So-Hyun PARK / International Journal of Industrial Disribution & Business Vol 10 No 11 (2019) 71-80

Print ISSN: 2233-4165 / Online ISSN 2233-5382
doi:http://dx.doi.org/10.13106/ijidb.2019.vol10.no11.71

Design of Algorithm Thinking-Based Software Basic Education for Nonmajors
비전공자를 위한 알고리즘씽킹 기반 소프트웨어 기초교육 설계*

So-Hyun PARK(박소현)[1]

Received: November 07, 2019 Revised: November 07, 2019 Accepted: November 10, 2019

Abstract

Purpose: The purpose of this study is to design the curriculum of Basic College Software Programming to develop creative and logical-thinking. This course is guided by algorithmic thinking and logical thinking that can be solved by computing for problem-solving, and it helps to develop by software through basic programming education. Through the stage of problem analysis, abstraction, algorithm, data structure, and algorithm implementation, the curriculum is designed to help learners experience algorithm problem-solving in various areas to develop diffusion thinking. For Learners aim to achieve the balanced development of divergent and convergent-thinking needed in their creative problem-solving skills. **Research design, data and methodology**: This study is to design a basic software education for improving algorithm-thinking for non-major. The curriculum designed in this paper is necessary to non-majors students who have completed the 'Creative Thinking and Coding Course' Design Thinking based are targeted. For this, contents were extracted through advanced research analysis at home and abroad, and experts in computer education, computer engineering, SW education, and education were surveyed in the form of quasi-openness. **Results**: In this study, based on ADD Thinking's algorithm thinking, we divided the unit college majors into five groups so that students of each major could accomplish the goal of "the ability to internalize their own ideas into computing," and extracted and designed different content areas, content elements and sub-components from each group. Through three expert surveys, we established a strategy for characterization by demand analysis and major/textbook category and verified the appropriateness of the design direction to ensure that the subjects and contents of the curriculum are appropriate for each family in order to improve algorithm-thinking. **Conclusions**: This study helps develop software by enhancing the ability of students who practice various subjects and exercises to explore creative expressions in various areas, such as 'how to think like a computer' that can implement and execute their ideas in computing. And it helps increase the ability to think logical and algorithmic computing based on creative solutions, improving problem-solving ability based on computing thinking and fundamental understanding of computer coding and development of logical thinking ability through programming.

Keywords : Software Education, Computational Thinking, Design Thinking, ADD Thinking, Algorithm Thinking

JEL Classification Code : I20, I21, L86

1. 서론

4 차 산업혁명은 전통적인 제조업 중심 체제를 뒤로하고 모든 기술과 정보를 공유하는 사회적 네트워크와 협업에 의한 새로운 시스템으로의 변화로 기존 산업혁명들과 차별화 된다. 과거에는 독자적 기술을 고집했다면 이제는 다양한 각계 각층의 전문가들이 많은 분야의 정보를 융합

* This research was supported by the MISP(Ministry of Science, ICT & Future Planning), Korea, under the National Program for Excellence in SW (2017-0-00091), supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation)

1 First Author, Lecturer Professor, Dankook SW-Centeric University Project, Korea.
Tel: +82-31-8021-8444, Email: sohyunpark@dankook.ac.kr

하여 새로운 부가가치를 창출해내는 것이 4 차 산업혁명의 핵심요소라고 할 수 있다(Shon, 2019). 이러한 4 차 산업혁명을 맞이하여 현대사회를 이끌 창의 융합형 인재의 필요성이 그 어느 때보다 크게 대두되고 있다. 그러므로, 복잡한 문제 해결 능력이나 여러 사람과의 관계가 얽혀 있는 복합적인 판단에 의한 의사결정을 뒷받침 할 수 있는 역량을 키우는 것이 매우 중요하다. 이에 국가 차원에서 소프트웨어 교육의 중요성을 강조하고, 컴퓨팅 사고에 기반한 소프트웨어 교육을 차세대 산업혁명의 핵심 역량으로 주목하고 있다. 또한, 급변하는 시대에 발맞춰 논리적이면서도 창의적인 컴퓨팅 사고력을 갖춘 인재를 육성하기 위해 소프트웨어 교육과정을 설계하고, 연령에 따라 의무 교육과정을 수립하는 등 세계적으로 코딩 열풍이 불고 있다.

