Journal of Digital Convergence
Vol. 17. No. 10, pp. 23-35, 2019

ISSN 1738-1916
https://doi.org/10.14400/JDC.2019.17.10.023

이해중심 SW기초교육 프로그램의 컴퓨팅사고 효과성 검증 연구

오경선[1], 권정인[2*]
[1]건국대학교 상허교양대학 조교수, [2]상명대학교 계당교양대학 조교수

A Study on the Verification of Computational Thinking Effectiveness of Understanding-Oriented SW Basic Education Program

Kyung-Sun Oh[1], Jung-In Kwon[2*]
[1]Assistant Professor, Sang-Huh College, Konkuk University
[2]Assistant Professort, College of Kyedang General Education, Sangmyung University

요 약 많은 대학교가 4차 산업혁명이라는 시대적 흐름에 따라 컴퓨팅사고로 문제해결력을 지닌 인재양성을 위해 SW교육을 활발히 진행하고 있다. 본 연구는 비전공대상의 컴퓨팅사고 향상을 위한 이해중심의 SW교육과정이 필요하다고 보았으며 이를 실현하는 구체적 개입으로 이해중심의 SW기초교육을 소개하고 그 효과성을 확인하고자 하였다. 이러한 목적을 달성하기 위해 백워드설계모형을 적용한 활동중심 컴퓨팅사고 교육과 프로그래밍교육을 하나의 이해중심의 SW기초교육 프로세스로 설계하였다. 이후 15주 수업에 적용하고, 3차례에 걸쳐 검사를 실시하여 비전공자들에 대한 컴퓨팅사고의 정의적인 측면을 분석하였다. 연구 결과 활동중심의 SW기초교육의 컴퓨팅사고효능감과 컴퓨팅사고인식에 있어서 긍정적인 효과를 검증하였다. 본 연구는 특정대학의 일부 학생들을 대상으로 실시하여 연구결과를 일반화하는데 다소 무리가 있으나, 교육 현장에서 백워드설계모형을 적용한 이해중심의 SW기초교육이 컴퓨팅사고를 향상시킬 수 있는 효율적인 방법의 하나로 적용될 수 있을 것으로 기대한다.

주제어 : 컴퓨팅사고, SW교육, 프로그래밍교육, SW기초교육, 백워드설계모형

Abstract In order to cultivate talented people who have problem solving ability due to computational thinking according to the trend of the fourth industrial revolution, each university is actively promoting software education. This study suggests that understanding-oriented SW curriculum is needed for non-majors students to improve computational thinking. In order to achieve the purpose of the study, this study designed the basic education program based on the understanding of the SW with the backward design model. The SW Basic Education Program was applied to 15 weeks of instruction and conducted three surveys. The positive effects of the understanding-oriented SW basic education on the computational thinking efficacy and the computer perception were verified. In addition, it was found that the understanding-oriented computational thinking and programming education are effective when they are linked to one process. It is expected that understanding-based SW based education, which uses the backward design model, can be applied as one of the efficient ways to improve computational thinking in the education field.

Key Words : Computational Thinking, SW Education, Programming Education, SW Basic Education, Backward

*Corresponding Author : Jung-In Kwon(jikwon@smu.ac.kr)
Received July 10, 2019 Revised September 6, 2019
Accepted October 20, 2019 Published October 28, 2019

1. 서론

최근 4차 산업혁명과 함께 급속도로 변화되어 가고 있는 우리의 삶은 창의 · 융합형 인재를 필요로 하고 있다.

과거에는 한 분야에 전문적이고 뛰어난 기술을 만드는 것이 궁극적인 목표였다면 이제는 다양한 분야를 연결하여 새로운 가치를 창출 할 수 있는 것으로 바뀌었고 이는 4차 산업혁명을 이끌 주요 요소가 되었다. 세계는 지금 4차 산업혁명을 이끌 인재 양성에 힘을 쏟고 있고 그 중심에 컴퓨팅사고가 있다[1].

최근 이러한 맥락으로 소프트웨어교육(SW교육)의 필요성에 대한 공감이 커지면서 관련 교육을 확대하고 있다. 국내의 경우 초등학교와 중학교 교육과정에서 소프트웨어교육을 의무화하여 어린 시절부터 컴퓨팅사고를 통해 문제를 해결할 수 있는 힘을 기를 수 있도록 하고 있다. 또한 직업생활과 직접적인 관련이 있는 대학교에서는 2015년부터 소프트웨어중심대학을 필두로 소프트웨어 코딩에 관련된 교과목을 개설하고 필수로 이수하도록 운영하는 사례가 증가하고 있다[2].

비전공자에게 SW교육의 목적은 단순히 SW개발자를 양성하는 것이 아니라 실제 문제 상황에서 해결할 수 있는 자신의 아이디어를 컴퓨터과학 원리를 활용하여 표현해 낼 수 있는 능력을 길러내는 것이다. 새로운 시대에는 새로운 교육법이 필요하듯 컴퓨팅사고는 4차 산업혁명에 적합한 인재를 만들기 위해 기존의 프로그래밍언어 교육에서 탈피하여 문제정의를 통해 무엇인 문제인지 파악하고 문제를 해결할 수 있는 다양한 아이디어를 SW관점에서 제안하고, 설계, 개발할 수 있는 교육을 진행할 필요가 있다. 프로그래밍언어는 그 자체가 목적이 아니라 사고를 향상할 수 있도록 도와주는 도구로 사용하는 것이다[3].

이러한 이유로 프로그래밍언어를 통해 컴퓨팅사고능력이 향상될 것이라고 기대하고, 국내의 컴퓨팅사고력 향상을 위한 SW교육은 대부분 교육용프로그래밍언어, 프로그래밍 학습활동의 연구가 대부분이다[5]. 같은 맥락으로, 미래 직업생활과 직접적인 영향이 있는 대다수 대학교는 비전공학부생대상의 SW 기초교육인 필수교양을 특정 프로그래밍언어(예: Python, 스크래치, 앱인벤터, C)를 활용한 문제해결과 컴퓨터과학 원리와 개념으로 구성하고 있다.

그러나 실제 수업을 진행하다 보면 컴퓨팅사고력보다는 프로그래밍언어를 가르치는데 중점을 두기도 한다. 또한 비전공 학부생들은 SW의 중요성은 인식하고 있으나 프로그래밍교육으로 SW교육 자체를 어려워하고 있어 자신의 아이디어를 SW관점에서 풀어내는 것은 쉽지 않다. 이러한 상황에서는 컴퓨팅사고과정을 통한 문제해결도 힘들고, 지속적인 SW교육 또한 더욱 어려울 수밖에 없다.

