「정보시스템연구」 제28권 제4호
한국정보시스템학회
2019년 12월, pp. 403~424

http://dx.doi.org/10.5859/KAIS.2019.28.4.403

비전공자 소프트웨어 기초교육을 위한 프로그래밍 언어 결정에 관한 연구

박 소 현*

〈목 차〉

Ⅰ. 서론
Ⅱ. 관련 연구
2.1 비전공자 대상의 소프트웨어 기초교육
2.2 ADD씽킹
2.3 프로그래밍 언어
Ⅲ. 소프트웨어 기초교육 프로그래밍 언어 결정
3.1 소프트웨어 기초교육 교과과정
3.2 소프트웨어 기초교육 프로그래밍 언어 결정을 위한 연구
3.2.1 델파이 조사
3.2.2 델파이 조사 대상
3.2.3 델파이 설문문항
Ⅳ. 델파이 설문결과
4.1 2차 델파이 설문결과 분석
4.2 3차 델파이 설문결과 분석
4.3 최종 결과
Ⅴ. 결론 및 향후계획
참고문헌
<Abstract>

Ⅰ. 서 론

4차 산업혁명은 모든 기술과 정보를 공유하고 협업한다는 점에서 기존의 산업혁명과 차별되며 새로운 부가가치를 창출한다는 특징을 가진다. 2018년 9월 세계경제포럼은 '일자리의 미래' 보고서는 4차 산업혁명으로 인해 기업에서 요구하는 직무 역량에 큰 변화가 있을 것이라고 예상하였다. 이 보고서에서는 2018년부터 2022년까지 5년간의 일자리 변화를 예측하기 위해 기술 요구사항, 채용패턴 교육의 필요성과 관련해 설문조사를 실시하였다. 그 결과 비교적 단기 미래 전망이긴 하나 인공지능의 발전으로 비숙련 혹은 저숙련 단순노동 뿐 아니라 숙련이 필요하나 반복적인 업무 영역의 경우까지 AI나 로봇이 대체할 가능성이 큰 것으로 발표하였다. 모든 산업 분야에서는 창의적이고 혁신적인 방향으로 문제해결이 가능한 융·복합 인재를 요구하고 있으며(Bennett et al., 2013), 창의적으로 문제를 해결하는 데 있어 컴퓨팅 사

* 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학지원사업의 연구결과로 수행되었음(2017-0-00091)
** 단국대학교 SW중심대학, sohyunpark@dankook.ac.kr

고력을 갖춘 인재를 확보하는 것이 곧 국가의 경쟁력이 될 것이라고 예상한다(성정숙과 김현철, 2015; 안인회, 2016; 신윤희 등, 2019; Senske, 2017; Denning, 2017). 그러므로 4차 산업혁명 시대를 이끌 창의적이며 논리적인 컴퓨팅 사고력을 갖춘 인재 양성을 위한 소프트웨어 교육의 중요성이 날로 증가하고 있으며, 디지털 시대를 창의적이고 주도적으로 디자인할 수 있는 소프트웨어 교육이 필요하다. 이에 국가 차원에서 먼저 소프트웨어 교육의 중요성을 인지하고 정보 교육을 위한 교육과정을 제시하였다. 2018년 중학교 소프트웨어 교육 의무화를 시작으로 2019년 초·중·고등학교까지 소프트웨어 교육 도입을 점차 확대하고 있다. 또한, 과학기술정보통신부에서는 2019년 10월을 기준으로 전국의 40개 대학을 'SW중심대학'으로 선정하여 소프트웨어를 중심으로 대학 교육 혁신을 통해 인재를 육성하고 있다. SW중심대학으로 선정된 대학들은 소프트웨어 전공자에 대한 전문성을 기르는 것은 물론이고, 모든 계열의 비전공자들을 대상으로 소프트웨어 교육 확대를 위한 교양필수 과목을 개설하여 창의 융합형 인재를 양성하고 있다(서응교, 2017). 소프트웨어 개발은 더 이상 소프트웨어 전공자들만을 위한 것이 아니다. 다가올 미래를 대비하여 컴퓨팅 사고력을 기반으로 한 융·복합 전문 역량을 가진 인재를 양성하기 위하여 다양한 분야에서 새로운 형태의 교육에 관한 연구가 진행 중이다(김시정과 조도은, 2018; 신희성 등, 2018; 오경선 등, 2018; Gerber et al, 2015).

본 논문에서는 비전공자의 소프트웨어 기초교육을 위하여 기존의 프로그래밍 언어의 단순 습득에 그치는 획일적인 소프트웨어 교육을 지양하고, 다양한 관점과 새로운 문제 발굴을 통해 창의적인 해결방법을 찾아 이를 소프트웨어로 구현할 수 있는 역량을 갖춘 인재로 성장시킬 수 있는 소프트웨어 교육을 지향한다. 이에 단과대학별 전공에서 필요로 하는 역량에 따라 인문, 사회, 공학, 자연, 예체능 5개 계열로 분류하여 계열별 특성에 맞춘 교육과정을 운영할 수 있도록 비전공자 소프트웨어 기초교육을 위한 맞춤 프로그램 언어들을 제시한다. 이를 위해 컴퓨터교육, 컴퓨터공학, SW교육, 교육학 전문가를 대상으로 준개방 형태의 델파이 기법을 이용한 전문가 조사를 실시하였으며, 문헌연구 및 전문가 자문회의를 통해 선정한 계열별 SW기초교육에 대한 도구(프로그래밍 언어)의 선택 및 적합성을 리커트(Likert) 척도(7점 척도)와 개방형 의견을 통해 검증하였다.

서응교 등(2018)에 의해 제시된 계열별 맞춤형 비전공자 소프트웨어 기초교육은 ADD씽킹에 따라 3단계로 구성되어 있으며, 본 논문에서는 1단계 디자인씽킹의 교육과정을 참고하여, 2단계 알고리즘씽킹을 위한 교육과정에서 사용하는 프로그래밍 언어를 계열별 특성에 맞게 결정함으로써 비전공자에 대한 소프트웨어 역량 강화를 목표로 한다.