소프트웨어정책과에 따르면 우리나라의 경우 2015 년 소프트웨어 교육운영지침과 교육과정 개정을 통해 소프트웨어 교육을 강조하기 시작하였고, '소프트웨어 중심 사회를 위한 인재양성 추진 계획'을 통해 국가 차원의 정보 교육 과정을 제시하고 소프트웨어 교육을 강조하였다. 이에 2018 년부터 중학교 정보과목을 필두로 2019 년 초•중•고등학교 학생들의 소프트웨어 교육을 필수로 도입하여 의무화하고 있다. 또한, 많은 대학에서 소프트웨어와 코딩과 관련한 교과목을 개설하고 있으며, 전교생을 대상 필수 교과목으로 지정하여 소프트웨어 비전공자들도 코딩 교육을 하는 대학이 늘어나고 있다(Lee, Kwon & Jung, 2018). 과학기술정보통신부에서는 고등교육기관의 소프트웨어 교육을 위해 '19 년 10 월기준 전국 40 개교를 'SW 중심대학'으로 선정하여 대학교육을 소프트웨어 중심으로 혁신함으로써, 소프트웨어 전문인력을 양성하고 각각의 전공과 소프트웨어 소양을 겸비하여 가치를 확산 할 수 있는 융합형 인재육성에 힘을 쏟고 있다. 선정된 대학들은 소프트웨어 전공 학생을 포함한 모든 계열을 대상으로 소프트웨어 교육을 진행하여 창의적 융합 사고를 갖춘 소프트웨어전문·융합인력을 양성하고 있다.

다른 한편에서는 소프트웨어교육이 필수 과정으로 정착하는 것에 대해 우려의 목소리도 있다. 여전히 소프트웨어 활용 능력 함양에만 중점을 둔 교육이 진행되는 교육기관이 많기 때문이다. 실제로 국내 대학의 경우 계열이 다른 전교생에게 동일한 교육과정으로 프로그래밍 언어 교육을 진행하거나, 워드프로세서, 파워포인트, 엑셀 등과 같은 응용 프로그램의 활용 위주의 교육이 이루어지는 경우가 많다(Pi, 2016). 이러한 문제점을 극복하기 위해 Suh(2017)는 창의적 문제해결 능력 함양에 목표를 둔 '디자인씽킹 기반 코딩교육 프로그램'을 제안하여 비전공계열 학생들을 대상으로 하는 교수 프로그램을 개발하였다. 이 프로그램은 활동주체가 학생이며, 학생 스스로 일상생활에서 겪어온 불편한 문제들을 정의하고 이를 해결하기 위해 가능한 방법들을 모색한 후 다양한 아이디어를 도출한다. 그 다음 최선의 아이디어를 선택하여 소프트웨어적으로 접근 해결해 나감으로써 창의 문제 해결능력에서 필요로 하는 발산적 사고와 수렴적 사고의 균형 잡힌 발전을 목표로 한다. 또 다른 문제점으로 교육목표와 방향이 명확하지 않거나 계열 및 학습자 특성을 반영하지 못하고 있다는 지적이 있다(Yoon, 2017). 이에 비전공 계열 학생을 위한 단계적 코딩 교육의 전략 연구가 필요하다(Shin, 2019).

따라서, 본 연구에서는 대학의 컴퓨터나 소프트웨어 비전공 학생들 대상의 소프트웨어 교육에 있어 기존의 프로그래밍언어를 습득하는 일차원적이고 획일적인 소프트웨어 코딩 교육이 아닌 다양한 관점과 새로운 질문을 통해 창의적인 문제를 정의하고 해결 방법을 찾아 소프트웨어로 해결하기 위한 교육과정을 설계하고자 한다. 이를 위한 이론적 배경으로 컴퓨팅 사고에 대해 연구하였고, 교육과정의 해답을 ADD Thinking 에서 찾았다(Suh, Oh, & Chung, 2018). 본 연구에서 제안한 교육과정은 ADD Thinking 의 두 번째 사고인 알고리즘씽킹(Algorithm Thinking)을 기반으로 하여 현실에서 접하는 문제 해결을 위한 컴퓨팅 사고 기반의 논리적 설계가 가능하도록 한다. 또한, 비전공자 학생들의 전공에 따라 인문, 사회, 자연, 공학, 예체능 5 개의 계열로 세분화하여 설계하였다. 이는 계열별 학생들의 특성에 맞추어 교육과정을 다르게 설계함으로써 논리적, 알고리즘적 사고능력 함양에 이바지하고, 학생들의 코딩 능력 향상 및 계열별 응용 분야에 적용할 수 있는 능력을 키우는 것을 목적으로 한다. 계열별 교육과정 설계를 위해 3 차에 걸쳐 전문가 설문을 진행하였고, 그 결과를 분석 반영하여 설계의 신뢰도를 검증하였다. 본 연구는 학생들이 코딩이나 프로그래밍 언어를 처음 접했을 때 가질 수 있는 거부감이나 어려움에 따른 부담감, 나에게는 불필요한 교육이라는 생각을 낮출 수 있고, 팀 활동을 통해 협업과 소프트웨어 교육의 필요성과 가치를 높일 수 있을 것으로 기대한다.