이는 프로그래밍활동으로 컴퓨팅사고력을 향상시키기 위한 노력을 제한하기 때문이다. 자신의 맥락에서 컴퓨팅사고를 통해 문제를 해결하기 위해서는 다양한 부분을 고려해야 하므로 프로그래밍활동으로 이를 제한 할 필요는 없다[16,17].

컴퓨팅사고 향상을 위해서는 컴퓨팅사고의 본질적인 내용을 이해하고 다양한 상황에 적용할 수 있는 이해중심교육이 필요하다[4,5]. 이러한 배경으로, 본 연구의 목적은 비전공대상의 컴퓨팅사고 향상을 위한 이해 중심의 SW기초교육을 소개하고 그 효과성을 확인하는 것이다. 컴퓨팅사고는 문제해결능력으로 확실한 개념의 이해과 연습으로 그 능력을 향상할 수 있다[18]. 이해중심교육을 위해 백워드모형으로 교수설계를 하였다. 백워드설계모형은 학습내용에 대한 궁극적인 이해가 바탕이되고, 전이를 통해 다른 다양한 상황에서 적용해 볼 수 있도록 설계하는 것이다. 백워드 설계 모형을 통해 컴퓨팅사고에 대한 영속적인 이해가 가능하다. 현재, 컴퓨팅사고관련 연구에서 수업모형 중심으로 활발히 진행되고 있지만 컴퓨팅사고의 백워드 교수설계에 관련된 연구는 미비한 실정이다[19].

따라서 본 연구에서는 컴퓨팅사고의 백워드교수설계 모형을 적용하였다. 먼저, 프로그래밍활동으로 교육을 제한하지 않고 다양한 부분을 고려하여 활동중심의 컴퓨팅사고 교육과 프로그래밍교육을 하나의 프로세스로 조직한 후, 백워드모형으로 교수 설계하였다. 이후 실제 수업을 통해 비전공자들에 대한 컴퓨팅사고의 정의적인 측면을 분석하여 효과성을 검증하고자 하였다.

연구문제는 다음과 같다.

1) 이해중심의 SW교육전과 후 학습자의 컴퓨팅사고의 인식과 효능감에 유의미한 차이를 보이는가?
2) 활동중심의 컴퓨팅사고 1차 교육 전과 후에 컴퓨팅사고의 효능감과 인식에 유의미한 차이를 보이는가?
3) 프로그래밍교육 전과 후에 컴퓨팅사고 인식과 효능감에 유의미한 차이를 보이는가?

2. 이론적 배경

2.1 백워드설계모형

백워드설계는 학습자들의 '진정한 이해'를 강조하여

교육과정을 설계하는 방법으로 단원 수준의 설계에 초점을 주로 두고 있다. 이 설계 방식은 Tyler모형과 Bruner의 내용모형에 기반을 두고 발전하였다. 1998년 Wiggins와 McTighe에 의해 제안된 백워드설계모형은 미국 교육 현장과 교원연수기관에서 널리 활용되고 있으며 프로그램의 효과도 검증되고 있다[6]. 백워드설계모형의 주요특징은 다음과 같이 크게 세 부분으로 구분할 수 있다[7].

- 첫째, 절차상의 측면에서 목표도달여부를 확인할 수 있는 평가를 강조한 설계 방법
- 둘째, 전이 가능성이 높은 주요 아이디어를 가르치는데 초점을 둔 설계 방법
- 셋째, 학습자의 진정한 이해 강조

이러한 특징을 지닌 백워드설계모형은 설계를 시작할 때 최종 결과를 고려하고 계획하는 단원 수준의 설계 방식이다. 즉, 학습자의 진정한 이해를 목적으로 하는 교과라면 백워드설계는 적용할 수 있는 설계 방식이다[6]. 백워드설계모형은 Table 1에서 제시한 바와 같이 '바라는 결과', '이해의 증거결정', '학습경험계획' 단계로 이루어진다[8].

Table 1. Backward Design Stage

Stage 1	Stage 2	Stage 3
The end result is that the students will~	Need proof of ~ students are capable of doing.	Need to ~ in a learning situation

전통적으로 이루어지는 교수설계방법은 교육과정에 제시된 목표를 기준으로 수업내용과 방법을 먼저 설계하고 목표도달을 파악한다. 전통적설계모형에서의 교수자는 교과의 내용을 단순 전달하거나 흥미위주의 수업을 조직하는 문제가 발생할 수 있다. 반면, 백워드설계모형은 학습내용에 대한 궁극적인 이해가 바탕이 되고, 전이를 통해 다른 다양한 상황에서 적용해 볼 수 있도록 설계하는 것이다. 백워드설계모형에서의 교수자는 유연한 수행, 전이, 문제해결등과 같은 학습결과에 초점을 두고 수업을 조직한다[10].

백워드교수설계를 적용한 교육은 학습자가 수업목표에 대한 높은 이해도를 가지고 학습과정에서 주도적인 역할을 할 수 있기 때문에 교육의 효과성과 관련된 연구가 꾸준히 증가하고 있다. 조현희 · 김종윤(2019)은 2010년부터 2018년까지 백워드설계의 학습효과를 검증한 선행연구 19편 하위 65개 그룹에 대한 양적 메타분석을 실시하였다. 그 결과 다양한 교과에서 학습효과가 매우 높은 것으로 나타났다[20]. 다양한 교과에서 이해중심의 백워드설계모형을 접목하고 있지만 SW교육에서의 백워드설계와 관련된 연구는 이영호 · 구덕희(2015)의 연구 1편으로 아직 미비한 실정이다.

2.2 컴퓨팅사고와 SW교육

Wing(2008)은 '3R(읽기, 쓰기, 셈하기)과 더불어 모든 학습자가 갖추어야 할 기본 능력이고, 추상화(Abstraction)와 자동화(Automation)를 통한 문제해결 능력'이라고 컴퓨팅사고를 정의하였다. 즉, 컴퓨팅사고란 문제 해결 과정에서 핵심요소를 추출하는 추상화 과정을 통해 모델링하여 해법(Solution)을 자동화하는 능력을 의미한다. 컴퓨팅사고는 문제발견 및 정의, 자료 수집과 분석, 문제 해결책 마련을 위하여 다양한 사고(Thinking)를 하고, 그 과정에서 컴퓨팅 능력을 활용하여 SW관점으로 해결안을 설계하고 구현하는 문제해결능력인 것이다[9]. 이러한 컴퓨팅사고의 향상을 위해 프로그래밍언어를 가르친다[21].

