http://www.jsebs.org
ISSN: 2288-3908

The Journal of Society for e-Business Studies
Vol.24, No.4, November 2019, pp.17-30
https://doi.org/10.7838/jsebs.2019.24.4.017

계열별 학습자 분석 기반의 컴퓨팅사고력 연구

Research of Computational Thinking based on Analyzed in Each Major Learner

권정인(Jungin Kwon)*

초 록

소프트중심의 급속한 사회변화로 인해 모든 학문분야 인재상의 기본 조건으로 소프트웨어 역량의 중요성이 강조되고 있다. 본 연구는 현재 대학에서 신입생을 대상으로 실시되고 있는 기초소프트웨어교육의 계열별 학습자들의 소프트웨어교육 인식 차이를 조사하고자 한다. 연구결과 문제해결을 위한 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소 적용에 계열별 학습자들은 다음과 같은 차이를 보였다. 인문계열 학습자들은 자료수집, 문제분해, 자동화의 요소를 주로 문제해결과정에 적용하는 반면, 자연계열 학습자들은 자료분석, 알고리즘, 자동화의 요소를 주로 적용하였다. 또한, 예술계열 학습자들은 자료표현, 추상화, 자동화의 요소를 주로 적용하였다. 소프트웨어 개발에 컴퓨팅사고력(Computational Thinking)을 적용할 때 역시 인문계열 학습자들은 자료수집, 알고리즘, 자동화의 요소를 주로 적용하는 반면, 자연계열 학습자들은 자료분석, 알고리즘, 자동화의 요소를 주로 적용하였다. 또한, 예술계열 학습자들은 자료표현, 추상화, 시뮬레이션의 요소를 주로 적용하였다. 본 연구의 결과를 토대로 각 대학에서 실시하고 있는 기초소프트웨어교육 과정의 설계에 계열별 학습자 분석이 반드시 포함되어 학습자의 교육적 효용성이 극대화되기를 바란다.

ABSTRACT

The rapid development of a software-core society emphasizes the importance of software competence as a basic condition for all academic disciplines. The purpose of this study is to investigate the difference of perceptions among students of basic software education which is currently being conducted in university. The results of applying the nine core elements of Computational Thinking for Problem Solving to the learners of the each majors are as follows. In humanities, learners mainly applied the elements of Data Collection, Problem Decomposition and Automation. On the other hand, natural science department learners mainly applied the elements of Data Analysis, Algorithm and Automation. In addition, arts learners mainly applied elements of Data Representation, Abstraction, and Automation. To apply Computational Thinking to the development of software, humanities learners mainly applied elements of Data Collection, Algorithm, Automation. On the other hand, natural science department learners mainly applied the elements of Data Analysis, Algorithm and Automation. In addition, arts learners mainly applied elements of Data Representation, Abstraction, and Automation. Based on the results of this study, it is expected that the educational effectiveness of the learner will be maximized by including the learner analysis with each majors in the design of the basic software curriculum that each university is conducting.

키워드 : 소프트웨어교육, 컴퓨팅사고력, 컴퓨팅사고력 핵심요소, 교양교육
Software Education, Computational Thinking, Core Elements of Computational Thinking Basic Software Curriculum, Liberal Arts Education

This research was funded by a 2018 research Grant from Sangmyung University.
* College of kyedang General Education, Sangmyung University(jikwon@smu.ac.kr)
Received: 2019-07-29, Review completed: 2019-10-15, Accepted: 2019-10-21

1. 서 론

2016년 다보스 포럼을 통해 클라우드 슈밥이 선언한 제4차 산업혁명이 화두가 되면서 세계는 지금 소프트웨어를 중심으로 한 융합과 혁명에 초점을 맞추고 있다[13]. 이는 소프트웨어와 산업분야의 융합뿐만 아니라 인간과의 융합까지도 포괄적으로 의미하면서 새로운 시대를 이끌어갈 혁명적 창의・융복합 인재 양성에 촉각을 세우고 있다[3]. 이 같은 추세는 전체 교육영역의 인재상에도 변화를 요구하고 있어 제4차 산업혁명시대를 주도할 창의・융복합적 소프트웨어인재양성의 중요성이 대두되고 있다[2, 15]. 우리나라에서도 이러한 시대적 흐름을 반영하고자 2018년도부터 소프트웨어교육이 의무화되고 정부 차원에서 소프트웨어 교육을 적극 장려하고 있다. 초중등학교에서 '정보'교과를 필수로 지정하는가 하면 대학에 입학하는 신입생을 대상으로 기초교양필수 교과목으로 기초소프트웨어교육이 실시되고 있다[7]. 그러나, 이러한 시대적 변화를 교육에 반영함에도 불구하고, 대학에서 기초교양필수 교과목으로 운영되고 있는 기초소프트웨어교육에 대한 학습자와 교수자의 불만은 끊이지 않고 나타나고 있다[15]. 우선 학습자 입장에서는 기초소프트웨어교육이 어렵다는 의견과 본인 전공과의 연계성부분에 의구심을 갖는 경우가 많다. 현재 대학에 입학하는 신입생은 초중등학교의 정규 교육과정을 통해 정보교과를 이수하지 않은 채 대학에 입학하는 학습자들이 대부분이다. 또한 대학 입학을 통해 본인의 관심분야에 맞춰 학습을 하고자 하는 학습자의 요구와 다르게 대부분의 대학에서 소프트웨어교육을 기초교양필수로 지정하면서 이들의 불만이 커지고 있다. 교수자 입장에서도 컴퓨팅사고력(Computational Thinking)을 기반으로 한 기초소프트웨어교육을 어떻게 현장에 적용해야 하는지에 대한 체계적인 연구가 시행되고 있지 않은 상태에서 불만이 가득한 학습자와 4차 산업혁명이나, 창의・융복합을 논하기란 여간 어려운 일이 아니다. 또한 일부 대학에서는 기초소프트웨어교육이 단순 프로그래밍 언어 수강자의 확대라는 생각으로 과거 전공자에게 사용했던 교육자료 중 일부를 수정하는 방식으로 수업을 진행하고 있거나, 컴퓨팅사고력(Computational Thinking)의 적용은 버려둔 채 프로그래밍 언어의 개념만을 위주로 기초소프트웨어교육을 실시하고 있어 시대적 변화에 따른 창의・융복합적 인재를 양성하기 위한 기초소프트웨어교육은 아직까지도 보강해야 할 부분이 많다. 학습자 입장에서는 전공과의 연계성 여부에 대한 의구심과 불만을 최소화할 수 있어야 하고, 교수자 입장에서도 기초소프트웨어교육에 컴퓨팅사고력(Computational Thinking)을 어떤 방법으로 적용시킬 수 있는지에 대한 지속적 연구가 필요하다. 따라서 현재 대학의 기초소프트웨어교육은 교수자와 학습자 모두를 위한 체계적인 교육과정뿐만 아니라 계열별 전공과의 연계성을 높일 수 있는 기초소프트웨어교육의 개발을 필요로 하고 있다.