2. 선행연구 고찰

2.1. 컴퓨팅 사고

컴퓨팅 사고는 컴퓨터가 일을 처리하는 수행과정을 이해하고 논리적인 설계를 통한 문제 해결의 접근 방식을 뜻하는 것으로 미국의 Wing 교수에 의해 처음 언급되었다(Wing, 2006).

Bennett, Koh, and Repenning(2013)에 따르면 컴퓨팅 사고는 컴퓨팅의 기본적인 개념과 원리를 기반으로 문제를 효율적으로 해결할 수 있는 사고능력으로 소프트웨어 교육에서 알고리즘씽킹의 중요성을 강조한다. 이에 컴퓨팅 사고 기반의 교육과정은 추상화를 통한 모델링과 자동화하는 능력이 포함되어야 한다. 즉, 추상화 과정을 통해 현실에서 접하는 문제를 정의하여 모델링한 후 문제 분석과 분해를 통해 패턴을 찾고, 프로그래밍 교육을 통해 이를 자동화할 수 있어야 한다. 그러나, 컴퓨팅 사고는 코딩활동으로 제한되어서는 안되며 다양한 방식으로의 접근법이 필요하다(Tedre & Denning, 2016). Oh and Kwon(2019)은 비전공자 대상의 컴퓨팅 사고 향상을 위한 이해중심의 SW 교육과정이 필요하다고 보았으며, 이해중심의 소프트웨어 기초교육을 소개하고 그 효과성을 확인하고자 하였다.

2.2. ADD Thinking

ADD Thinking 은 4 차 산업혁명 시대에 적합한 인재 육성을 위해 "창의성, 비판적 사고능력, 융•복합적 사고 능력"을 배양하는데 목표를 두고 있으며, 창의적으로 사고하여 소프트웨어로 문제를 해결하는 방식에 익숙하게 하고, 소프트웨어를 자신의 영역에서 내재화하여 새로운 가치를 만들어 내는 능력을 키우기 위한 방법론이다(Suh et al. 2018).

ADD Thinking 은 Figure 1 과 같이 소프트웨어 코딩을 구현하기 위한 3 가지 사고로 구성되어 있다. 첫 번째는 창의적 사고를 통해 혁신적인 문제 해결을 목표로 하며 본질적인 문제의 원인을 찾아 문제를 재정의하고 창의적인 문제 해결을 돕는 디자인씽킹(Design Thinking)이다. 두 번째는 컴퓨터 기반의 논리적 알고리즘 사고를 배움으로써 컴퓨터처럼 생각하는 법을 이해하고 더 나아가 비판적 사고를 키울 수 있는 알고리즘씽킹(Algorithm Thinking)이다. 세 번째는 딥러닝 사고를 통해 다양한 정보들을 바탕으로 다양한 관점과 시각으로 사고하고, 주어진 문제를 새롭게 해석하고 해결하는 융•복합적 사고를 키우는 딥러닝씽킹(Deep Leaning Thinking)이다(Wang, Widrow, Zadeh, Howard, Wood, Bhavsar, Budin, Chan, Fiorini, Gavrilova, & Shell, 2016).