Derus & Ali(2012)는 학습자들이 프로그래밍을 배우는 첫 단계에부터 어려움을 겪고 있으며, 문제를 분석하는 능력이 부족하기 때문에 실제 문제로 시작 한 후, 프로그래밍언어를 학습하는 것을 제안하였다[22]. Tedre & Denning(2016)의 연구에서는 컴퓨팅사고 교육을 오직 코딩으로만 가르친다면 문제해결방법에 대한 관점이 좁아질 수 있다고 지적하였다[18]. Denning(2017)은 컴퓨팅사고의 확실한 개념을 학습하는 것이 중요하다고 하였다[21]. 그러나 국내 컴퓨팅사고 향상을 위한 SW교육에 관련된 연구는 프로그래밍활동과 교육용프로그래밍교육이 대부분이다[19]. 컴퓨팅사고력을 향상하기 위해서는 프로그래밍활동으로 교육을 제한하지 않고 다양한 부분을 고려할 필요가 있다[16,17]. 문제해결능력인 컴퓨팅사고력은 복잡한 문제를 지속적으로 해결해 나가는 과정 속에서 향상될 수 있기 때문에 일회성 있는 흥미위주의 내용이나 분철된 내용 중심으로는 학습자들의 심층적 이해를 보장하기 어렵다. 성취기준을 기반으로 개념과 원리들로 해석해야하고, 학습자의 맥락 속에서 지속적인 이해를 지원하는 방식으로 가르칠 때 학습자들의 심층적 이해가 높아져 문제해결에 적용할 수 있다. 따라서 컴퓨팅사고 향상을 위한 교육은 컴퓨팅사고에 대한 확실한 이해를 통해 전이가 가능하도록 해야 하며 이것은 반드시 수행과제로 나타낼 수 있도록 해야 한다[23].

3. 연구방법

3.1 연구대상

본 수업은 컴퓨터비전공자 1학년 대상의 SW기초 교육으로 '자신의 아이디어를 컴퓨팅으로 구현하여 실행 할 수 있도록 '컴퓨터사고' 증진'을 수업목표로 두었다.

본 연구는 경기도 소재의 D대학에서 2단계 SW기초과목을 수강하는 1학년 학부생을 대상으로 하였다. 본 연구에서 연구자가 직접 실험한 해당 교과목 수강 학생 수는 총 106명이다. 총 3회의 설문을 통해 결측값을 제외하고 최종 79명을 추출하였다[24].

2단계 SW기초과목의 목표는 '자신의 아이디어를 컴퓨팅으로 구현하여 실행 할 수 있도록 '컴퓨터사고' 증진 '이다. 연구를 위한 사전설문은 1주 수업(2018.09)에서 시행하였으며 중간 설문은 8주 수업(2018.10)에 시행하였고 사후 설문은 15주 수업(2018.12)에 진행되었다. 학생 대다수가 1단계 SW기초 교육을 받은 경험이 있지만 프로그래밍경험이 있다고 인식하지는 못하는 것으로 나타났다. 응답자의 특성은 아래의 Table 2와 같다.

Table 2. Respondent characteristics

College	experience in programming		Gender		count
	Experience	count	Male	Female	
Humanities	no	14	4	20	24
	have	10			
Science	have	0	3	2	5
	no	5			
Social Sciences	have	22	23	21	44
	no	22			
Engineering	have	0	1	1	2
	no	2			
Arts	have	0	4	0	4
	no	4			
Sum			35	44	79

3.2 연구절차

컴퓨팅사고력 강화에 초점을 맞춰 전문가 설문조사를 3차례 실시한 결과로 나온 내용을 적용하였다[13]. 이것을 2명의 교수자(컴퓨터공학박사1인, 컴퓨터교육1인)가 방학기간동안 내용을 검토 및 확인하였으며 활동유형을 점검하였다. 백워드 설계모형을 적용한 이해중심의 SW기초교육에 대한 개발 세부 절차는 Table 3과 같다.

Table 3. SW basic education development process

Preparation	computational Thinking –Research Papers Analysis Select the final survey item	
Program development	–Computational Thinking Education based on the Backward Design –Choosing Teaching & Learning Method	
Apply & Evaluation	Before	1ST survey
	7Weeks class	activity (Unplugged Learning)
	Middle	2ST survey
	7Weeks class	programming education
	After	3ST survey
Results analysis	Verifying the Effectiveness	

학생들의 컴퓨팅사고 효능감과 인식에 어떤 영향을 미치는지 알아보기 위해 실험집단의 사전, 사후 검사 설계를 하였다. 또한 이해중심의 프로그래밍 교육의 효과성도 함께 알아보기 위해 최종적으로 사전, 중간, 사후 검사를 설계 하였다.

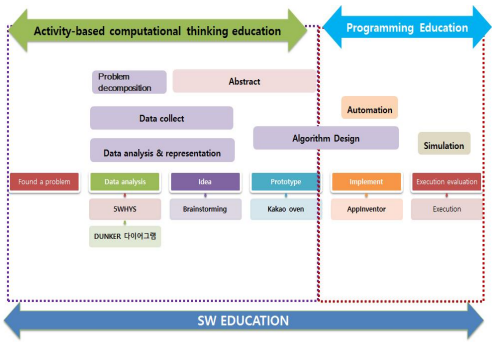


Fig. 1. SW Education Process

본 수업은 '자신의 아이디어를 컴퓨팅으로 구현하여 실행 할 수 있도록 '컴퓨터가 문제를 해결해 나가는 방식으로 생각하는 방법' 증진'을 목표로 설정했다. Fig. 1은 전체 SW교육과정을 나타낸다. Fig. 1에서 나타난 것처럼 학습방법은 크게 두 부분으로 나누어 연속적으로 진행했다. 활동중심의 컴퓨팅사고교육을 7주에 걸쳐 진행했고, 컴퓨팅사고를 통한 프로그래밍 실습을 8주에 걸쳐 운영하였다[11,12].

3.3 연구 도구

본 연구에서는 컴퓨팅사고의 효능감과 인식에 대한 검사 도구는 한국과학창의재단에서 개발한 '초중등 SW교육 실태조사 및 효과성 측정지표' 중 컴퓨팅사고의 효과

성 부분을 추출하여 사용하였다[25]. 추출된 문항은 2명의 전문가(컴퓨터교육전공 교수 1인과 컴퓨터교사 1인) 검토를 거쳐 Table 4와같이 12문항으로 수정 보완하였다.