본 연구는 현재 대학에서 신입생을 대상으로 실행되고 있는 기초소프트웨어교육의 계열별 소프트교육에 대한 학습자 인식의 차이를 조사하고자한다. 계열은 크게 인문계열과, 자연계열, 예술계열로 구분하고, 이들이 대학 입학 시 소프트웨어교육의 정도와 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소를 소프트웨어교육에 적용할 때 나타나는 인식의 차이를 분석하고자 한다.

2. 관련연구

2.1 컴퓨팅사고력(Computational Thinking)과 초중등 정보 교육과정

CSTA와 ISTE(International Society for Technology in Education)에서는 Bar et al.[3]의 연구결과를 토대로 컴퓨팅사고력(Computational Thinking)의 9가지의 핵심요소는 <Table 1>과 같이 제시하고 있다[1, 4, 12, 16, 17].

〈Table 1〉 Computational Thinking Main Concepts

Main Concept	Definition
Data Collection	Problem understanding, analysis and collect data based on analysis to solve the problem
Data Analysis	Carefully sorting and analyzing the data collected and data provided in the problem
Data Representation	Express data in problem using graphs, charts, words and images
Problem Decomposition	Dividing and analyzing the problem to solve the problem
Abstraction	Defining the main concepts to reduce the complexity of the problem
Algorithm and Procedures	Expressing the steps required to solve the problem until now
Automation	Creating an algorithm of the solution procedure for a computing machine to carry it out
Simulation	Creating an experimental model to solve the problem
Parallelization	Coming up with a common objective to solve a problem

우리나라 교육부에서도 2015년 9월에 '2015 개정 교육과정 총론'을 발표하면서 소프트웨어 교육의 목적은 컴퓨팅사고력(Computational Thinking)을 기반으로 문제해결역량을 함양하는데 있다고 하였다[7]. 정보교과 내 컴퓨팅 사고력을 영역은 다음의 <Table 2>에서 보는 바와 같이 자료와 정보 영역, 문제해결과 프로그래밍 영역이다[7, 9].

〈Table 2〉 Information Curriculum and Computational Thinking〔7〕

Category	Main idea	Generalized knowledge	Computational Thinking
Data & Information	Data & Information Representation	Digitize analog data such as numbers, letters, pictures, sounds, etc.	Data Collection Data Representation
	Data & Information of Analysis	Structuring data collection and data analysis required for problem solving	Data Analysis
Problem Solving & Programming	Abstraction	Abstraction is the process of eliminating unnecessary elements for problem solving and problem decomposition	Abstraction
	Algorithm	Algorithm is an efficient method and procedure for problem solving.	Algorithm and Procedures
	Programming	Programming implements Problem Solving as a programming language	Automation

이처럼 기존의 연구논문이나 정책보고서에서는 컴퓨팅사고력(Computational Thinking)의 요소나 교육과정에만 소개되어 있을 뿐 정작 교육의 수혜대상인 학습자의 특성분석에 대한 연구나 보고서는 전문한 상태이다.

2.2 대학의 컴퓨팅사고를 반영한 소프트웨어 교육 현황 분석

대학의 컴퓨팅사고력을 반영한 교육과정을 살펴보면 다음과 같이 정리할 수 있다.