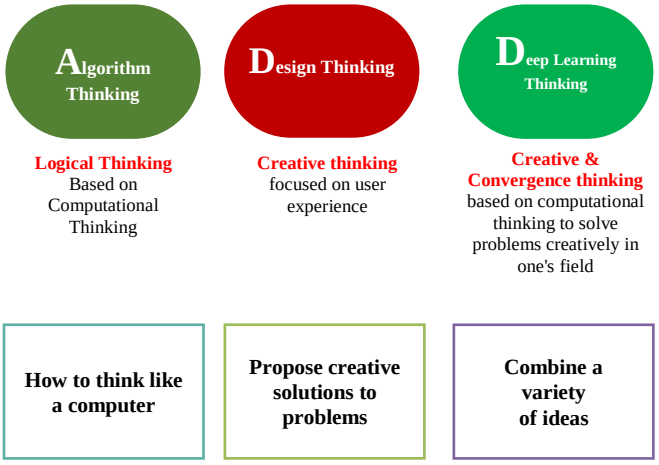

Figure 1: ADD Thinking

3. 연구방법

본 연구에서는 비전공자 학부생 대상의 알고리즘씽킹 향상을 위한 소프트웨어 기초교육을 설계하고자 하였다. 이를 위해 국내외 선행연구 분석을 통해 내용을 추출하고, 컴퓨터교육, 컴퓨터공학, SW 교육, 교육학 전문가를 대상으로 준개방 형태의 전문가 조사를 실시하였다. 전문가 조사는 전문가 집단의 의견과 판단을 근거로 현상을 파악하고 예측하기 위한 목적으로 많이 사용되는 연구 방법으로 본 연구에서는 차에 걸쳐 리커트(Likert) 척도를 이용한 전문가 조사를 실시하였으며, 설문 결과의 타당성 및 적합성 검증 방법으로 Lawshe(1975)가 제시한 CVR(Content validity Ratio: 내용타당도 비율)을 바탕으로 분석하였다. 또한, 설문 결과의 리커트 척도 평균 M 값과 CVR 값을 내적 일관성을 측정하는 크론바흐(Cronbach) 알파를 이용하여 설문 신뢰도를 높였다(Cronbach & Shavelson, 2004).

단과대학별로 5 개로 분류된 계열은 HLL(Humanities, Lawyers, Linguists), SCI(Scientists), RPL(Resource Planners or Managers), ENG(Engineers), ART(Artists)로 Table 1 과 같다.

Table 1: Affiliated major group name of the college

Name	College
HLL	College of Humanities / Law / Education / Foreign Languages
SCI	College of Natural Science / Life and Resource / Health Sciences / Medicine, Dentistry, Pharmacy, Nursing
RPL	College of Social Sciences / Business & Economics / International / Public Service
ENG	College of Architecture / Engineering / Convergence Technology
ART	College of Arts & Design / Music / Arts / Sports Science

3 차에 걸친 델파이 조사에 참여한 전문가는 박사 수료 이상 SW 관련 업무 5 년 이상 경력이거나 석사 이상 SW 관련 업무 10 년 이상 경력을 갖춘 전문가 21 명으로 분야별 인원수는 Table 2 와 같다.

Table 2: The number of Delphi expert through the field

Field	Computer Education	Education Engineering	Software Development	Computer Engineering
Num	12	1	6	2

3.1. 1 차 전문가 설문 결과 분석

관련 문헌 및 논문 분석 등의 선행연구와 전문가 자문회의를 통해 도출한 계열별 소프트웨어 기초교육에 대한 교육 내용 영역과 알고리즘씽킹에 관한 개방형 의견에 대해 1 차 전문가 설문을 진행하였다. 내용 영역은 5 개로 나누었으며 계열별 내용타당도는 리커트 7 점 척도를 이용하였으며, 설문 시 적절성 정도가 3 점 이하인 경우 그 사유를 적도록 설문을 진행하였다. 1 차 델파이 조사 결과로는 Table 3 과 같이 내용 영역별 CVR 값과 리커트 척도의 평균(M)이 산출되었으며, 크론바흐 알파를 이용하여 타당도를 검증하였다. 응답자 수가 21 명인 경우 CVR 임계치인 0.42 보다 낮은 영역은 제외하였다.

Table 2: Content area by affiliated group of college

Content Area	HLL		SCI		RPL		ENG		ART	
	CVR	M	CVR	M	CVR	M	CVR	M	CVR	M
Overview of Algorithm	0.9	5.9	1	6.5	1	6.353	1	6.85	0.7	5.8
Algorithms and Problem Separation	0.8	5.75	1	6.45	1	6.235	1	6.85	0.6	5.55
Algorithm and Abstraction	0.7	5.4	1	6.15	1	5.882	1	6.8	0.5	5.1
Algorithm Design	~~0.2~~	5	0.8	5.85	0.7	5.235	1	6.6	~~0.1~~	4.65
Algorithm implementation	~~0.2~~	4.55	0.7	5.5	~~0.4~~	5.118	1	6.65	~~-0.052~~	4.68
Cronbach's α	0.864		0.791		0.821		0.935		0.912	