Table 4. Diagnostic Tool

Number	Variable
1	Abstraction
2	Parallelism
3	simulation
4	Data analysis and representation
5	Problem decomposition
6	automation
7	algorithm
8	Data collection
9	Whether it is possible to solve a problem by a computational thinking
10	Recognize the necessity degree of computational thinking
11	Interest degree of computational thinking
12	The degree of relevance between computational thinking and the real world

3.4 이해 중심의 SW기초교육 수업설계

컴퓨팅사고 향상을 위한 다양한 교수학습방법 및 산출물로는 글쓰기, 프로토타입 및 흐름도(PBL)를 사용하였다. 프로토타입 및 흐름도는 카카오오븐을 활용하였다. 이후 앱인벤터로 프로그래밍교육을 실습하고, 마지막으로 카카오오븐으로 만든 자신의 아이디어를 앱인벤터로 직접 구현하였다. 이것을 적용한 차시별 학습내용은 다음 Table 5와 같다[11-13].

Table 5. Syllabus

Week	Content	Instruction method	Content Elements	Activity
1	Processes of Problem Solving	Lecture Lecture Team Activity	Processes of Problem Solving	Before survey
2-3	Problem-solving Procedures and Expressions	Lecture Team Activity	Problem analysis Problem decomposition	Find and analyze
4-5	Abstraction	Lecture Team Activity	Pattern-Recognition Abstraction	select idea & Idea design Representing abstractions
6-7	Algorithm -Sequential structure/ Selection structure / loop structure	Lecture Team Activity	Prototyping & flow chart	Algorithm write kakao oven
8	Presentation			Presentation & survey
9	Solving Problems with Sequential in Everyday Problems	Practice Team Activity	Algorithm design & programming	App Inventor
10	Solving Problems with selection in Everyday Problems	Practice Team Activity		
11	Solving Problems with Loops in Everyday Problems	Practice Team Activity		
12	Function	Practice Team Activity		
13	App development	Practice Team Activity		
14	Final presentation			Presentation & After survey

학습자들이 컴퓨팅사고를 단순한 이해 수준을 넘어 실제에 적용 가능한 이해를 할 수 있도록 '컴퓨팅사고' 향상을 목표로 '자동화', '추상화', '알고리즘'의 백워드단원으로 재구성하였다. 본 연구에서 사용한 이해중심의 SW기초교육 수업설계는 '프로그래밍교육을 위한 사고력 내용 구성에 관한 연구(오경선, 2016) '에서 제시한 백워드 단원 설계를 본 수업에 맞춰 컴퓨팅사고 함양을 위한 단원으로 수정•보완하였다. 또한 컴퓨터처럼 생각하기(노규성 외 2인)에서 제시된 알고리즘의 내용으로 일반적인알고리즘(1)과 컴퓨터알고리즘(2)로 나누었다. 컴퓨터알고리즘(2)과 추상화의 전체 목표를 가지고 '바라는 결과 확인'은 다음 Table 6과 같다.

Table 6. Backward Design Stage 2 : Abstraction

Abstract : Desired Results
Set goals It solves the problem by eliminating unnecessary elements and reducing the complexity of the problem and extracting key elements.

	Data abstraction removes unnecessary elements for purpo extracts meaningful elements.
	Identify the relationships among the functions, eliminate unnecessary elements for the purpose, and extract key fun elements.
	Remove the unnecessary elements in the story and write th with the necessary elements.

Transition

T1.Students can express themselves as essential elements without unnecessary elements in various problem situations.
T2. Students can prototype with the features and elements that are essential to their ideas.

Understandings

Core Concepts	Explanation	Translate	apply	aspect	sympathy	Self knowledge
Data abstract	Description of data abstraction and characteristics of data abstraction by data type	Identify data abstraction by data type and structure	Select and express data abstraction method according to data type	Select the data abstraction needed for the problem situation	Listen to examples of abstracts without thinking about the type and structure of the data	Reflect on data abstraction process by data type
Function abstract	Explain the concept and features of functional abstraction	Identify the analysis methods according to the type and structure of functional abstraction	Select and express function abstraction method according to relation type	Select the function abstraction needed for the problem situation	Find and express cases represented by functional abstractions	Reflect on function abstraction process
Story abstract	Explain the concept and features of story abstract	Understand how to analyze story abstraction according to type and structure	identify the story structure and focus on the core contents	Identify the entire story and select the abstraction strategies needed for the situation	A story presentation based on story abstraction regardless of the technique of story abstraction	Reflection on the process of story abstraction focusing on core contents and events

Essential Questions

Explanation	How do you create an abstract
Translate	How can the relationship between the real world and abstract be expressed?
Apply	How do you make an abstract in the real world?
Aspect	How does the abstract change according to purpose?
Self knowledge	How can I make an abstract to solve the problem?
Sympathy	What is the abstract used around me and why is it used?

Knowledge	Function
Abstraction concept and abstraction process Concept of data abstraction Data abstraction technique Relationship between data abstraction and functional abstraction Functional abstraction concepts and processes Concept and process of story abstraction	−Analyzing: Observing, measuring, recording, analyzing data −Abstraction design: Expression using writing tools and comparison with prototype tools. −Applying abstraction techniques: Setting up inquiry topics and exploring data −Selecting and Evaluating Abstraction: Summarizing and Evaluating Inquiry Results

Table 7. Backward Design Stage 2−Algorithm

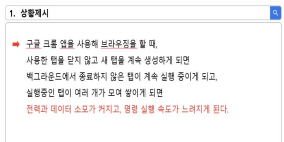
Computer algorithm(2) : Desired Results						
Set goals Develop the ability to list and arrange procedures for problem solving in an orderly fashion. Design algorithms using sequential, selective, and repeating structures.						
Transition T1. Design a logical procedure for implementing your ideas in SW T2. Logically design the process of handling work in various situations.						
Understandings						
Core Concepts	**Explanation**	**Translate**	**apply**	**aspect**	**sympathy**	**Self knowledge**
Basic logical structure	Describe the types of procedures to solve the problem	Identify the various structures in the process of solving the problem	Find a logical structure in a given problem	Compare and contrast differences between basic structures such as sequential, selection, iteration, and recursion	Find and express the basic structure of the algorithm shown in the idea	Reflect the problem−solving process through the basic structure
Essential Questions						
Explanation	What algorithms are used to solve the problem?					
Translate	How does the problem−solving process relate to the various algorithms and structures?					
Apply	How can we design a logical process for problem solving in the real world?					
Aspect	How does the algorithm change depending on the state and condition of the data?					
Self knowledge	How can I design algorithms to find solutions in a number of problem situations?					
Sympathy	What algorithms are used around me and why are they used?					
Knowledge	**Function**					
Concepts and Representations of Algorithms Concept and principle of basic structure of algorithm Basic algorithm types and principles	−Analyze: Observe, measure, record −Algorithm Design: Flowchart of comparison, comparison, sequence and procedure −Algorithm Implementation: Understand the characteristics of the algorithm and create and run a flowchart −Algorithm Implementation: Write flowcharts of each algorithm −Evaluation of algorithm execution result					

3.5 학생활동과 결과

다음은 일부 학생의 15주간 수업 중 활동 자료이다. 4명이 팀을 이뤄 학생 스스로가 느끼는 실제 문제를 발견하고 해결점을 찾아 SW로 구현하기 위한 앱 개발 기획서를 만드는 활동을 각 주차별로 절차화했다[28]. 이때 창의적인 문제를 해결할 수 있는 과제를 부여하기 위해 자신이 직접 문제를 발견하고 정의한 후 아이디어를 찾아 흐름도를 작성 하여 SW개발이 산출물이 만들어 질 수 있도록 하였다. 발표는 모든 팀원이 참여할 수 있도록 하였다. 특히 팀원 당 1개의 기능을 만들어 모두 참여 할 수 있는 장치를 마련하였다.