경기도 소재 Da대학은 2016년부터 창의적 사고와 문제해결력 향상을 위한 프로그램 구현을 위해 교재를 개발하여 기초소프트웨어교육을 실시하고 있다. 창의적사고와 코딩에서는 컴퓨팅사고력(Computational Thinking)을 계열별로 나누어 스크래치, 파이선, Blockly 등의 프로그래밍 언어를 교육하고 있다[14]. 서울소재 Ko대학은 2015년부터 엑셀, 워드, 스크래치, 엔트리, 파이썬을 활용한 비전공자 소프트웨어 교육을 실시해 오고 있다. OA교과목의 자격증 취득과 스크래치와 엔트리를 통해 컴퓨팅사고력(Computational Thinking)의 의미를 이해하고 예제를 이용하여 프로그래밍을 경험적으로 습득하도록 하는데 교육에 목적을 두고 있다[10]. 서울소재 Su대학은 2016년부터 컴퓨팅 사고력 기반의 문제해결 능력 함양을 위한 소프트웨어 교육을 실시하고 있다. '컴퓨팅사고와 SW코딩'은 엔트리 프로그램을 기반으로 '문제해결과 알고리즘' 파이썬을 기반으로 한 소프트웨어 교육을 실시하고 있다[8]. 서울소재 So대학은 2016년부터 '컴퓨팅적사고' 교과를 통해 스크래치, 파이썬, 엔트리, 앱인벤터를 활용한 소프트웨어 교육을 실시해 오고 있다. 교과내용은 컴퓨팅 사고의 요소와 절차, 이를 활용한 문제해결에 초점을 두고 있으며 오프라인 수업은 스크래치와 파이썬을 사용하여 실습하도록 구성되어 있다[6].

지금까지 발표된 관련연구를 분석해 보면 대부분의 대학에서는 컴퓨팅사고력(Computational Thinking)기반 문제해결의 중요성을 인식하여 컴퓨팅사고력(Computational Thinking)을 반영한 기초소프트교육을 설계하고 실행하고 있다. 하지만, 이는 교육을 제공하는 입장에서 바라본 교육설계 일뿐 실제 교육을 필요로 하는 학습자의 요구사항은 그 어디에도 찾을 수가 없다. 또한 학습자 분석을 기반으로 한 소프트웨어교육의 설계는 지금까지 연구에서는 전무한 상태이다. 2018년도부터 소프트웨어 교육이 초중등교육과정 내 필수로 지정되었기 때문에 현재 대학에 입학하는 학습자의 대부분은 소프트웨어교육에 대한 선수지식에 다양한 차이를 갖고 있다. 대학에서 제공되고 있는 기초소프트웨어교육을 학습자 대부분이 필요하다고는 느끼나 그 내용과 정확한 교육적 취지를 이해하기는 어렵다는 의견이 대부분이다[5, 11]. 지금 대학의 기초소프트웨어교육은 학습자의 요구사항에 대한 분석을 토대로 소프트웨어교육에 컴퓨팅사고력(Computational Thinking)을 어떻게 적용시킬 수 있는지에 대한 지속적인 연구와 정책이 필요한 시점이다

따라서, 본 연구는 계열별 학습자가 갖는 특성을 분석하기 위해 컴퓨팅사고력(Computational Thinking)의 인식 차이를 설문 조사하고자 한다. 수집된 자료는 기초통계를 중심으로 분석하여 기초소프트교육모델 설계에 계열별 방향성을 제시하고자 한다.

3. 연구방법 및 절차

3.1 연구대상 설정

본 연구의 검증을 위해 서울 소재 S대학에 입학하는 신입생 321명을 대상으로 입학 전 소프트웨어교육에 관한 선수학습 설문조사, 컴퓨팅사고력(Computational Thinking)의 개념과 핵심요소를 학습한 후 문제해결에 적용하는 방법의 계열별 차이에 대한 설문조사 그리고 컴퓨팅사고력(Computational Thinking)를 실제 소프트웨어개발에 적용할 때 핵심이 되는 요소를 설문조사하였다. 3차에 걸친 설문결과를 취합 한 후 통계적 분석이 가능한 300명의 자료만을 추려내어 계열별 학습자의 특성을 분석하였다. 본 연구의 설문분석에 참여한 300명은 전공분야별로 인문계열, 자연계열, 예술계열로 구분하였다. 계열별 구분 중 인문계열은 영어, 지적재산, 교육학 등의 전공으로 구성되어져 있고, 자연계열은, 수학, 화학공학, 생명공학 등을 포함하고 있으며, 예술계는, 미술, 무용, 음악을 전공하고 있었다. 연구대상의 구분에 따른 특징은 <Table 3>과 같다.

〈Table 3〉 Characteristics of Research Subjects

Dept.	Sex(Number of people		Total Number of people	pro-portion
Humanities	Male	54	133	44%
	Female	79		
Natural Science	Male	61	120	40%
	Female	59		
Art	Male	12	47	16%
	Female	35		

3.2 연구 방법 및 절차

계열별 학습자 특성을 분석하기 위해 신입생들의 소프트웨어교육에 대한 선수학습의 정도를 측정하고자 설문조사를 실시하였다. 이후 기초소프트웨어교육 15주간 중 2~5주차까지는 컴퓨팅사고력(Computational Thinking)에 관한 수업을 실시한 후 계열별 학습자들의 소프트웨어교육의 이해정도와 인식의 차이를 측정하였고, 9~13주차 수업을 실시하여 컴퓨팅사고력(Computational Thinking)을 실제 소프트웨어개발에 적용할 때 중요하게 생각하는 컴퓨팅사고력(Computational Thinking)의 요소에 대한 설문 조사를 실시하였다.