*CVR(Content Validity Ratio): 내용타당도, M: 리커트 평균

3.2. 2 차•3 차 전문가 설문 결과 분석

2 차 전문가 설문에서는 1 차 델파이 분석결과 및 소프트웨어 전문가 대상 포커스 그룹 인터뷰(FGI) 결과를 분석하고, 2 차례의 전문가 협의를 통해 계열별로 추출한 SW 기초교육 과정의 내용영역, 내용요소, 하위요소까지 제시하였으며, 설문결과의 타당성 및 적합성을 1 차와 마찬가지로 리커트 척도(7 점 척도)와 개방형 의견을 통해 검증하였다. 2 차의 경우 계열별 공통 영역의 내용 요소가 달라졌으며, 그에 따른 하위 요소도 계열별로 차이를 보였다. Table 4 는 2 차 설문의 인문 계열에서 적합한 내용영역, 내용요소, 하위요소 결과로 크론바흐 알파 신뢰도계수를 측정하여 타당성을 증명하였다.

3 차 설문을 통해서는 2 차 델파이 조사 단계까지의 결과로 각 그룹별 내용영역과 내용요소, 하위요소를 검증하고, 개방형 질문을 통해 최종적으로 Table 5 와 같이 각 계열별 적합한 내용영역과 내용요소를 도출하였다. 연구에서 제안하는 하위요소를 포함한 계열별 교육과정은 4 장 연구결과에서 기술한다.

Table 3: The result of Content area, Content element and Subcomponent by Major Group

		Algorithm concept	Algorithm Properties and Representation	Problem Finding and Defining	Problem analysis	problem division	Pattern recognition	functional abstraction	Data abstraction	Algorithm structure	Algorithm Expression	Logic	Function definition	Algorithm analysis	Programming	Simulation	Debugging	Cronbach α
HLL	CVR	0.80	0.80	0.90	0.90	0.80	0.60	0.70	0.40	0.30	0.40	0.10	-0.1	-0.2	0.30	0.00	-0.2	0.896
	M	6.05	5.55	5.90	6.00	5.85	5.25	5.05	5.00	4.80	5.00	4.50	4.00	3.75	5.00	4.15	3.90	
SCI	CVR	1.00	1.00	1.00	1.00	1.00	0.80	0.80	0.80	0.70	0.80	0.60	0.300	0.200	0.80	0.60	0.50	0.873
	M	6.65	6.60	6.70	6.70	6.55	6.05	5.85	5.75	5.60	5.65	5.45	4.75	4.75	5.90	5.25	5.30	
RPL	CVR	1.00	0.90	0.90	1.00	1.00	0.80	0.70	0.60	0.50	0.60	0.30	0.10	0.10	0.60	0.30	0.20	0.865
	M	6.30	5.90	6.00	6.25	6.25	5.80	5.35	5.45	5.00	5.10	4.90	4.30	4.00	5.35	4.45	4.55	
ENG	CVR	1.00	0.90	1.00	1.00	1.00	1.00	1.00	1.00	0.90	1.00	0.90	0.70	0.80	1.00	0.90	0.70	0.906
	M	6.85	6.50	6.85	6.85	6.85	6.65	6.55	6.55	6.20	6.50	6.40	5.65	6.10	6.65	6.30	6.15	
ART	CVR	0.90	0.70	0.90	0.80	0.80	0.50	0.60	0.40	0.10	0.20	-0.1	-0.3	-0.3	0.10	-0.1	-0.2	0.940
	M	5.95	5.45	5.80	5.75	5.65	5.10	4.90	4.70	4.40	4.80	4.30	3.85	3.50	4.60	4.00	3.75	

Table 4: The result of content element

content area	content element	Affiliated group of college				
		HLL	SCI	RPL	ENG	ART
Overview of Algorithm	Algorithm concept					
	Algorithm Properties and Representation					
Algorithms and Problem Decomposition	Problem Finding and Defining					
	Problem analysis					
	problem division					
Algorithm and Abstraction	Pattern recognition					
	functional abstraction					
	Data abstraction					
Algorithm Design	Algorithm structure					
	Algorithm Expression					
	Logic					
	Function definition					
	Algorithm analysis					
Algorithm implementation	Programming					
	Simulation					
	Debugging					