가장 먼저 일반적인 문제해결절차를 알고리즘으로 설계할 수 있도록 하였고, 추상화를 통해 중요한 요소를 추출하도록 진행하였다. 이것을 반영한 Table 8은 1주에서 7주차까지의 학생들이 활동한 결과이다.

Table 8. Week 1 ~ 7 Student Activities

Learning	Outcomes
Problems found	
Problem analysis	

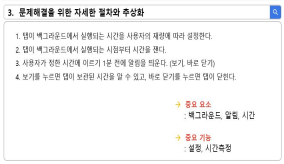
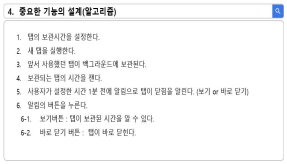
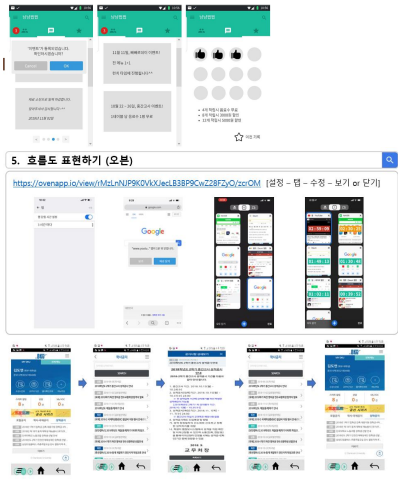
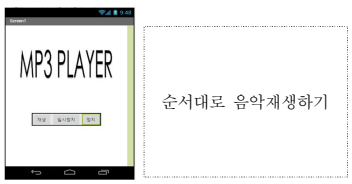
Resolution procedure & Abstract	
Algorithm design	
Prototype	

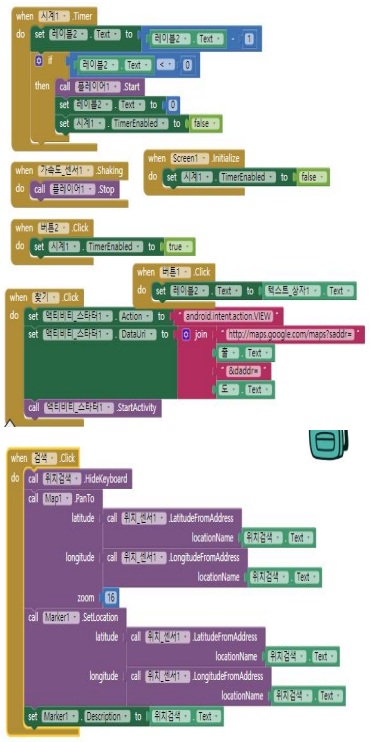
Table 9-1과 Table 9-2는 9주~13주차에서 코딩을 학습 한 후, 자신이 설계한 알고리즘을 토대로 최종 결과물을 만들 수 있도록 프로그래밍을 실습한 내용이다.

Table 9-1. programming education-Student Activities

Learning	Outcome
Sequential	 순서대로 음악재생하기

Selection	 퀴즈 맞추기
Loop	 반복재생하기

Table 9-2. programming education-Student Activities

Learning	Outcome
Final product	 학교 길 안내 어플 제작 

팀원 마다 각 기능을 구현하도록 하고 팀원끼리 공유하여 언제 어디서든지 참여할 수 있도록 하였다. 앞에서 8주에 발표한 것과 동일하게 14주차에 팀원 모두가 발표에 참여하도록 하고, 각자 개발한 어플을 설명할 수 있는 기회를 제공하였다.

학업성취는 학습과정 중 생산물인 과제와 기말고사를 합하여 평가하였다. 과제의 경우 문제를 발견하고 해결하여 SW개발로 산출물을 제출하도록 하였다. 이를 통해 학습자에게 이해중심의 컴퓨팅사고교육이 이루어지도록 진행하였다. 또한 과제 평가는 학생을 참여 시키는 동료평가방법을 도입하여 학습효능감에 긍정적인 영향을 미치도록 하였다.

4. 연구결과

4.1 컴퓨팅사고 관련 문항 요인분석 결과

일반적으로 요인분석 수행 시 추출되는 요인은 표본의 수는 변수 수의 3배에서 4배 이상일 때 안정적이라고 할 수 있다[26]. 본 연구에서 사용한 검사도구의 변수는 12개이고, 응답자수는 79명이기 때문에 표본수가 4배 이상이다. 따라서 안정적인 요인분석이 가능하다. 검사도구의 타당도와 신뢰도를 검증하기 위해 문항들 간의 요인분석을 통해 이상치가 발생되는 항목을 제거하면서 유사항목으로 묶어낸 후, 문항 간의 내적 일치정도를 측정하기 위해 크론바흐-알파 계수를 사용하였다. 모형의 유의성을 검증하기 위해 KMO와 Bartlett를 이용하였다. 요인분석이 가능한 수치는 일반적으로 KMO값은 .5보다 크고 Bartlett는 .05보다 작아야한다. 또한 정보손실을 최소화하면서 주성분분석을 사용하고, 요인 적재치는 배리맥스 방법을 선택하였다. 각 변수와 요인간의 상관관계의 정도를 나타내는 요인 적재치는 수치가 가장 높은 요인에 속하게 된다. 일반적으로 사회과학분야에서는 고유 값이 1.0이상이고 요인적재치가 0.4이상이면 유의한 변수로 간주한다[14,15]. Table 10은 검사도구의 타당도와 신뢰도 결과를 나타낸 것이다. 본 연구에서 사용하는 검사도구는 KMO값이 0.919이고 Bartlett값이 유의수준 .000으로 측정되어 적절하다고 검증되었다.