3.2.1 선수학습 사전설문실시

계열별 학습자들의 소프트웨어교육에 대한 선수학습의 정도를 측정하고자 총 15개의 설문문항을 구성하였다. 설문문항은 소프트웨어교육의 경험 유무와 교육기간, 교육기관, 교육내용에 관한 부분으로 구분하여 실시하였다. 작성된 설문 문항은 중고등학교 교사의 검증을 걸쳐 기초소프트웨어 교육이 시작되기 전 1주차에 실시하였다.

3.2.2 기초소프트웨어교육 내 컴퓨팅사고력 적용 교육설계

서울 소재 S대학 기초소프트웨어 교과목 내 15주차 교육설계과정을 컴퓨팅사고력(Computational Thinking)의 개념 이해와 적용, 구현의 3단계와 매핑하였다[2, 12]. 이해과 적용단계인 2~5주차에는 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소 개념 및 실제 문제상황에서의 적용과 이해를 학습하였고, 9~13

주차에는 실제 소프트웨어를 개발하는 과정과 절차에서 컴퓨팅사고력(Computational Thinking)을 적용하는 교육을 실시하였다. 컴퓨팅사고력(Computational Thinking) 적용 교육설계는 <Table 4>와 같다.

〈Table 4〉 Computing Thinking Applied in Education Design

Week	Curriculum	Step
2	We live in SW World	Understanding Concepts
3	Concepts of Computational Thinking	
4	Applying Computational Thinking	Application
5	Computational Thinking and Problem Solving	
9	Data Representation and Memory	Software Development
10	Python Control Statements	
11	Python Selection	
12	Python Loop	
13	Python Functions	

3.2.3 컴퓨팅사고력의 핵심요소 설문조사

계열별 학습자들의 컴퓨팅사고에 대한 이해 정도와 인식의 차이를 측정하고자 총 30문항의 설문문항을 구성하였다. 설문문항은 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소 중 문제해결을 위해 가장 중요하다고 생각되는 요소에 대한 설문을 실시하였다. 15주간의 수업 중 2~5주차 사이에 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소를 학습하고 6주차에 설문을 실시하였다.

3.2.4 적용 설문조사

계열별 학습자들의 컴퓨팅사고력(Computational Thinking)기반 소프트웨어개발을 할 때 중요하게 생각하는 요소에 대한 설문을 측정하고자 총 20문항의 설문문항을 구성하였다. 계열별 학습자들의 컴퓨팅사고력(Computational Thinking)에 대한 이해 정도와 인식의 차이를 측정하기 위해 수업 중에 학습한 컴퓨팅사고력(Computational Thinking)을 기반으로 팀프로젝트를 구현하는 과정에서 가장 중요하게 생각하는 컴퓨팅사고의 요소를 측정하였다. 15주차의 수업 중 9~13주차 사이에 컴퓨팅사고력(Computational Thinking)을 적용한 팀프로젝트 수업을 실시하고, 이후 14주차에 설문을 실시하였다.

4. 소프트웨어교육의 계열별 인식 차이

4.1 사전설문실시 결과

계열별 학습자들의 소프트웨어교육에 대한 선수학습의 정도를 분석한 결과는 <Table 5>와 같다. 소프트웨어교육의 경험이 있습니까? 라는 질문에 전체 17%의 학습자만 소프트웨어 교육 경험이 있다고 답했으며, 나머지는 소프트웨어 교육의 경험이 없다고 답했다. 소프트웨어 교육을 받은 경험이 있는 51명 학습자 중 초중등학교의 학교교육을 통해서 교육을 받은 학습자는 전체 71%였으며, 이들은 주로 초등학교에서 방과 후 활동으로 소프트웨어 교육을 받았다. 계열별로는 자연계열 학습자들이 입학 전 소프트웨어교육의 경험이 가장 많았다. 그 밖에 학교 이외의 장소는 사설학원, 온라인학습, 개인과외 등을 포함한 결과이다. 교육내용

〈Table 5〉 Preliminary Survey Results of Subjects

Item	Division		Humanitie	Natural Science	Art	Total Number of people	Proportion
experience	have experience		12	37	2	51	17%
	don't have experience		121	83	45	249	83%
organization	School	Elementary	9	20	2	31	61%
		Middle	1	3	-	4	8%
		High	-	1	-	1	2%
	Non-school		2	13	-	15	29%
contents	Blocked language		2	10	-	12	24%
	Text Language		1	12	-	13	25%
	OA		9	15	2	26	51%

에 관해서는 OA교육을 받았다는 학습자가 가장 많았으며, 프로그램 언어로 C언어와 블록형 언어인 스크래치를 가장 많이 알고 있었다.

4.2 문제해결과정과 컴퓨팅사고력 관계

<Table 4>에서 제시한 것과 같이 컴퓨팅사고력(Computational Thinking)을 적용한 교육설계 중 2~5주차에는 개념과 적용을 학습한다. 학습자들은 컴퓨팅사고력(Computational Thinking)을 적용하여 문제를 해결하기 위한 과정으로 문제 분석, 설계, 구현의 단계가 있음을 학습하였다. 5주차 수업을 마치고, 문제를 해결하는 과정에 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소를 적용할 때 각 단계별 꼭 필요하다고 생각되는 요소를 순서대로 나열하시오라는 질문에 인문계는 분석, 설계, 구현 순으로 자연계는 설계, 구현, 분석 순으로 예술계는 구현, 설계, 분석 순으로 나타났다. 자세한 내용은 <Table 6>과 같다.