4. 연구 결과

알고리즘 기반의 소프트웨어 기초교육과정은 컴퓨터과학과 전공관련영역에서 컴퓨팅적이고 논리적인 사고력을 신장시키기 위한 활동이다. 따라서 컴퓨터의 가치를 이해하고 관심을 유도하기 위한 내용으로 구성하였다. 이는 창의성과 상상력 그리고 흥미를 유발하여 자신의 실생활과 관련한 산출물을 컴퓨터 프로그램을 활용하여 개발하는데 도움을 주도록 한다. 디자인씽킹 기반의 창의적 사고는 생산자 측면에서 학생들이 그들의 일상에서 느끼는 역동적이고 상호작용적인 것들을 만들어 내는데 필요한 지식과 실습을 강조한다. 따라서 Figure 2 와 같이 문제분해, 추상화, 알고리즘, 자료구조, 알고리즘 구현 단계를 해 학습자들이 다양한 영역에서 알고리즘적 문제해결을 경험하도록 교육과정을 설계하여 확산적 사고를 발전시키고자 한다.

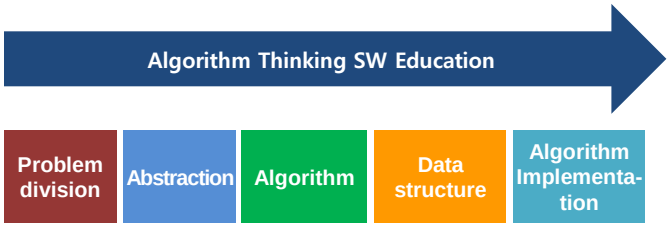

Figure 2: Algorithm Thinking Software Education

4.1. 소프트웨어 기초 교육과정 설계

본 연구에서 제안하는 비전공자를 위한 계열별 소프트웨어 기초교육과정은 3 장에서 연구한 전문가 델파이 조사와 분석을 바탕으로 설계되었으며 계열별로 Table 6-10 과 같다.

Table 6: The Curriculum for the HILL

content area	content element	subcomponent
Problem Resolution And Algorithm	Problem resolution process	General problem resolution process
		Problem resolution process on Computational thinking
	Algorithm and Procedures	Algorithm definition
		procedure for solving logical problems
	Algorithm Properties and Representation	Algorithm Features and Expression
Algorithms and Problem Decomposition	Problem Detection and Defining	Problem detection
		Problem comprehension
		problem definition
		Problem resolution planning
	Problem analysis and Representation	Current Status
		Aim Status
		Element distinction
		Condition
	Problem division	Feature of problem decomposition
		Data-centric decomposition
		Procedure-centered decomposition
		List of work
Algorithm and Abstraction	Pattern recognition	Common element detection
	abstraction of function	Core Function Extraction
		Logical procedure

Table 7: The Curriculum for the SCI

content area	content element	subcomponent
Problem Resolution And Algorithm	Problem resolution process	General problem resolution process
		Problem resolution process on Computational thinking
	Algorithm and Procedures	Algorithm definition
		procedure for solving logical problems
	Algorithm Properties and Representation	Algorithm Features
		Algorithm Technical Language
Algorithms and Problem Decomposition	Problem detection and defining	Problem detection
		Problem comprehension
		problem definition
		Problem resolution planning
	Problem analysis	Current Status
		Aim Status
		Element distinction
		Condition
	Problem division	Feature of problem decomposition
		Data-centric decomposition
		Procedure-centered decomposition
		List of work
Algorithm	Pattern recognition	Common element detection

and Abstraction	abstraction of function	Core Function Extraction
		Logical procedure
	Data of function	Core Data Extraction
		Characteristics of Data Abstraction
		Variable
		Data Type
Algorithm Design	Algorithm Structure	sequential structure
		Selective structure
		Repeated structure
		Superposition control structure
	Algorithm Expression	Algorithm representation method
		Algorithm design method
		linear structure
		Nonlinear structure
	Logic	Assignment operation
		Arithmetic operation
		Comparative operation
		Logical operation
Algorithm Implementation	Programming	Programming environment
		Variable
		Implementing the control structures
		Function implementation
		Event implementation
	Simulation	Models and Scenarios
		Simulation implementation
		Simulation Execution and Evaluation