Table 10. Reliability and Validity Testing tools

Number	Factor		Cronbach α
	CT-Self	CT-Awareness	
1	.926	.221	.965
2	.915	.165	
3	.897	.199	
4	.882	.170	
5	.841	.243	
6	.831	.386	
7	.830	.176	
8	.780	.361	
9	.736	.317	
10	.249	.908	
11	.101	.894	.848
12	.531	.666	
Eigenvalue	8.067	1.484	
KMO& Bartlett Test			
Kaiser-Meyer-Olkin measure.			.919
Bartlett sphericity test		chi-square	984.615
		df	66
		significance probability	.000

수집된 실험집단의 컴퓨팅사고의 역량에 대한 검사 결과에 대해 SPSS 25.0을 이용하여 대응표본 T검정을 실시하였다.

4.2 이해중심SW기초교육의 효과 검증결과

4.2.1 이해중심의 SW교육전과 후 학습자의 컴퓨팅사고의 인식과 효능감에 유의미한 차이를 보이는가?

수업 전 평가에 비해 수업 이후 사후 평가의 평균이 높고, 그 차이가 통계적으로 유의미하다면 컴퓨팅사고 교육이 효과가 있다고 볼 수 있다. 평균을 비교하기 위해 대응표본 T검증을 실시한 결과는 Table 11과 같다. 컴퓨팅사고 효능감과 컴퓨팅사고 인식에 대한 사후 점수 평균이 사전평균 보다 약 0.4높았고, 편차가 낮아지는 것을 확인 할 수 있으며 유의수준은 .04이하로 통계적으로 유의미하다는 것을 알 수 있다. 이것은 학습자들에게 이해중심의 SW교육은 컴퓨팅사고를 적용하여 문제를 해결할 수 있다는 효능감이 증가했고, 컴퓨팅사고에 대한 긍정적인 인식변화가 있음을 알 수 있다. 즉, 이해중심의 SW기초교육은 학습자의 컴퓨팅사고에 긍정적인 인식의 향상과 컴퓨팅사고를 통한 문제 해결력에 긍정적인 영향을 준다는 것으로 해석할 수 있다.

Table 11. Analysis of difference before and after basic education

Factor		Mean	N	SD	Std Error	t	P-value (two-sided)
CT-Self	Before	4.34	79	1.35	.152	-2.008	.048
	After	4.70	79	1.02	.114		
CT-Awareness	Before	4.09	79	1.50	.169	-2.091	.040
	After	4.54	79	1.20	.135		

4.2.2 활동중심의 컴퓨팅사고 1차교육 전과 후에 컴퓨팅사고의 효능감과 인식에 유의미한 차이를 보이는가?

Table 12는 활동중심 컴퓨팅사고 전과 후의 분석결과이다. 활동중심 컴퓨팅사고 수업 전의 효능감은 4.343이 측정된 반면 수업 후에는 4.346이 측정되었다. 또한 활동중심 컴퓨팅사고 수업 전의 인식은 4.097이 측정되었고 수업후의 인식은 4.151로 측정되었다. 특히 수업전 인식과 활동중심 컴퓨팅사고 수업 후 인식 간에는 서로 관련이 있다는 것을 알 수 있다.

Table 12. Activity-based class Before and After Statistics

Factor		Mean	N	SD	Std. Error	Coef	P-value
CT-Self	Before	4.343	79	1.358	.1528	-.151	.185
	After	4.346	79	1.029	.1158		
CT-Awareness	Before	4.097	79	1.509	.1698	-.288	.010
	After	4.151	79	1.252	.1409		

그러나 Table 13에서 나타난 바와 같이 대응표본 T 검증을 실시한 결과 7주간의 활동중심의 컴퓨팅사고 교육 후에는 평균차이가 나타났으나 유의수준이 .05이상으로 통계적으로 유의미하지 않았다. 이는 컴퓨팅사고의 인식과 효능감에 대한 평균적으로 차이가 없다는 것으로 파악할 수 있다. 이미 1학기에 1단계 SW기초교육을 받았기 때문에 2단계의 SW기초교육에서의 활동중심의 컴퓨팅사고교육이 주는 변화를 파악할 수 없다는 것으로 볼 수 있다.

Table 13. Analysis of Difference Between Activity-Based Computational Thinking Education

Factor	Mean	SD	Std Error	t	df	P-value (two-sided)
CT-Self	.003	1.82	.20	-.016	78	.987
CT-Awareness	.054	2.22	.25	-.219	78	.827

4.2.3 프로그래밍교육 전과 후에 컴퓨팅사고 인식과 효능감에 유의미한 차이를 보이는가?

Table 14는 프로그래밍교육 전과 후의 분석 결과이다. 표본을 79로 하여 측정한 결과, 평균값과 표준편차에 차이가 있음을 알 수 있다. 컴퓨팅사고 자기효능감은 7주 동안 활동중심의 컴퓨팅사고교육 후 진행한 2차 조사보다 프로그래밍 교육 후 실시한 3차(사후) 조사에서 약 0.3정도 평균이 높았다. 또한 학습자들은 프로그래밍 교육을 통해 컴퓨팅사고 인식변화가 0.4정도 긍정적으로 상승하였다. 이러한 통계수치가 유의한지 확인하기 위해 t값과 유의수준을 살펴본 결과 Table 14와 같이 t값은 -2.26이하이고, 유의수준은 0.05이하로 통계적으로 유의미한 차이가 있음을 알 수 있다. 이는 활동중심 컴퓨팅사고를 통해 자신의 제안한 문제해결안을 프로그래밍교육으로 직접 구현하고 체험했을 때 컴퓨팅사고에 대한 효과가 있다는 것으로 볼 수 있다.

Table 14. Analysis of difference before and after programming education

Factor		Mean	N	SD	Std. Error	t	P-value (two-sided)
CT-Awareness	Before	4.151	79	1.252	.1409	-2.340	.022
	After	4.557	79	1.210	.1362		
CT-Self	Before	4.346	79	1.029	.1158	-2.526	.014
	After	4.711	79	1.037	.1167		

5. 결론

최근 많은 대학에서는 전교생을 대상으로 SW기초교육을 필수로 운영하고 있다. 이는 미래사회에서 요구하는 컴퓨팅사고를 길러내어 창의융합인재 양성의 토대를 마련하기 위한 것이다[27]. 이러한 배경으로 SW교육을 통해 컴퓨팅사고를 향상하는 많은 연구가 진행되었다. 특히 프로그래밍활동은 컴퓨팅사고력을 향상하기 위한 효율적인 도구로 보고 SW교육을 진행하고 있다. 그러나 컴퓨팅사고는 프로그래밍 활동으로 제한하는 것이 아니라 다양한 접근을 고려해야한다[17,18]. 즉, 컴퓨팅사고의 확실한 개념을 획득하고 연습하는 과정이 고려되어야하는 것이다. 또한 컴퓨팅사고는 단순한 지식의 이해 수준이 아니다. 자신의 문제를 컴퓨팅사고를 통해 해결 가능하도록 전이할 수 있도록 해야 한다. 따라서 컴퓨팅사고를 향상하기 위해서는 단순한 이해 수준이 아니라 영속적인 이해 중심으로 컴퓨팅사고를 통해 문제를 해결 할 수 있는 (doing) 교육으로 설계가 교수설계가 이루어져야한다.