또한 분석의 단계 중 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소 중 문제의 분석을 위해 가장 중요한 순으로 나열하시오라는 질문에 인문계(n = 133)는 자료수집(67명), 표현(43명), 분석(23명)의 순으로, 자연계(n = 120)는 자료분석(71명), 수집(34명), 표현(15명)의 순으로, 예술계(n = 47)는 자료표현(21명), 수집(15명), 분석(11명)의 순으로 나타났다. 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소 중 문제 해결의 설계를 위해 가장 중요한 요소를 순서대로 나열하시오라는 질문에 인문계(n = 133)는 문제분해(74명), 알고리즘(41명), 추상화(18명)의 순으로, 자연계(n = 120)는 알고리즘(50명), 추상화(41명), 문제분해(29명)의 순으로, 예술계(n = 47)는 추상화(19명), 문제분해(15명), 알고리즘(13명) 순으로 나타났다. 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소 중 소프트웨어의 구현을 위해 가장 중요한 순으로 나열하시오라는 질문에 인문계(n = 133)는 자동화(80명), 시뮬레이션(41명), 병렬화(12명)의 순으로, 자연계(n = 120)는 자동화(87명), 병렬화(20명), 시뮬레이션(13명)의 순으로, 예술계(n = 47)는 자동화(27명), 시뮬레이션(15명), 병렬화(5명) 순으로 나타났다.

〈Table 6〉 Main Factor of Problem Solving Procedure and Computational Thinking

Item	Nine Key Elements of Computational Thinking	Humanities n = 133	Natural Science n = 120	Art n = 47
Analysis	Data Collection	50.4%(67people)	28.3%(34people)	31.9%(15people)
	Data Analysis	17.3%(23people)	59.2%(71people)	23.4%(11people)
	Data Representation	32.3%(43people)	12.5%(15people)	44.7%(21people)
Design	Problem Decomposition	55.7%(74people)	24.2%(29people)	32.0%(15people)
	Abstraction	13.5%(18people)	34.2%(41people)	40.4%(19people)
	Algorithm and Procedures	30.8%(41people)	41.6%(50people)	27.6%(13people)
Development	Automation	60.2%(80people)	72.5%(87people)	57.5%(27people)
	Simulation	30.8%(41people)	10.8%(13people)	31.9%(15people)
	Parallelization	9.0%(12people)	16.7%(20people)	10.6%(5people)

4.3 소프트웨어개발과 컴퓨팅사고력 관계

<Table 4>에서 제시한 것과 같이 컴퓨팅사고력(Computational Thinking) 적용 교육설계 내 9~13주차에는 컴퓨팅사고력(Computational Thinking)를 실제 소프트웨어교육에 적용하기 위한 교육에서 학습자들은 팀프로젝트를 통해 컴퓨팅사고력(Computational Thinking)를 적용한 소프트웨어개발을 실시하였다.

13주차 수업을 마치고 컴퓨팅사고력(Computational Thinking)을 실제 소프트웨어개발에 적용할 때 꼭 필요하다고 생각되는 요소를 순서대로 나열하시오 라는 질문에 인문계는 분석, 구현, 설계 순으로 자연계는 분석, 설계, 구현 순으로 예술계는 설계, 구현, 분석 순으로 나타나 컴퓨팅사고력(Computational Thinking)을 문제해결과정에 매핑 할 때와는 결과의 차이를 보였다. 각 단계별 세부항목에 대한 인식의 차이는 <Table 7>에서 보는 바와 같다. 또한 분석의 단계 중 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소 중 소프트웨어개발을 위한 문제 분석을 위해 가장 중요한 순으로 나열하시오라는 질문에 인문계(n = 133)는 자료수집(85명), 표현(43명), 분석(5명)의 순으로, 자연계(n = 120)는 분석(84명), 수집(26명), 표현(10명)의 순으로, 예술계(n = 47)는 표현(25명), 수집(19명), 분석(3명)의 순으로 나타났다.

컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소 중 소프트웨어개발을 위한 문제 설계를 위해 가장 중요한 요소를 순서대로 나열하시오라는 질문에 인문계(n = 133)는 알고리즘(61명), 문제분해(60명), 추상화(12명)의 순으로, 자연계(n = 120)는 알고리즘(57명), 추상화(52명), 문제분해(11명)의 순으로, 예술계(n = 47)는 추상화(22명), 알고리즘(20명), 문제분해(5명) 순으로 나타났다. 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소 중 소프트웨어개발을 위해 가장 중요한 순으로 나열하시오라는 질문에 인문계(n = 133)는 자동화(75명), 시뮬레이션 (52명), 병렬화(6명)의 순으로, 자연계(n = 120)는 자동화(64명), 병렬화(42명), 시뮬레이션(14명)의 순으로, 예술계(n = 47)는 시뮬레이션(20명), 자동화(14명), 병렬화(13명) 순으로 나타났다.