Ⅱ. 관련 연구

2.1 비전공자 대상의 소프트웨어 기초교육

소프트웨어 교육이란 단순히 컴퓨터를 활용하는 방법을 학습하는 교육이 아니라 컴퓨터과

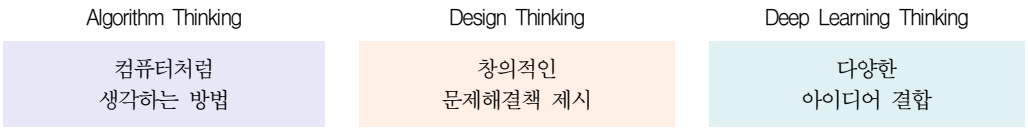
학의 원리를 적용하여 창의적으로 문제를 해결하는 방법을 학습하는 것이다(오경선과 안성진, 2016; 이명숙, 2017; 오창규 외, 2015). 그러나 기존의 소프트웨어 교육은 프로그래밍 언어에 대한 문법만을 지나치게 강조하거나 문제해결에 대해서 중요하게 다루지 않는다든지 혹은 프로그램 실행에 대한 경험이 부족한 상태에서 프로그램 교육이 이루어지고 있는 경우가 많다. 이는 실제 문제를 해결하는 데 도움이 된다고 볼 수 없으며, 프로그래밍 학습에 대한 반감을 사는 요인이 되기도 한다. 이러한 이유로 Crews(2018)는 프로그래밍 학습에 입문 시에 초보자들이 프로그래밍 언어의 문법만을 강조한 교육보다 순서도를 사용하는 것이 더 유용하다고 검증하였다. 특히 비전공자들의 교육에서는 프로그래밍 학습에 대한 어려움과 거부감으로 인하여 학습 효과가 차이 날 수 있으므로 기초교육용 프로그래밍 언어의 선택은 무엇보다 중요하다(안상진 등, 2012; 김영민과 이민정, 2019; 서현주 외, 2019).

비전공자를 위한 소프트웨어 교육의 목적은 모든 사람을 프로그래머로 만드는 것이 아니라 복잡하고 복합적인 원인으로 발생하는 문제를 인식할 수 있는 역량, 다양한 가치를 더하여 문제를 해결할 수 있는 능력, 협력을 통해 문제를 해결할 수 있는 능력을 기르는 것이다.

2.2 ADD씽킹

서응교 등(2018)에 의하면 ADD씽킹(ADD Thinking)은 창의적 사고를 요구하는 시대의 변화에 따라 소프트웨어로 구현하는 방법을 3가지 사고로 정의하고 있다. <그림 1>과 같이 ADD씽킹의 첫 번째 알고리즘씽킹(Algorithm Thinking)은 컴퓨터 기반의 논리적 사고 절차를 이해하고 컴퓨터처럼 생각하는 방법이다. 두 번째 디자인씽킹(Design Thinking)은 사용자 경험 중심으로 문제를 바라보고 공감, 문제정의, 아이디어, 프로토타입, 테스트 5단계의 디자인씽킹 프로세스에 따라 창의적인 사고로 문제를 해결하는 방법이다. 세 번째 딥러닝씽킹(Deep Learning Thinking)은 새로운 관점에서 새롭게 문제를 접근하고, 자신의 분야에서 컴퓨팅 사고를 내재화하여 창의 융합적 사고로 문제를 해결할 수 있는 방법이다.

ADD씽킹에서는 '문제를 만들어 내는 것이 때로는 문제를 해결하는 것보다 더 본질적인 것'이라고 말한 아인슈타인과 같이 창의적 문제 해결력에는 문제를 해결하는 것뿐만 아니라 새로운 문제를 제안하는 것 또한 중요하다는 점을 강조하고 있다. 좋은 연구 문제를 가지고 문제해결 가능성을 탐색하고, 구체적인 전략을 통해 문제를 해결해 나가야 한다는 것이다. 즉, 공감으로 시작하여 문제를 정의하고, 현실에서 접하는 문제해결을 위해 컴퓨팅 사고 기반의 논리적 설계를 거쳐, 새로운 관점으로 접근한 다양한 아이디어를 결합하여 창의적으로 문제



<그림 1> ADD씽킹의 정의

를 해결할 수 있는 확장적 사고를 해야 한다.

신희성 등(2018)은 대학 내 컴퓨터 교육의 목표와 지향하고자 하는 역량수준에 기반하여 ADD씽킹의 순서를 정의하였는데, 첫 번째 단계로 창의적 문제해결을 위한 일련의 순서를 창의적 사고를 통해 문제를 해결하는 디자인씽킹, 두 번째 단계로 컴퓨터 사고력을 바탕으로 비판적 사고를 키울 수 있는 알고리즘씽킹, 세 번째 단계를 다양한 정보들을 통해 문제를 새롭게 접근하고 해결할 수 있는 융·복합적인 사고를 위한 딥러닝씽킹으로 정의하였다. 이는 문제를 해결할 때 디자인씽킹으로 발산적 사고를 하여 다양한 아이디어를 탐색하고, 컴퓨팅 사고 기반의 알고리즘씽킹으로 절차화하여 실행하며, 디자인씽킹과 컴퓨팅사고의 결합을 통해 새로운 문제를 발견하고, 창의적으로 문제를 해결할 수 있는 능력의 확장이 가능하도록 도와준다. 본 연구에서는 두 번째 단계인 알고리즘씽킹 기반의 교육과정 수립을 위한 최적의 소프트웨어 개발언어 선정을 목표로 하였고, 전공영역별로 알고리즘씽킹 역량 향상을 위한 영역별 프로그래밍 언어를 제시하였다.

2.3 프로그래밍 언어

일반적으로 대학 내 소프트웨어 교육에서 많이 활용하는 언어는 <표 1>의 C, Python, Java 등과 같은 텍스트 기반의 언어이다. Python은 최근 가장 주목받고 있는 프로그래밍 언어로 높은 생산성과 확장성을 가지고 있으며 개발 시 메모리 관리가 자동으로 이루어지므로 문법이 간결하고 사용하기 쉬워 입문용 프로그래밍 언어로 각광 받고 있다. C언어는 미국 벨 연구소의 리치(D.M. Ritchie)가 개발한 시스템 기술(記述) 언어로 컴파일러나 소프트웨어 개발용 도구로도 사용되는 범용프로그래밍 언어의 일종이며, 실용적인 표현 형식, 근대적인 제어구조와 데이터구조, 그리고 풍부한 연산자를 갖고 있다. C++은 C언어 대부분의 특징을 포함하고 있는 확장판 개념으로 만들어진 것으로 시스템

