

[25] Lee, Hyun Jean. "A study on the necessity of convergence approach in coding education through case-study interview with individual Scratch learner", Basic formative research, Vol.18, No.6, pp.488-500, 2017

[26] College Board (2017). AP Computer Science Principles: Including the Curriculum Framework. New York: College Board.

[27] Retrieved from https://apcentral.collegeboard.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf

[28] Retrieved from https://apcentral.collegeboard.org/pdf/ap-program-guide-2018-19.pdf

Authors

EunYoung Cheon received the B.S. and M.S. degrees in Computer Engineering from Hannam University, Korea, in 2002 and 2004, respectively. And she finished the coursework in computer science & engineering from Chungnam National University. She joined the instructor of the Department of Mechanical Automotive Engineering at Kongju University, Cheonan, Korea, in 2004 and Department of Computer Science & Engineering at Chungnam University, Daejeon, Korea, in 2009. She is currently a Instructor in the Department of Mechanical Automotive Engineering at Kongju University and Computer Science & Engineering at Chungnam National University. She is interested in SPLE, MDA, software architecture and sotfware education.