Table 8: The Curriculum for the RPL

content area	content element	subcomponent
Problem Resolution And Algorithm	Problem resolution process	General problem resolution process
		Problem resolution process on Computational thinking
	Algorithm concept	Algorithm definition
		procedure for solving logical problems
	Algorithm Properties and Representation	Algorithm Features
		Algorithm Technical Language
Algorithms and Problem Decomposition	Problem detection and defining	Problem detection
		Problem comprehension
		problem definition
		Problem resolution planning
	Problem analysis	Current Status
		Aim Status
		Element distinction
		Condition
	Problem division	Feature of problem decomposition
		Data-centric decomposition
		Procedure-centered decomposition
		List of work
Algorithm and Abstraction	Pattern recognition	Common element detection
	abstraction of function	Core Function Extraction
		Logical procedure
	Data of function	Core Data Extraction
		Characteristics of Data Abstraction

		variable
		Data Type
Algorithm Design and Programming	Algorithm Expression	Algorithm representation method
		Algorithm design method
	Programming	Programming environment

Table 9: The Curriculum for the ENG

content area	content element	subcomponent
Problem Resolution And Algorithm	Problem resolution process	General problem resolution process
		Problem resolution process on Computational thinking
	Algorithm concept	Algorithm definition
		procedure for solving logical problems
	Algorithm Properties and Representation	Algorithm Features
		Algorithm Technical Language
Algorithms and Problem Decomposition	Problem Detection and Defining	Problem detection
		Problem comprehension
		problem definition
		Problem resolution planning
	Problem analysis	Current Status
		Aim Status
		Element distinction
		Condition
	Problem division	Feature of problem decomposition
		Data-centric decomposition
		Procedure-centered decomposition
		List of work
Algorithm and Abstraction	Pattern recognition	Common element detection
	abstraction of function	Core Function Extraction
		Logical procedure
	Data of function	Core Data Extraction
		Characteristics of Data Abstraction
		Variable
		Data Type
Algorithm Design	Algorithm Structure	Sequential structure
		Selective structure
		Repeated structure
		Superposition control structure
	Algorithm Expression	Algorithm representation method
		Algorithm design method
		linear structure
		Nonlinear structure
	Logic	Assignment operation
		Arithmetic operation
		Comparative operation
		Logical operation
	function definition	Concept of function
		Function Definition
		Parameter
		return
	Algorithm Analysis	Efficiency analysis
Algorithm Implementation	Programming	Programming environment
		Variable

		Implementing the control structures
		Function implementation
		Event implementation
	Simulation	Models and Scenarios
		Simulation implementation
		Simulation Execution and Evaluation
	Debugging	Code Error detection
		Modify Code Error

Table 10: The Curriculum for the ART

content area	content element	subcomponent
Problem Resolution And Algorithm	Problem resolution process	General problem resolution process
		Problem resolution process on Computational thinking
	Algorithm and Procedures	Algorithm definition
		procedure for solving logical problems
Algorithms and Problem Decomposition	Problem Detection and Defining	Problem detection
		Problem comprehension
		problem definition
		Problem resolution planning
	Problem analysis	Current Status
		Aim Status
		Element distinction
		Condition
	Problem division	Data-centric decomposition
		Procedure-centered decomposition
		List of work
Algorithm and Abstraction	Pattern recognition	Common element detection

5. 연구 결과 토론 및 시사점

소프트웨어 교육은 창의적인 문제 해결력을 위한 것이다. 여기서의 창의적 문제 해결력에는 문제를 해결하는 것뿐 아니라 새로운 문제를 만들어 내는 것도 포함된다. 좋은 연구 문제를 만들고 문제 해결의 가능성을 탐색하고 알고리즘 설계를 통해 구체적인 전략을 세워 문제 해결을 위한 과정을 순환적 또는 비선형적으로 반복해야 한다.

컴퓨터 비전공자를 대상으로 하는 소프트웨어 기초교육에서는 무엇보다 학습자들의 소프트웨어교육의 필요성 대한 인식 제고가 계속해서 이루어져야 한다. 소프트웨어 교육의 필요성에 대한 인식 정도가 낮으면 SW 교육의 수업에 대한 만족도, 성취감, 흥미가 떨어지기 때문에 단순히 알고리즘을 가르치는 것이 아니라 동기부여가 될 수 있는 관련 내용을 가지고 가르칠 필요가 있다.