<표 1> 텍스트 기반의 대표적 프로그래밍 언어

프로그래밍 언어	설명
Python	· C언어를 기반으로 한 오픈소스 고급 프로그래밍 언어
C	· 유닉스 개발하기 위한 프로그래밍 언어 문법이 간결하고 호환성, 이식성 등이 좋아 가장 오랜 시간 동안 사용되는 텍스트 기반의 프로그래밍 언어
C++	· 확장된 C언어. C언어에 객체지향 개념을 더하여 대규모의 프로그램을 만드는 언어
Java	· 객체지향프로그래밍 언어로서 보안성이 뛰어나며, C/C++에 비해 간략하고 쉬우며 네트워크 기능의 구현이 용이한 언어
Java Script	· 객체 기반의 스크립트 프로그래밍 언어이며, 문법이 간결하므로 입문자들도 쉽게 이해가 가능한 언어

프로그래밍에 적합할 뿐만 아니라 클래스, 연산자 중복, 가상 함수 등과 같은 특징을 갖추고 있어 C 언어에 객체지향의 개념을 더한 언어라고 할 수 있다. Java는 객체지향프로그래밍 언어로서 C/C++에 비해 간략하고 쉬우며 네트워크 기능의 구현이 용이하기 때문에, 인터넷 환경에서 가장 활발히 사용되는 프로그래밍 언어로 이식성이 좋아 운영체제의 종류에 관계 없이 대부분의 시스템에서 실행 가능하다. Java Script는 웹 페이지에서 사용자로부터 특정 이벤트나 입력 값을 받아 동적인 처리를 목적으로 고안된 객체 기반의 스크립트 프로그래밍 언어로 웹 브라우저 내에서 주로 사용하며, 다른 응용 프로그램의 내장 객체에도 접근할 수 있는 기능을 가지고 있다. 이와 같은 텍스트 기반의 언어들의 특성으로 인하여 기존의 프로그래밍 교육은 문법 위주의 내용으로 구성되어 있다. 이는 소프트웨어 비전공자들에게 프로그래밍은 어렵다고 느껴지게 하고, 프로그래밍 교육 자체의 흥미를 잃게 하는 원인이 되고 있다.

텍스트 기반 언어들은 수렴적 사고를 설계할 수 있는 프로그래밍 언어이며, 그와 반대로 확산적 사고로 설계할 수 있는 프로그래밍언어 즉 스토리텔링이 가능한 교육용 프로그래밍 언어로 App Inventor2, Scratch, ENTRY, Alice 등이 있다. 대표적 교육용 프로그래밍 언어에 대한 <표 2>에서와 같이 스크래치(Scratch)는 미국 매사추세츠공과대학(MIT) 미디어랩(Media Lab)의 라이프롱킨더가든그룹(LKG)이 만들어 무료로 제공한 것으로 저자와 독자가 양방향으로 소통하는 동화, 게임, 애니메이션 등으로 활발히 개발되고 있다. 프로그래밍을 처음 배우는 이들이 엄격한 문법에 대한 어려움 없이 쉽게 자신의 생각을 프로그램으로 표현할 수 있도록 디자인된 small talk 기반 EPL(MIT에서 개발)이며, 이용자가 창의적으로 생각하고, 체계적으로 판단하며, 협업하는 방법을 배울 수 있게 한 게 개발팀의 목표였다. 앱인

<표 2> 대표적 교육용 프로그래밍 언어

프로그래밍 언어	설명
스크래치 (Scatch)	· MIT에서 개발한 교육용 프로그래밍 언어. 프로그래밍을 처음 배우는 사람에게 적합하며, 엄격한 문법에 대한 어려움 없이 프로그램이 가능한 전 세계에서 가장 유명한 블록형 프로그래밍 언어
앱인벤터 (App Inventor)	· 태블릿과 스마트폰에서 사용 가능한 안드로이드 운영체제용 응용 소프트웨어를 구현 가능한 블록형 프로그래밍 언어
블록리 (Blockly)	· Google에서 개발한 교육용 프로그래밍 언어로 명령어 블록을 조립하여 만든 프로그램을 JavaScript, Python, PHP 등으로 변환할 수 있으며, 블록형 다음 단계의 프로그래밍 교육 단계로 발전하는데 도움을 줄 수 있는 블록형 프로그래밍 언어
러플(RUR-PLE) (RUR - Python Learning Environment)	· 화면의 로봇을 명령을 통해 움직일 수 있으며, 컴퓨터 프로그래밍에 대하여 학습이 가능한 파이썬을 이용한 교육용 프로그래밍 언어
엔트리(Entry)	· 국내에서 개발한 교육용 프로그래밍 언어로 블록형 프로그래밍 개발환경 사용하고, 국내 초·중등 학생 및 SW교사 대상의 프로그래밍 콘텐츠를 개발 보급 중인 프로그래밍 언어

벤터는 원래 구글이 제공한 오픈 소스 웹 애플리케이션으로, 2019년 현재는 매사추세츠 공과대학교(MIT)에 의해 관리되고 있으며, 안드로이드 기기에 쓰일 프로그램을 개발할 때 내 컴퓨터에서 해당 프로그램을 편리하게 디자인할 수 있도록 에뮬레이터를 제공해준다. 또한, 스크래치처럼 블록들을 움직이고 조립하여 프로그램을 완성할 수 있으며, 프로그래밍을 처음 접하는 사람들도 안드로이드 운영체제용 응용 소프트웨어를 쉽게 만들 수 있게 해준다. 블록리(Blockly)는 구글에서 개발한 웹 기반 블록형 프로그래밍 언어로 스크래치와 비슷하지만 블록리로 개발한 프로그램은 자바스크립트, 파이썬, 다트(Dart), XML과 같은 일반 프로그래밍 언어로 변환할 수 있어 일반 프로그래밍 언어에 대한 이해를 높일 수 있다. 러플(RUR-PLE)은 파이썬을 이용한 교육용 프로그래밍 언어 환경으로 명령을 통해 화면에 있는 로봇을 움직이며 컴퓨터 프로그래밍에 대한 학습을 도와준다. 엔트리는 언플러그드 활동을 지원하는 엔트리 보드게임과 함께 다양한 피지컬 컴퓨터 기기들도 연계할 수 있도록 지원해주고 있다.