이러한 맥락으로 본 연구는 실제 SW기초교육에서 컴퓨팅사고의 확실한 개념 획득과 연습이 이루어지도록 크게 컴퓨팅사고활동영역과 프로그래밍실습영역으로 순서대로 조직하였다. 이후 이해중심의 SW교육으로 진행하였다. 즉, 학습자 중심의 백워드모형으로 교수설계 한 활동중심의 컴퓨팅사고 교육과 프로그래밍교육을 하나의 프로세스로 운영하는 실험연구를 진행하였다. 실험으로 통해 이해중심의 SW기초교육이 실제 교육에 있어서 컴

퓨팅사고의 효과성이 있는지 판단해 보고자 하였다.

이에 따라 본 연구에서는 대학교 1학기에 SW기초교육을 이수한 1학년 비전공자 학부생을 대상으로 3차례에 걸쳐 컴퓨팅사고의 인식과 효능감에 대해 조사를 실시하였다. 1차 조사는 이해중심의 SW기초교육이 시작되기 전에 실시하였다. 2차(중간) 조사는 7주간의 활동중심의 컴퓨팅사고 교육 후에 실시하였고, 3차(사후) 조사는 중간고사 이후 8주간의 프로그래밍교육 후 실시하였다.

비전공자 학부생을 대상으로 한 연구결과, 이해중심의 SW기초교육은 학습자의 컴퓨팅사고에 긍정적인 인식이 향상되었고, 컴퓨팅사고를 통해 문제를 해결할 수 있다는 효능감이 향상된다는 것을 확인할 수 있었다.

전체 교육과정을 활동중심 컴퓨팅사고 교육과 프로그래밍 교육으로 순차적으로 운영하였다. 첫 번째 과정인 활동중심 컴퓨팅사고 교육을 통해 컴퓨팅사고의 효능감과 인식변화에는 통계적인 차이를 찾을 수는 없었다. 이미 1단계 SW기초교육을 수강했던 경험이 있는 학습자들이기 때문에 활동중심의 SW기초교육을 통해 효과를 찾을 수 없었던 것으로 파악할 수 있다. 이어서 진행된 이해중심의 프로그래밍교육은 컴퓨팅사고의 긍정적인 인식변화가 생겼고 효능감이 상승하였다. 이는 활동중심 컴퓨팅사고를 통해 자신의 제안한 문제해결안을 프로그래밍 교육으로 직접 구현하고 체험했을 때 컴퓨팅사고에 대한 효과가 있다는 것으로 볼 수 있다.

본 연구의 결과를 종합해 보면 백워드설계모형을 적용한 SW기초교육은 학습자들로 하여금 스스로 생각하는 힘과 논리적인 절차를 구현할 수 있는 힘이 생긴다는 것을 알 수 있다. 다시 말해, SW기초교육에서는 이해를 바탕으로 설계한 교육을 통해 긍정적으로 효과를 준다는 것을 연구를 통해 알 수 있었다.

그렇기 때문에 백워드설계모형 기반의 이해중심의 SW교육적 의의는 다음과 같다.

첫째, 이 연구는 백워드설계모형을 적용한 이해중심의 SW기초교육의 긍정적인 효과성을 검증하였다. 비전공자 학부생을 대상으로 프로그래밍활동중심으로 제한하지 않고 수업내용을 재조직하고 설계하여 운영하면 컴퓨팅사고의 긍정적인 효과를 준다는 것을 알 수 있다.

둘째, 백워드설계모형을 접목하여 컴퓨팅사고의 지식과 기능을 하나로 연결할 수 있는 교수설계를 제공하였다. 프로그래밍활동 중심으로 설계한 것이 아니라 활동중심의 컴퓨팅사고 교육과 프로그래밍실습으로 연결하여 컴퓨팅사고를 통한 문제해결로 접근하였다.

셋째, 백워드설계모형을 적용한 SW기초교육과정은 핵심 아이디어와 원리를 이해할 수 있는 지식과 기능으로 단원을 설계하므로 학생들의 영속적인 이해에 한 걸음 더 접근할 수 있다.

이러한 이해중심의 SW기초교육을 통해 학습자는 기초단계에만 머무르는 것이 아니라 SW융합 로드맵의 시작점으로 나갈 수 있다.

더 나아가 실제 수업을 담당하는 교수자는 백워드설계모형을 적용한 이해중심의 SW기초교육은 컴퓨팅사고의 향상이라는 효과적인 수업을 운영할 수 있다. 교수자가 단원을 개발할 때 가르칠 내용의 영속적인 이해를 중심으로 교육내용을 재조직하고, 학생의 이해정도를 점검할 수 있는 평가과제와 기준을 결정 한 후 학습활동을 설계한다. 이러한 과정 속에서 학생들의 수준에 적합한 SW기초교육을 체계적으로 운영할 수 있다.

본 연구는 다만 다음과 같은 한계점을 가진다. 첫째 실험연구로 진행했지만 연구대상의 수가 적고, 계열별로 다양하게 구성하지 못했기 때문에 일반화하기에는 무리가 있다. 둘째, 실험에 참여한 학생들이 1학기에 1단계 SW기초교육을 이수했기 때문에 처음 배우는 1단계 SW기초교육의 학생들에게 일반화하기에 무리가 있다. 이 같은 한계점은 지속적인 후속 연구를 통해 보완해 나가야 할 것이다.

이러한 한계점에도 불구하고 비전공자 SW기초교육에서는 백워드설계모형을 적용하여 직관적으로 체험할 수 있는 이해중심, 실생활 중심의 프로그래밍교육을 통해 컴퓨팅사고를 향상할 수 있는 효율적인 방법의 하나로 이해중심SW기초교육을 적용할 수 있을 것으로 기대한다.

REFERENCES

[1] Education department. (2019). *2015 Revised National Curriculum Retrieved.* http://www.ncic.re.kr/nation.dwn.ogf.inventoryList.do?orgAttNo=10000078

[2] J. S. Sung & H. C. Kim. (2015). Analysis on the international comparison of computer education in schools. *The Journal of Korean association of computer education, 18(1),* 45-54. http://www.koreascience.or.kr/article/JAKO201509365223298.pub

[3] K. S. Noh & S. J. Ahh & K. S. Oh. (2019). *Computational Thinking.* seoul : Ehan.