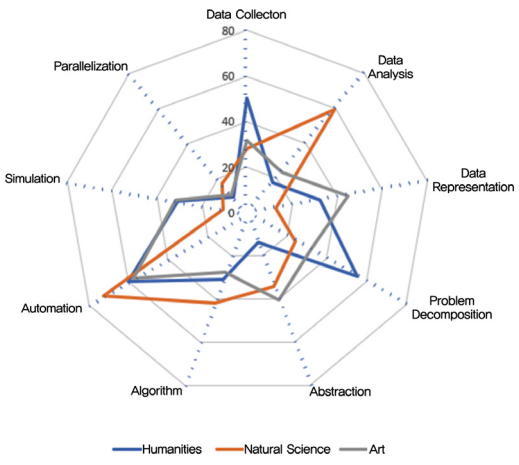
4.4 계열별 인식의 차이

계열별 학습자들의 컴퓨팅사고력(Computational Thinking)에 대한 이해 정도와 인식의 차이를 측정한 결과 <Figure 1>과 같은 결과를 도출할 수 있었다. 분석단계에서 인문계열 학습자는 문제해결에 가장 핵심이 되는 컴퓨팅사고력(Computational Thinking) 요소로 자료수집을, 자연계열 학습자는 자료분석을, 예술계열 학습자는 자료표현을 선택하였다. 분석단계에서 가장 영향력이 없는 요소로 인문계열은 자료분석을, 자연계열은 자료표현을, 예술계에서는 자료분석을 선택하였다. 설계단계에서 인문계열 학습자는 문제해결에 가장 핵심이 되는 컴퓨팅사고력(Computational Thinking) 요소로 문제분해를, 자연계열 학습자는 알고리즘을, 예술계열 학습자는 추상화를 꼽았다. 설계단계에서 가장 영향력이 없는 요소로 인문계열은 추상화를, 자연계열은 문제분해를, 예술계에서는 알고리즘을 선택하였다. 구현단계에서 인문계열 자연계열 예술계열 학습자 모두가 문제해결에 가장 핵심이 되는 컴퓨팅사고력(Computational Thinking) 요소로 자동화를 꼽았다. 설계단계에서 가장 영향력이 없는 요소로 인문계열과 예술계는 병렬화를, 자연계열은 시뮬레이션을 선택하였다.

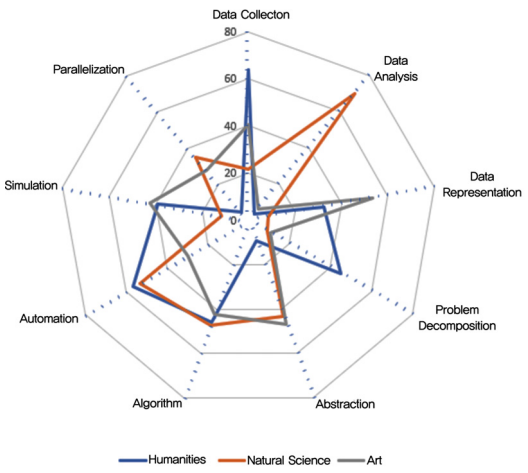
계열별 학습자들의 소프트웨어개발에 핵심이 되는 컴퓨팅사고력(Computational Thinking)의 요소를 실제 소프트웨어를 개발하고 난 후 측정한 결과는 <Figure 2>와 같은 결과를 도출할 수 있었다. 분석단계에서 인문계열 학습자는 문제해결에 가장 핵심이 되는 컴퓨팅사고력(Computational Thinking) 요소로 자료수집, 자연계열 학습자는 자료분석, 예술계열 학습자는 자료표현을 선택했다. 분석단계에서 가장 영향력이 없는 요소로 인문계열은 자료분석, 자연계열은 자료표현, 예술계에서는 자료분석 선택하였다. 소프트웨어개발의 분석단계 결과는 계열별 학습자들이 컴퓨팅사고력(Computational Thinking)을 문제해결과정과 연계한 결과와 같은 결과가 도출되었다. 설계단계에서 인문계열 학습자는 문제해결에 가장 핵심이 되는 컴퓨팅사고력(Computational Thinking) 요소로 알고리즘과 문제분해를, 자연계열 학습자는

〈Table 7〉 Development of SW and Computational Thinking

Item	Nine Key Elements of Computational Thinking	Humanities n = 133	Natural Science n = 120	Art n = 47
Analysis	Data Collection	63.9%(85people)	21.7%(26people)	40.4%(19people)
	Data Analysis	3.8%(5people)	70.0%(84people)	6.4%(3people)
	Data Representation	32.3%(43people)	8.3%(10people)	53.2%(25people)
Design	Problem Decomposition	45.1%(60people)	9.2%(11people)	10.6%(5people)
	Abstraction	9.0%(12people)	43.3%(52people)	46.8%(22people)
	Algorithm and Procedures	45.9%(61people)	47.5%(57people)	42.6%(20people)
Development	Automation	56.4%(75people)	53.3%(64people)	29.8%(14people)
	Simulation	39.1%(52people)	11.7%(14people)	42.6%(20people)
	Parallelization	4.5%(6people)	35.0%(42people)	27.6%(13people)



〈Figure 1〉 A Computational Thinking Factor at the Main of Problem Solving



〈Figure 2〉 A Computational Thinking Factor at the Main of SW Development

추상화와 알고리즘을, 예술계열 학습자는 추상화와 알고리즘을 선택하였다. 설계단계에서 가장 영향력이 없는 요소로 인문계열은 추상화를, 자연계열과 예술계열은 문제분해 선택하였다. 설계단계에서는 계열별 학습자들이 컴퓨팅사고력(Computational Thinking)이 필요한 요소와 그렇지 않은 요소들 간에 구분이 명확해졌다. 또한 예술계열 학습자들은 컴퓨팅사고력(Computational Thinking)의 알고리즘이 문제해결과정에서는 영향력이 없다고 답했으나, 실제 소프트웨어개발 단계에서는 알고리즘이 핵심이 되는 요소라고 변화된 결과를 도출했다. 구현단계에서 인문계열 자연계열 학습자는 문제해결에 가장 핵심이 되는 컴퓨팅사고력(Computational Thinking) 요소로 자동화를 꼽았으나, 예술계 학습자들은 시뮬레이션을 꼽았다. 설계단계에서 가장 영향력이 없는 요소로 인문계열은 병렬화를, 자연계열은 시뮬레이션을 선택하였고, 예술계의 경우에는 자동화와 병렬화가 비슷한 수준으로 나타났다.