그 외 카네기멜론 대학에서 만든 3D 프로그래밍 언어로 사람/동물/탈 것 등의 3D 오브젝트들과 drag-drop 인터페이스를 이용하여 스토리텔링형 애니메이션이나 간단한 비디오형 게임 등을 만드는데 효과적인 앨리스(Alice), 마이크로소프트(Microsoft)에서 개발한 게임 제작용 3D 비주얼 프로그래밍 언어로 다양한 게임 프로그래밍 및 스토리텔링형 프로그램 제작에 효과적인 코두(Kodu), MIT 인공지능 연구소에서 교육적 사용을 위해 개발한 Lisp 기반 언어로 거북이 로봇을 명령어를 통해 움직여 그림을 그리는 인터프리터 형태로 대중화한 로고(Logo), 황병욱 정보교사가 개발한 교육용 프로그래밍 언어로 주어진 상황 속에서 문제를 해결하는 데 초점이 맞춰져 있어 초등 이후의 학생들에게 적합한 교육용 프로그래밍 언어인 플레이봇(Playbot) 등이 있다.

최근에는 앞서 언급한 스크래치(Scatch), 앱 인벤터(App Inventor), 블록리(Blockly)와 같은 블록형 프로그래밍 언어와 러플(RUR-PLE), 플레이봇(Playbot) 등과 같은 다양한 교육용 프로그래밍 언어(EPL, Educational Programming Language)의 개발 덕분에 이러한 교육용 프로그래밍 언어들을 활용한 소프트웨어 교육이 활발하게 진행되고 있으며, 자신의 아이디어를 간단한 스크립트만으로도 표현할 수 있게 되었다(안상진 등, 2012; 김영민과 이민정, 2019). 그러나 이러한 교육용 프로그래밍 언어는 입문용 프로그래밍 언어의 목적을 가지고 개발되었으며 활용 폭이 넓지 않아 실무에서는 적극적으로 활용되지 않기 때문에 파이썬, Java Script, C, C++과 같은 텍스트 기반 프로그래밍 언어도 교육도 여전히 필요하다. 한편, Kuen(2011)의 랩터(raptor)나 Cook(2015)의 플로우고리즘(flowgorithm) 등과 같은 다이어그래밍 언어를 이용하여 쉽게 순서도를 작성할 수 있고, 순서도를 작성한 뒤에는 실행을 통한 검증이 가능하므로 어려운 프로그래밍 문법을 알지 못해도 알고리즘에 집중한 소프트웨어 교육이 가능하다.

이처럼 프로그래밍 언어는 다양한 특성을 가지고 있으며, 프로그래밍 언어에 관한 연구도 활발하게 진행되고 있다(안상진 등, 2012; 신수범과 구진희, 2014; 김영민과 이민정, 2019). 계

열별 특성에 맞게 프로그래밍 언어 선택하여 학습하고 활용한다면 소프트웨어 교육의 효과를 더 높일 수 있을 것으로 예상한다.

Ⅲ. 소프트웨어 기초교육 프로그램 언어 결정

3.1. 소프트웨어 기초교육 교과과정

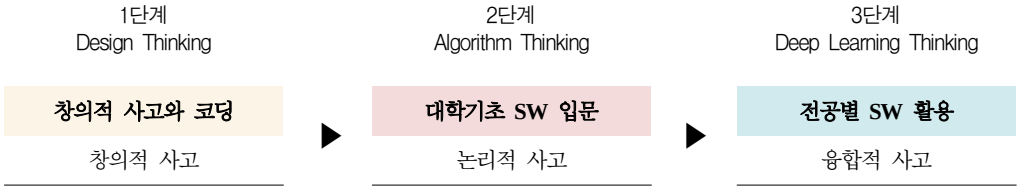
본 연구에서는 수도권 소재 D 대학교의 비전공자 대상의 소프트웨어 기초교육 사례에 기반한다. <표 3>은 D 대학교의 계열별 구분과 해당하는 단과대학이다. D 대학교는 소프트웨어 기초교육을 소프트웨어 전공자 SW융합대학과 비전공자 5개 계열로 구분한다. 비전공자는 인문(Humanities, Lawyers, Linguists), 자연(Scientists), 사회(Resource Planners and Managers), 공학(Engineers), 예체능(Artists)으로 구분된다.

비전공자의 소프트웨어에 대한 흥미 유발과 전문적인 소프트웨어 교육을 위하여 ADD씽킹의 3가지 사고들을 이용하여 교육과정을 3단계로 나누었다. <그림 2>는 소프트웨어 기초교육 3단계에 관한 설명이다. D 대학교에서는 2018년 1학기에 1단계인 '창의적 사고와 코딩' 교과목 운영을 시작하였다. 이 단계에서는 창의적 사고를 기반으로 창의성, 상상력, 흥미를 유발하여 컴퓨터를 활용한 자신의 실생활에서 존재하는 문제를 해결할 수 있도록 한다.

2단계의 교과목은 '대학기초 SW 입문'이다. 이 단계에서는 논리적 사고를 기반으로 좋은 문제를 만들어 문제해결 가능성을 탐색하고 구체적인 전략을 세워 문제를 해결한다. 마지막으로 3단계 교과목은 '전공별 SW 활용'이다. 이 단계에서는 융합적 사고를 기반으로 기존의 해결방법에 새로운 아이디어를 결합하여 문제를 해결한다.

<표 3> D 대학교의 계열별 구분과 해당 단과대학

SW	인문	자연	사회	공학	예체능
SW 융합대학	인문대학, 법과대학, 사범대학, 외국어대학	자연과학대학, 생명자원과학대학, 의과대학, 치과대학, 간호대학, 약학대학, 보건과학대학	사회과학대학, 상경대학, 국제학부, 공공인재대학	건축대학, 공과대학, 융합기술대학	예술디자인대학, 음악대학, 예술대학, 스포츠과학대학



<그림 2> 소프트웨어 기초교육 3단계

<표 4> ‘창의적 사고와 코딩’ 과목에서 계열별 선정된 프로그래밍 언어

계열	인문계열	자연계열	사회과학계열	공학계열	예체능계열
프로그래밍 언어	스크래치 (Scatch)	스크래치 (Scatch)	피오리 (Fiori)	블록리 &앱인벤터 (Blockly & AppInventor)	스크래치 (Scatch)

소프트웨어 기초교육 1단계 교과목에서는 계열별 전공 영역과 컴퓨터과학에서 창의적 사고력을 신장시키기 위한 내용을 학습하고, 이를 통해서 컴퓨터의 가치를 이해하고 소프트웨어 교육에 관한 관심을 유도하기 위한 내용으로 구성되었다. <표 4>는 창의적 사고와 코딩 과목에서 계열별 선정된 프로그래밍 언어이다.