이에 본 연구에서는 ADD Thinking 의 알고리즘씽킹을 기반으로 각 계열 학생들이 '자신의 아이디어를 컴퓨팅으로 내재화 하는 능력'을 함양하는 목표에 부합할 수 있도록 단과대학별 전공을 영역별 5 개의 그룹으로 계열을 나누고, 계열별로 내용 영역, 내용 요소, 하위 요소를 수준을 다르게 추출, 설계하였다. 3 차에 걸친 전문가 설문을 통해 요구분석 및 전공별/교과목별 특성화 전략 수립하고, 알고리즘씽킹의 향상을 위해 교육과정의 주제와 내용 구성이 각 계열에 적절한지 설계 방향의 적절성을 검증하였다.

향후 모바일 시대에 부합하는 마이크로너닝, 플립드러닝 등의 이러닝 콘텐츠에도 본 교육과정을 적용해 보는 연구가 필요하다. 또한, 학습자의 통합적인 학습경험을 고려한 사용자 중심의 학습설계를 위해 학습자의 만족도와 성취율 조사를 통한 학습 모니터링 연구를 실행(Kim & Jung, 2018)함으로써 학습자의 요구를 교육과정에 반영해야 할 것이다.

본 연구에서 제안한 교육과정은 대학의 소프트웨어 기초교육의 방향 정립에 도움을 주고, 학습자들은 전공영역에서의 문제해결 능력을 갖춘 4 차 산업에 적합한 소프트웨어 인재로 거듭날 것으로 기대된다.

References

Bennett, V., Koh, K., & Repenning, A. (2013). Computing creativity: Divergence in computational thinking. *Proceeding of the 44th ACM Technical Symposium on computer science education*, 359-364.

Cronbach, L. J., & Shavelson, R. J. (2004). My current thoughts on coefficient Alpha and Successor Procedures. *Educational and Psychological Measurement, 64*(3), 391–418.

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33-39.

Kim, D. H., & Jung, H. J. (2018). Learning process monitoring of e-learning for corporate education. *Journal of Industrial Distribution & Business*, *9*(8), 35-40.

Lawshe, C. H. (1975). A quantitative approach to content validity. *Personnel Psychology*, *28*(4), 563-575.

Lee, J. E., Kwon, S. J., & Jung, H. J. (2018). Introduction and activation strategies for smart training of corporate. *Journal of Industrial Distribution & Business, 9*(5), 83-91.

Min, G. O., & Loh, J. H. (2016). Impact of educational service information distribution on students' satisfaction and achievement rate. *Journal of Industrial Distribution & Business, 7*(4), 17-31.

Oh, K. S., & Kwon, J. I. (2019). A study on the verification of computational thinking effectiveness of understanding-oriented SW basic education program. *Journal of Digital Convergence*, *17*(10), 23-35.

Oh, K. S., Suh, E. K., & Chung, H. J. (2018). A study on development of educational contents about combining computational thinking with design thinking. *Journal of Digital Convergence*, *16*(5), 65–73.

Pi, S. Y. (2016). A study on coding education of non-computer majors for IT convergence education. *The Korea Society of Digital Policy & Management*, 1-8.

Shin, Y. H., Jung, H. J., & Suh, E. K. (2019). Analysis of learning experience in design thinking-based coding education for SW non-major college students. *Journal of Digital Contents Society*, *20*(4), 759-768.

Shon S., H. (2019). A study on the exploration of the core capabilities of design future talent against the fourth Industrial Revolution. *Journal of the Korean Society Design Culture*, *25*(2), 305-315.

Suh, E. K. (2017). Development of creative thinking and coding course method on design thinking using flipped learning. *Korean Association for Learner-Centered Curriculum and Instruction*, *17*(16), 173-199.

Suh, E. K., Oh, K. S., & Chung, H. J. (2018). *Creative thinking and coding(Engineering).* Yongin-si, Korea: Nosvos.

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

Wang, Y., Widrow, B., Zadeh, L. A., Howard, N., Wood, S., Bhavsar, V. C., Budin, G., Chan, C., Fiorini, R. A., Gavrilova, M. L., & Shell, D. F. (2016). Cognitive intelligence: Deep learning, thinking, and reasoning by brain-inspired systems. *International Journal of Cognitive Informatics and Natural Intelligence*, *10*(4), 1-20.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Yoon, O. H. (2017). A study of the instructional systems design model for STEAM education: Focus on design thinking. *Korean Journal of General Education*, *11*(1), 443-474.