5. 결 론

제4차 산업혁명이 화두가 되면서 세계는 지금 소프트웨어 중심의 융합과 혁명에 초점을 맞추고 있다. 과거 특정학문분야 중심의 소프트웨어교육에서 벗어나 이제는 모든 사람이 읽고, 쓰고, 계산하는 것과 마찬가지로 소프트웨어를 배워야 한다고 하고 있다[3, 9]. 교육분야에서도 이러한 시대적 흐름을 반영하고자 소프트웨어 중심의 창의·융복합적 인재 양성에 교육의 방향을 맞추고 있으며 소프트웨어 중심의 교육을 최근 시행하고 있다. 최근 몇 년 사이 대학에 입학하는 신입생들은 모두 기초소프트웨어교육의 일환으로 컴퓨팅사고력(Computational Thinking)기반 소프트웨어교육을 받고 있다. 과거에 프로그래밍이나 툴 중심의 소

프트웨어교육이 아닌 이해와 절차를 중심으로 한 사고력 중심의 소프트웨어교육이 필요한 시점이다. 그러나 현재 대학에서 실시되고 있는 컴퓨팅사고력(Computational Thinking) 기반 기초소프트웨어교육은 대부분의 교육과정이나 교수방법, 교육내용이 교수자 입장으로 연구 되고 있을 뿐 정작 교육의 수해자인 학습자에 대한 분석이 전무한 상태이다. 더구나, 현재 대학에 입학하는 대부분의 신입생들은 초중등 교육과정에 정보 혹은 소프트웨어에 대한 교육을 전혀 받지 못한 경우가 대부분이다. 이런 학습자들에게 이해와 절차를 중심으로 한 사고력 수업은 'ㄱ', 'ㄴ'을 모르는 학습자에게 소설 쓰는 방법을 설명하는 것과 같다. 기초소프트웨어교육이 제대로 설계되어 학습자에게 효과성이 발휘되기 위해서는 체계적인 교육과정, 교수법, 교육내용 보다 먼저 설계된 교육을 이수해야 하는 학습자 분석이 반드시 우선시 되어야 한다.

따라서, 본 논문은 현행 대학에서 실시하고 있는 기초소프트웨어교육의 문제점을 학습자의 이해 측면에서 연구하기 위해 기초소프트웨어 교육을 이수하는 학습자 분석을 실시하였다. 학습자를 크게 3개의 계열(인문계열, 자연계열, 예술계열)로 구분하여 학습자들이 소프트웨어 교육을 받으면서 나타나는 특징을 분석하였다. 그 결과 다음과 같은 특징을 찾을 수 있었다.

첫째, 현재 대학에 입학하는 신입생들의 입학전 소프트웨어교육의 경험은 상당한 차이를 보인다. 2018년부터 초중등교육과정에 정보교과가 필수로 지정되면서, 현재 대학에 입학하는 신입생들의 80% 정도는 초중등교육과정에서 소프트웨어교육을 전혀 경험하지 못한 채 입학하는 경우가 대부분이다. 대학에서는 이들을 위한 기초용어정리 및 개념을 학습할 수 있는 충분히 내용과 시간을 교육과정에 고려해야 한다.

둘째, 문제해결을 위한 컴퓨팅사고력(Computational Thinking)의 개념 및 이해를 적용할 때 계열별 학습자의 차이가 분명함을 도출하였다. 인문계열 학습자들은 문제해결을 이해하기 위해 컴퓨팅사고의 개념 중 자료수집, 문제분해, 자동화의 요소를 주로 적용하는 반면, 자연계열 학습자들은 자료분석, 알고리즘, 자동화의 요소를 주로 적용하였다. 또한, 예술계열학습자들은 자료표현, 추상화, 시뮬레이션의 요소를 주로 적용하였다.

셋째, 소프트웨어를 개발하기 위한 컴퓨팅사고력(Computational Thinking)의 적용 단계에서 계열별 학습자의 차이가 분명함을 도출하였다. 인문계열 학습자들은 문제해결을 이해하기 위해 컴퓨팅사고의 개념 중 자료수집, 알고리즘, 문제분해, 자동화의 요소를 주로 적용하는 반면, 자연계열 학습자들은 자료분석, 알고리즘, 추상화, 자동화의 요소를 주로 적용하였다. 또한, 예술계열학습자들은 자료표현, 추상화, 시뮬레이션의 요소를 주로 적용하였다.