소프트웨어 기초교육 2단계 교과목인 ‘대학기초 SW 입문’ 교과에서는 창의적 사고 기반의 다양한 해결책 중에서 소프트웨어 개발 관점으로 해결 가능한 주제를 선정한다. 그 후 최적의 해결방법을 선택하고, 이를 절차화하여 소프트웨어로 개발할 수 있는 능력을 기를 수 있도록 하는 데 목적이 있으며, 소프트웨어 비전공자 학생들의 알고리즘적 사고능력 함양을 위해 설계되었다. ‘대학기초 SW 입문’ 교과목의 대상은 1단계 교과목인 ‘창의적 사고와 코딩’을 이수한 학생이다. 이 교과목에서는 컴퓨팅을 이용하여 문제를 해결할 수 있도록 알고리즘적 사고와 논리적 사고를 유도하며, 전공별 특징에 맞는 프로그램 언어와 교육 내용을 통해 소프트웨어로 개발하는 방법을 학습한다. 2단계 교과목에서 가장 중요한 것은 자신의 아이디어를 컴퓨팅 도구를 활용하여 구현하고 실행할 수 있는 ‘알고리즘씽킹’ 역량을 키우는 것이다. 즉 ‘대학기초 SW 입문’ 교과목의 목표는 다양한 영역에서 창의적 사고를 표현하는 방법을 학습한 학생들이 ‘알고리즘씽킹’ 역량을 키워 자신만의 소프트웨어로 구현하는 방법을 학습하는 것이다.

3.2. 소프트웨어 기초교육 프로그램 언어 결정을 위한 연구

3.2.1 델파이 조사

본 연구에서는 델파이 조사를 통해 ‘무엇을 가르쳐야 하는가?’에 초점을 두고 계열별 특성에 맞는 알고리즘씽킹 역량을 키우기 위한 비전공자 소프트웨어 기초교육에서 활용될 프로그래밍 언어를 선정하다.

델파이 기법은 설문을 통한 전문적 견해에 근거하여 예측하는 방법으로 사회과학 및 교육 분야에서 널리 활용되고 있다. 본 연구의 델파이 조사 방법에서 정확한 추정치를 측정하기 위해 타당도 비율, 안정도 검증과 같은 여러 가지 통계분석 방법을 사용한다. 타당도 비율의 경우 Lawshe(1975)가 제안한 양적분석(CVR: Content Validity Ratio)값을 측정하여 평가한다. 각 설문 항목들의 적절성에 대해 리커트(Likert) 척도를 사용한다. 리커트 척도는 많은 사람을 대상으로 일관성을 확보하여 높은 신뢰도를 보인다는 점이 장점이다. 또, 대상자의 응

답 값을 직접 활용하므로 오류를 최소화할 수 있고, 다양한 설문 문항을 활용하므로 타당도가 높다. 리커트 척도는 3점, 5점, 7점 등 다양한 방식으로 구성될 수 있으나 측정 대상에 대한 응답자의 태도에 대한 보다 정교한 답변이 필요할 경우에는 7점 척도를 사용하므로, 본 논문에서는 리커트 7점 척도를 사용하여 보다 신뢰도 있는 조사라 할 수 있다.

CVR의 값은 식(1)과 같이 측정하며, Steiner와 Norman(2015)은 <표 5>에서와 같이 전문가 20명 이상인 경우 CVR 값이 0.42 이상일 때 통계적으로 유의미하며 내용이 타당하다고 해석하였다.

$$CVR = \frac{n_e - \frac{N}{2}}{\frac{N}{2}} \qquad \text{식 (1)}$$

n_e: 적합하다고 응답한 전문가의 수

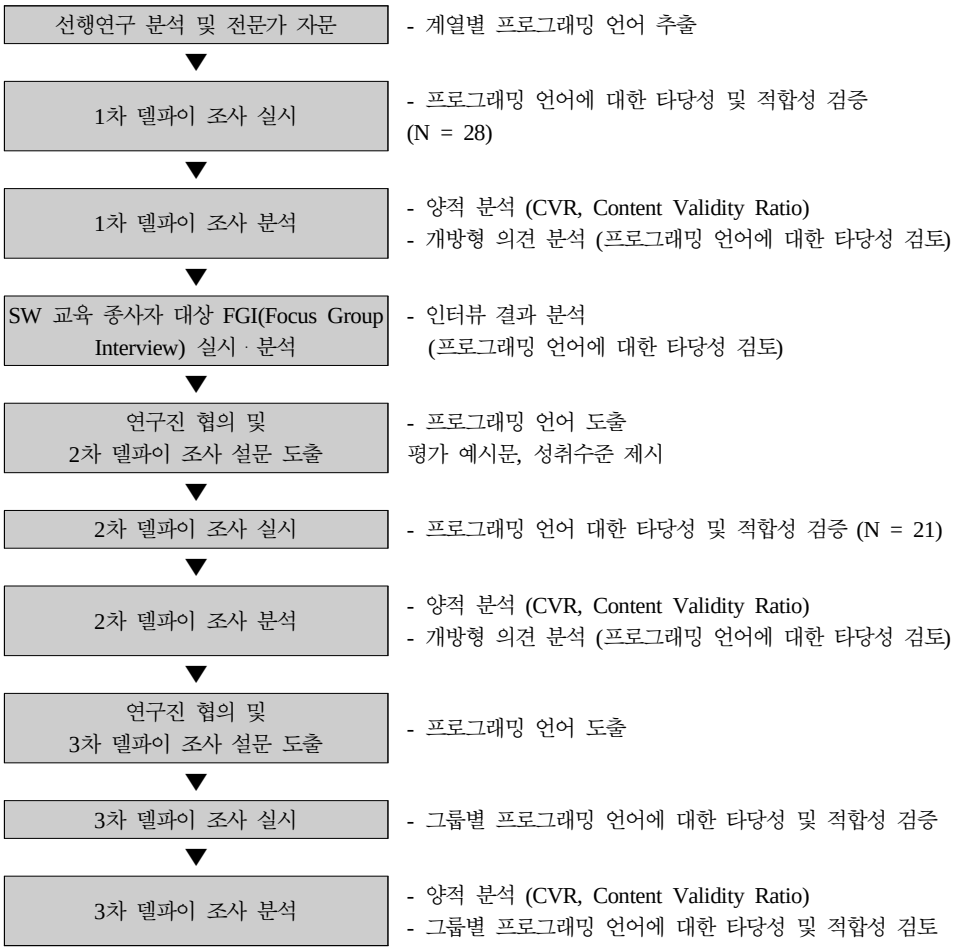
N: 전체 응답 전문가 수

<표 5> 응답 인원별 CVR 임계치

응답 인원수	임계치
5	0.99
6	0.99
7	0.99
8	0.75
9	0.78
10	0.62
11	0.59
12	0.56
13	0.54
14	0.51
15	0.49
20	0.42
25	0.37
30	0.33
35	0.31
40	0.29

양적 분석은 전문가의 선호도의 양적 결과가 높을수록 전문가의 선호도가 높은 것으로 해석할 수 있으며, 안정도(Stability)는 전문가의 합의 정도를 확인할 수 있다. 반복되는 설문결과들에서 전문가들의 설문 응답의 차이가 작은 경우 응답의 일치성이 높다고 판단할 수 있으며, 이것을 안정도가 확보되었다고 해석한다. 만약 모든 전문가가 같은 의견을 가지고 있을 때 안정도 값은 0이다.