[4] S. J. Ahh & K. S. Oh. (2015). A study on the relationship between difficulty in learning to program and Computational Thinking. *The journal of korean*

association of computer education, 18(5), 55-62.

[5] C. S. Na, H. Joo, J. J. Lee & D. S. Kim. (2018). Inducing Computational Thinking in Korean SW Education: Synthesizing Standardized Mean Changes through Meta-analysis. *The Journal of Educational Technology, 34(3),* 775-815.
DOI : 10.17232/KSET.34.3.775

[6] Y. N. Lim & H. J. Hwang. (2018). An Analysis of the Characteristics of Curriculum Development Practices by "Understanding by Design": Focusing on the "Identifying Desired Results" Phase of Backward Design. *Korean Association For Learner-Centered Curriculum And Instruction. 18(20),* 243-268.
DOI : 10.22251/jlcci.2018.18.20.243

[7] H. S. Kang. & J. E. Yi. (2010). In Search of the Applicability of Backward Design to Elementary Classroom. *The Journal of Elementary Education). 23(2),* 383-409.
http://db.koreascholar.com/Article?code=347519

[8] McTighe, J. & Wiggins, G. (2004). *Understanding by design: Professional development workbook. Alexandria,* VA : Association for Supervision and Curriculum Development.
DOI : 10.14483/calj.v19n1.11490

[9] Wing. J. M. (2008). Computational Thinking and Thinking About Computing. *Philosophical Transactions of the Royal Society. 366(1881),* 3717-3725.
DOI : 10.1109/ipdps.2008.4536091

[10] K. S. Oh. (2016). *A study on the contents of computational thinking for programming education.* Ph.D. dissertation, Sungkyunkwan University, Seoul.

[11] K. S. Oh, E. K. Su & H. J. Chung. (2018). A study on development of educational contents about combining computational thinking with design thinking. *Journal of Digital Convergence, 16(5),* 65-73.
DOI : 10.14400/JDC.2018.16.5.065

[12] S. Y. Pi. (2016). A Study on Coding Education of Non-Computer Majors for IT Convergence Education. *Journal of Digital Convergence, 14(10),* 1-8.
DOI : 10.14400/JDC.2016.14.10.1

[13] Y. H. Shin, H. J. Jung & E. K. Suh. (2019). Effect of Coding Education Program based on Design Thinking for Non-engineering students. *Korean Association For Learner-Centered Curriculum And Instruction, 19(10),* 351-373.
DOI : 10.22251/jlcci.2019.19.10.351

[14] K. S. No. (2014). *SPSS & AMOS 21.* Seoul: Hanbit.

[15] H. J. No. (2014). *Principal component analysis & factor analysis.* Seoul: Hanol.

[16] K-12 Computer Science Framework Steering Committee. (2016). *K-12 Computer Science Framework(2016).* NY : CSTA.
DOI : 10.1007/s10639-016-9493-x

[17] Alan Bundy(2007). Computational Thinking is Pervasive. *Journal of Scientific and Practical Computing, 1(2),* 67-69.
https://core.ac.uk/download/pdf/28961399.pdf

[18] M. Tedre & P. J. Denning. (2016). The long quest for computational thinking. *Proceedings of the 16th Koli Calling Conference.* (pp. 120-129). New York, NY : Computing Education Research.
DOI : 10.1145/2999541.2999542

[19] H. S. Choi. (2018). Domestic Literature Review on Computational Thinking Development through Software Programming Education. *Journal of Educational Technology, 34(3),* 743-774.
DOI : 10.17232/KSET.34.3.743

[20] H. H. Cho & J. Y. Kim. (2019). A Meta-Analysis of the Effects of Backward Design-Based Instruction. *The Journal of Curriculum Studies, 37(1),* 57-84.
DOI : 10.5230/jgc.2019.19.e7

[21] P. Denning. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60(6),* 33-39.
DOI : 10.1145/2998438

[22] S. R. Derus & A. Z. M. Ali. (2012). Difficulties in learning programming Views of students. *Proceedings of the International Conference(pp.74-78).* Singapore : ICIE.
DOI : 10.13140/2.1.1055.7441

[23] Y. H. Lee & D. H. Gu. (2015). A Study on Instructional Design of Software Curriculum Using Backward Design Model. *Journal of The Korean Association of information Education, 19(4),* 409-418.
DOI : 10.14352/jkaie.2015.19.4.409

[24] W. W. Park & S. Y. Son & H. S. Park & H. S. Park. (2010). A proposal on determining appropriate sample size considering statistical conclusion validity. *Seoul Journal of Industrial Relations, 21,* 51-85.
http://hdl.handle.net/10371/144993

[25] Korea Foundation for the Advancement of Science and Creativity. (2016). *A Study on Surveying the Actual Conditions and Evaluating the Effectiveness of SW Education in Elementary and Secondary Schools.* Seoul : KFASC.
http://www.ndsl.kr/ndsl/search/detail/report/reportSearchResultDetail.do?cn=TRKO201600014678

[26] J. G. Jeong & J. N. Baek & S. S. Kim. (2009). The Development of Adoption Criterion about Multiple Intelligence Theory for Special Class Teachers. *The Journal of Special Education : Theory and Practice, 10(1),* 1-21.
DOI : 10.19049/JSPED.10.1.01

[27] G. D. Kim & K. S. Park. (2018). Educational Strategy for Practical Convergence using Module Curriculum in University. *Journal of the Korea Convergence Society, 9(7),* 205-211.
http://www.earticle.net/Article/A333820

[28] J. H. Ku. (2017). Designing an App Inventor Curriculum for

Computational Thinking based Non-majors Software Education. *Journal of Convergence for Information Technology, 7(1),* 61-66.
http://www.earticle.net/Article/A296598

오 경 선(Kyung-Sun Oh) [정회원]

· 2016년 8월 : 성균관대학교 컴퓨터교육전공(교육학박사)
· 2017년 8월 ~ 2018년 2월 : 단국대학교 강의전담조교수
· 2018년 3월 ~ 현재 : 건국대학교 상허교양대학교 조교수
· 관심분야 : SW교육, 프로그래밍교육, 컴퓨팅적사고
· E-Mail : skyal@konkuk.ac.kr

권 정 인(Jung-In Kwon) [정회원]

· 2014년 2월 : 성균관대학교 컴퓨터교육전공(교육학박사)
· 2015년 3월 ~ 2017년 2월 : 성균관대학교 초빙교수
· 2018년 3월 ~ 현재 : 상명대학교계당교양대학 조교수
· 관심분야 : SW교육, 컴퓨팅적사고, 정보윤리
· E-Mail : jikwon@smu.ac.kr