위의 내용을 토대로 계열별 학습자의 특성을 고려하여 내용요소에 구성할 수 있다. 인문계열 학습자에게는 자료수집, 문제분해, 자동화를 중심으로 자연계열학습자에게는 자료분석, 알고리즘, 자동화를 중심으로 예술계열학습자에게는 자료표현, 추상화, 시뮬레이션 등의 내용요소를 중심으로 교육요소를 구성했을 때 컴퓨팅사고력(Computational Thinking)에 대한 학습자의 이해도를 높일 수 있다는 결론을 얻었다.

본 연구는 다만 다음과 같은 한계점을 가진다. 첫째 본 연구에서 사용된 자료는 설문조사를 기반으로 표본 자체의 속성을 파악하는 기술통

계 중심으로 기술함으로써 자료의 분석을 기반으로 한 추측통계로 확대하기에는 다소 무리가 있다. 둘째, 실험에 참여한 학습자는 특정대학의 소속된 학습자들로만 구성되어 있어 전체로 확대하기에는 무리가 있다. 이 같은 한계점은 지속적인 후속 연구를 통해 보완해 나가야 할 것이다.

본 연구는 현재 대학에서 신입생을 대상으로 실행되고 있는 기초소프트웨어교육의 계열별 소프트교육에 대한 학습자들의 인식의 차이를 조사하고자 실시하였다. 계열별 학습자들은 문제해결의 절차를 분석, 설계, 구현의 3단계로 설정한 후 컴퓨팅사고력(Computational Thinking)의 9가지 핵심요소를 적용하여 문제를 해결하는 과정과 실제 소프트웨어개발 단계에서 각각의 특색을 나타내고 있다.

본 연구는 그간 기초소프트웨어교육 설계에서 고려되지 않았던 계열별 학습자의 특성 분석을 통해 학문계열간 차별적 교육설계의 필요성을 제시한 연구이다. 본 연구가 현재 대학에서 실시하고 있는 기초소프트웨어교육모델 체계화에 기초연구로 활용되길 바란다.

References

[1] Bar, D., Harrison, J., and Conery, L., “Computational thinking: A Digital Age Skill for Everyone,” Learning & Learning with Technology, Vol. 38, No. 6, pp. 20-23, 2011.

[2] Carlisle, M. C., Wilson, T. A., Humphries, J. W., and Hadfield, S. M., “RAPTOR: Introducing Programming to Non-Majors with Flowcharts,” Journal of Computing Sciences in Colleges, Vol. 19, No. 4, pp. 52-60, 2004.

[3] Choi, J. W., “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, 2017.

[4] Computer Science Teachers Association & International Society for Technology in Education, Computational Thinking Teacher Resources, http://csta.acm.org/Curriculum/sub/CompThinking.html, 2011.

[5] Kim, C., “Needs of Improving the Curriculum of National University of Education for Strengthening SW Education,” Journal of the Korean Association of Information Education, Vol. 23, No. 1, 2019.

[6] Kim, W. S., “A Study on the Recognition of Freshman on Computational Thinking as Essential Course,” Culture and Convergence, Vol. 39, No. 6, pp. 141-170, 2017.

[7] Korea Institute for Curriculum and Evaluation, “Introduction to the revised curriculum in 2015”, Research Report CRC 2015-28, 2015.

[8] Kwon, J. I. and Kim, J. H., “A Study on Design and Development of SW Course based on Computational Thinking”, in Proc of Asia Pacific International Conference on Information Science and Technology(APIC-IST), 2017.

[9] Kwon, J. I., “A Study on the Effectiveness

of Computational Thinking based Teaching and Learning on Students' Creative Problem Solving Skills," Ph.D. Dissertation, Sungkyunkwan University, 2014.

[10] Lee, M. S., "Non-Engineering Challenges of Software Education," The Korean Association of General Educational Conference, pp. 135-140, 2016.

[11] Lee, S. J., "A Study on Designing a Class of Convergence Thinking based on Computational Thinking," The Korean Society of Science & Art 36, pp. 255-26, 2018.

[12] Papert, S., "An Exploration in the Space of Mathematics Educations," International Journal of Computers for Mathematical Learning, Vol. 1, No. 1, pp. 95-123, 1996.

[13] Schwab, K., "Introduction in The Fourth Industrial Revolution," https://www.weforum.org/about/the-fourth-industrial-revolutionby-klaus-schwab (retrieved on February 1, 2017), 2016.

[14] Su, E. K., "Development of Creative Thinking and Coding Course method on Design Thinking using Flipped Learning," Journal of Learner-Centered Curriculum and Instruction, Vol. 17, No. 16, pp. 173-199, 2017.

[15] Urban-Lurain, M. and Weinshank, D. J., "Is There A Role for Programming in Non-Major Computer Science Courses?," Frontiers in Education Conference, FIE2000 30th Annual, 2000.

[16] Wing, J. M., "Computational Thinking and Thinking about Computing," Philosophical Transactions of the Royal Society, Vol. 366, pp. 3717-3725, 2008.

[17] Wing, J. M., "Computational Thinking," Communications of the ACM, Vol. 49, No. 3, pp. 33-35, 2006.

저 자 소 개

권정인	(E-mail : jikwon@smu.ac.kr)
2009년	홍익대학교 컴퓨터교육과 (석사)
2014년	성균관대학교 컴퓨터교육과 (박사)
2014년~2015년	경희대학교 공학인증센터 연구교수
2015년~2018년	성균관대학교 소프트웨어학과 초빙교수
2018년~현재	상명대학교 계당교양교육원 조교수
관심분야	컴퓨팅사고력, 소프트웨어교육, 교양교육