델파이 조사 진행 순서는 다음 <그림 3>과 같이 진행하였으며, 가장 먼저 델파이 1차 조사를 위해 국내·외 관련 연구 자료와 D 대학의 SW 교양필수 과목 1단계인 ‘창의적 사고와 코딩’ 교과목을 기반으로 프로그램 언어를 도출하였다. 1차 조사에서는 문헌연구와 전문가 자문을 통해 선정한 계열별 소프트웨어 기초교육에 관한 내용과 프로그래밍 언어를 선택하고, 설문결과의 적합성을 리커트 척도와 개방형 의견을 통해 검증한다. 1차 델파이 조사 분석에서 개방형 의견 분석과 인터뷰 결과를 분석하여 프로그래밍 언어에 대한 타당성 검토 및 추천 프로그래밍 언어의 범위를 좁혀간다. 1차 조사의 결과를 기반으로 2차 델파이 조사 설문 문항을 도출하고, 2차 조사에서 프로그래밍 언어에 대한 타당성 및 적합성을 검증한다. 2차 델파이 조사 분석의 결과로부터 양적 분석, 개방형 의견 분석을 통해 프로그래밍 언어에 대한 타당성을 검토한다. 2차 조사의 결과를 기반으로 3차 델파이 조사 설문 문항을 도출하고, 3차 조사에서 그룹별 프로그래밍 언어에 대한 타당성 및 적합성 검증을 통해 최종적으로 2단계 소프트웨어 기초교육을 위한 프로그래밍 언어를 선정한다.



<그림 3> 델파이 조사 진행 순서

3.2.2 델파이 조사 대상

강용주(2008)에 의하면 델파이 기법의 경우 절차의 반복과 피드백을 통해 그 결과를 수정 또는 보완할 수 있으며, 설문 대상자 수는 10명 이상인 경우 그 결과가 유의미한 것으로 본다. 본 연구에서는 정보교과 교사, 대학 강사 또는 교수, 연구원, 개발자 21명을 델파이 설문대상으로 선정하였다. 이들은 소프트웨어 관련 전공으로 학위를 취득하였거나, 소프트웨어 교육의 종사자로, 박사일 경우 경력 7년 이상, 교사나 산업체 재직중인 경우 석사학위 이상과 경력 10년 이상으로 대상자를 선정하였으며 조사 대상에 대한 분포는 <표 6>과 같다.

<표 6> 델파이 조사 대상 분포

	컴퓨터 공학				경영 정보				컴퓨터 교육			
	학사	석사	박사수료	박사	학사	석사	박사수료	박사	학사	석사	박사수료	박사
연구원		1								1		1
개발자		2	1							1		1
교사									1	4		
교수/강사				3				1				4
합계	7명				1명				13명			

3.2.3 델파이 설문 문항

1차 설문의 경우 '대학기초 SW 입문' 과목에서 알고리즘씽킹에 관한 전문가 의견과 각 5개 그룹별 프로그래밍 언어에 대하여 개방형 질문을 통해 작성하도록 하였다. 델파이 1차 설문 문항 내용과 전문가 답변 예시는 <표 7>과 같다.

<표 7> 델파이 1차 설문 문항 내용과 전문가 답변 예시

[**질문1**] [대학기초 SW 입문] 교과 내용 설문에 앞서 참여해 주시는 분들의 Algorithmic Thinking에 대한 생각을 듣고자 합니다. 자유롭게 기술해 주시기 바랍니다.

Algorithmic Thinking 기반의 [대학기초 SW 입문] 교과의 목표 : 디자인씽킹 기반의 SW 개발 관점으로 다양한 해결책 중에서 최적의 솔루션을 선택(수렴)하고 절차화하여 소프트웨어로 개발 할 수 있는 능력을 기를 수 있도록 하는 데 목적이 있다.

[질문1-1] Algorithm Thinking이란 무엇이라고 생각하십니까?

전문가 답변 1: 어떠한 문제에 대하여 논리적인 절차에 따라 문제를 해결하는 것을 의미한다고 생각한다. 논리적인 절차라는 것은 먼저 해결해야 할 문제에 대해 파악하여 문제의 정의를 내리고 그것을 해결하기 위한 절차나 방법을 구체적으로 구조화시켜 해결해 나가는 것을 의미한다.

전문가 답변 2: 알고리즘이란 주어진 문제를 해결하는 방법이나 절차의 집합으로 일상생활 속에서 부딪히는 문제를 효율적으로 해결하는 것이 필요하듯이 학생들이 주어진 문제를 효율적으로 해결해가는 과정을 소프트웨어 구현을 통해 논리적인 사고력과 문제해결 능력 증진에 큰 도움이 된다고 생각합니다.Algorithm Thinking과정에서 텍스트기반 언어를 통해 프로그래밍의 기본 원리를 이해할 수 있으며 다양한 문제에 대한 해결책의 설계와 구현을 통해 논리적 사고력 증진과 문제해결 능력을 향상시킬 수 있다고 생각합니다.

[질문1-2] Algorithm Thinking을 학생들이 습득하기 위해서는 어떠한 방법이 필요하다고 생각하십니까?

전문가 답변 1: 자신이 머릿속으로만 생각했던 것을 어떤 입력된 데이터에 대하여 처리과정을 거쳐 결과를 도출해 내는지를 구체적으로 사고하며 프로그래밍을 해나가는 과정을 이해하는 것이 중요하며, 논리적인 사고를 통해 여러 가지 다양한 문제를 해결할 수 있다는 것을 배울 수 있다.

전문가 답변 2: 문제를 해결하는 방법이나 절차를 습득하기 방법으로 다양한 방법 중에 순서도나 의사코드 등을 이용하여 주어진 문제를 해결해 과정을 이해할 수 있도록 하고 난 후 특정 프로그래밍 언어를 통해 실제 구현을 해봄으로써 결과를 확인해 볼 수 있도록 한다. 문제를 해결하기 위해서는 정확하게 무엇을 해야 할지 처리 내용과 처리 순서를 모두 구체적으로 명시하고 효율적으로 해결하는 과정이 중요하기 때문에 알고리즘 씽킹 교과목에서 문제를 해결하기 위한 논리적 사고력 증진 시킬 수 있도록 수업을 설계하고 다양한

분야의 주제에 대해 접근해볼 수 있도록 한다. 수업 초기에는 일상생활 속에서 부딪힐 수 있는 주제로 알고리즘을 통한 문제해결 과제를 접근해 보고, 수업 중기에는 알고리즘 분야에 몇 가지 대표적인 예로 접근해 본 후 수업 후기에는 주어진 문제에 대해 아이디어를 만들어 알고리즘을 설계하고 프로그래밍 언어를 통해 구현해봄으로써 문제해결력을 신장시키도록 한다.

[질문2] [대학기초 SW 입문] 교과목 개요는 다음과 같습니다.

· 학생대상자 : 창의적 사고와 코딩 교과목을 이수한 학부생
목표 : 1단계 창의적 사고와 코딩 교과목을 통해 다양한 영역에서 창의적 표현 형태를 탐험하도록 여러 가지 주제와 연습을 한 학생들이 **자신의 아이디어를 컴퓨팅으로 구현하여 실행 할 수 있도록 '컴퓨터처럼 생각하는 방법'의 역량을 높여 소프트웨어를 개발**한다.

[질문2-1]
비전공자 대상의 [대학기초 SW 입문]에서 컴퓨팅으로 구현하는 데 있어서 도움을 줄 수 있다고 생각하는 프로그래밍 언어를 적어 주시기 바랍니다. (여러 개를 제시해 주셔도 됩니다.)

전문가 답변 1:

구분	
계열	프로그래밍 언어(도구)
인문	Blockly, Scratch, Appinventor
자연	Python, RURPLE
사회	Appinventor, RURPLE, Blockly
공학	Python, C, Visual C++, Java
예체능	Blockly, Scratch

전문가 답변 2:

구분	
계열	프로그래밍 언어(도구)
인문	Scratch, Appinventor
자연	Appinventor, Python
사회	Appinventor, Python
공학	Python, Java, JavaScript
예체능	Scratch, Appinventor

2차 설문의 경우 1차 설문결과를 근거로 5개 그룹별 학습에 사용할 프로그래밍 언어에 대하여 설문하도록 구성하였다. <표 8>은 델파이 2차 설문 문항 내용과 대상자의 설문결과 예시이다.

<표 8> 델파이 2차 설문 문항 내용과 전문가 답변 예시

[질문1] 비전공자 대상의 [대학기초 SW 입문]에서 **논리적 사고**뿐만 아니라 **이후 전공계열과 연계**할 수 있는 프로그래밍 언어로 컴퓨팅으로 구현하는 데 있어서 도움을 줄 수 있다고 생각하는 **프로그래밍 언어에 대한 계열별 적절성 정도를 체크해 주시기 바랍니다. (적절성 정도가 3점 이하인 경우 그 이유를** 적어주시기 바랍니다.)

전문가 답변 1:

프로그래밍 언어	적절성 정도 (7점 척도)				
	인문	자연	사회과학	공학	예체능
스크래치(Scatch)	6	2	6	2	7
앱인벤터(App Inventor)	5	4	7	4	6
블록리(Blockly)	7	4	6	3	7
러플(RUR-PLE)	5	6	6	5	4
플레이봇(Playbot)	4	3	3	3	3
파이썬(Python)	4	7	5	7	4
JS(Java Script)	2	3	2	3	2
C	1	6	1	7	1
C++	1	6	1	7	1

전문가 답변 2:

프로그래밍 언어	적절성 정도 (7점 척도)				
	인문	자연	사회과학	공학	예체능
스크래치(Scatch)	7	7	7	4	7
앱인벤터(App Inventor)	7	7	6	5	6
블록리(Blockly)	7	7	7	4	7
러플(RUR-PLE)	4	7	5	5	4
플레이봇(Playbot)	4	7	5	5	4
파이썬(Python)	3	5	4	7	2
JS(Java Script)	2	2	1	6	1
C	1	1	1	6	1
C++	1	1	1	6	1

3차 설문의 경우 2차 설문결과로 전문가에게 선택받지 못한 언어를 제거하고, 5개 그룹별 학습에 사용할 프로그래밍 언어에 대하여 설문하도록 구성하였다. <표 9>는 델파이 3차 설문 문항 내용과 대상자의 설문 답변 예시이다.

<표 9> 델파이 3차 설문 문항 내용과 전문가 답변 예시

[질문1] 비전공자 대상의 [대학기초 SW 입문]에서 **논리적 사고**뿐만 아니라 **이후 전공계열과 연계**할 수 있는 프로그래밍 언어로 컴퓨팅으로 구현하는 데 있어서 도움을 줄 수 있다고 생각하는 **프로그래밍 언어에 대한 계열별 적절성 정도를 체크해 주시기 바랍니다. (적절성 정도가 3점 이하인 경우 그 이유를** 적어주시기 바랍니다.)

전문가 답변 1:

프로그래밍 언어	적절성 정도 (7점 척도)				
	인문	자연	사회과학	공학	예체능
스크래치(Scatch)	7		7		7
앱인벤터(App Inventor)	5	6	5		5
블록리(Blockly)	5		6		4
러플(RUR-PLE)		2	4	2	
플레이봇(Playbot)					
파이썬(Python)				7	
JS(Java Script)				2	
C				6	
C++				6	

전문가 답변 2:

프로그래밍 언어	적절성 정도 (7점 척도)				
	인문	자연	사회과학	공학	예체능
스크래치(Scatch)	7		7		7
앱인벤터(App Inventor)	6	7	6		6
블록리(Blockly)	7		7		7
러플(RUR-PLE)		6	6	7	
플레이봇(Playbot)					
파이썬(Python)				7	
JS(Java Script)				5	
C				5	
C++				4	

Ⅳ. 델파이 설문결과

4.1. 2차 델파이 설문결과 분석

1차 설문의 개방형 질문의 경우 선정된 대표적 언어 9개를 대상으로 계열별 적절성 정도를 조사하였다. 본 연구에서는 설문대상이 21명이므로 3.2.1의 <표 5>와 같이 20명 기준의 CVR 값을 사용하여, 최소 0.42 이상일 때 유의미하다고 분석하였다. 그 결과 <표 10>와 같이 인문계열에서 6개의 프로그래밍 언어가 배제되고 CVR 값이 0.42 이상인 3개의 프로그래밍 언어

<표 10> 계열별 델파이 설문 2차 결과

	언어	스크래치	앱인벤터	블록리	러플	플레이봇	파이썬	JS	C	C++
인문	평균	5.952381	5.238095	5.47619	4.47619	4.047619	4.190476	3.571429	2.428571	2.285714
	CVR	0.818182	0.727273	0.727273	0.181818	0	0	-0.63636	-0.90909	-1
	안정도	0.197261	0.311223	0.301477	0.433959	0.427757	0.393976	0.477773	0.547617	0.525397
자연	언어	스크래치	앱인벤터	블록리	러플	플레이봇	파이썬	JS	C	C++
	평균	5.388889	5.333333	5	4.944444	4.555556	4.888889	3.944444	3.388889	3.333333
	CVR	0.272727	0.545455	0.363636	0.545455	0	0.363636	-0.36364	-0.27273	-0.27273
	안정도	0.352672	0.372033	0.382884	0.402415	0.454859	0.360925	0.392314	0.529654	0.551245
사회	언어	스크래치	앱인벤터	블록리	러플	플레이봇	파이썬	JS	C	C++
	평균	5.47619	5.333333	5.047619	4.952381	4.238095	4.666667	3.47619	2.761905	2.714286
	CVR	0.636364	0.727273	0.545455	0.454545	0	0.363636	-0.54545	-0.72727	-0.81818
	안정도	0.296163	0.313392	0.364156	0.376354	0.404806	0.345736	0.514748	0.59025	0.581335
공학	언어	스크래치	앱인벤터	블록리	러플	플레이봇	파이썬	JS	C	C++
	평균	4.47619	5.047619	4.47619	5.095238	4.238095	6.47619	5.904762	5.952381	5.761905
	CVR	-0.09091	0.363636	-0.09091	0.545455	-0.09091	0.909091	0.727273	0.818182	0.818182
	안정도	0.433959	0.33752	0.387678	0.314163	0.398203	0.155285	0.208433	0.2565	0.292893
예체능	언어	스크래치	앱인벤터	블록리	러플	플레이봇	파이썬	JS	C	C++
	평균	6	5.190476	5	4	4.095238	3.428571	3	2.190476	2.095238
	CVR	0.818182	0.636364	0.636364	-0.18182	0	-0.54545	-0.72727	-1	-1
	안정도	0.235702	0.295008	0.370328	0.469295	0.43879	0.387896	0.57275	0.538674	0.568636

<표 11> 계열별 델파이 2차 설문에 대한 개방형 전문가 의견

계열별	전문가 의견
인문계열	· 인문 학생들은 아이디어 수준으로 나타낼 수 있는 언어가 적절
자연계열	· 너무 쉬운 언어보다는 이후 수업을 위한 언어 혹은 과학을 접목하여 활용할 수 있는 언어를 가지고 알고리즘 접근하는 것이 동기부여가 될 것
사회과학계열	· 실제 프로그래밍으로 변환해 주는 언어 또는 가시적인 결과를 확인할 수 있는 언어가 적절
공학계열	· 이후 수업을 위한 텍스트기반 언어가 적당해 보임. 만약 파이썬 언어에 곧바로 접근하는 것이 부담되는 경우 러플을 통해 파이썬 기초를 학습하는 것도 용이
예체능계열	· 고유의 능력을 발휘하는 것에 목표를 둘 수 있는 언어가 적절

인 스크래치, 앱인벤터, 블록리가 선정되었으며, 자연계열은 7개의 프로그래밍 언어가 배제되고 CVR 값이 0.42 이상인 앱인벤터와 러플이 선정되었다. 사회계열은 5개의 프로그래밍 언어가 배제되고 CVR 값이 0.42 이상인 스크래치, 앱인벤터, 블록리 그리고 러플이 선정되었고, 공학계열은 4개의 프로그래밍 언어가 배제되고 CVR 값이 0.42 이상인 러플, 파이썬,

Java Script, C, C++이 선정되었으며, 예체능계열 델파이 설문 2차 결과는 7개의 프로그래밍 언어가 배제되고 CVR 값이 0.42 이상인 스크래치, 앱인벤터, 블록리가 선정되었다.

델파이 2차 설문 조사 결과로 나온 계열별 전문가 개방형 검토 의견은 위의 <표 11>과 같다.

4.2. 3차 델파이 설문결과 분석

계열별로 2차 델파이 설문결과 선정된 언어만을 채택하여, 3차 델파이 설문을 진행하였다. 3차 델파이 설문에서 각 프로그램 언어를 선정할 때에는 가장 높은 CVR 값을 가진 언어를 선택하고, 같은 CVR 값일 경우 안정도가 더 낮은 값을 가진 언어를 선정한다.

<표 12> 계열별 델파이 설문 3차 결과

인문계열	3차	스크래치	앱인벤터	블록리		
	CVR	0.684211	0.894737	0.578947		
	안정도	0.264309	0.232829	0.28482		
자연계열	3차	앱인벤터	러플			
	CVR	0.578947	0.578947			
	안정도	0.430813	0.259323			
사회계열	3차	스크래치	앱인벤터	블록리	러플	
	CVR	0.684211	0.894737	0.684211	0.473684	
	안정도	0.268039	0.222952	0.269756	0.228206	
공학계열	3차	러플	파이썬	JS	C	C++
	CVR	0.263158	0.894737	0.789474	0.894737	0.789474
	안정도	0.288097	0.140187	0.239246	0.179343	0.184091
예체능계열	3차	스크래치	앱인벤터	블록리		
	CVR	0.789474	0.789474	0.684211		
	안정도	0.235702	0.249141	0.273996		

위의 <표 12>는 계열별 델파이 3차 설문의 결과를 보여주는 것으로, 가장 높은 CVR 값의 결과가 선정된 프로그래밍 언어이다. 첫 번째, 자연계열에서 선정된 프로그래밍 언어는 러플이다. 러플의 3차 델파이 조사 결과 내용 타당도 비율 CVR 값은 0.58로 높은 편이며, 안정도 값은 0.26으로 매우 높은 안정 수준으로 볼 수 있다. 두 번째, 인문계열의 경우 선정된 프로그래밍 언어는 앱인벤터이다. 앱인벤터의 3차 델파이 조사 결과 내용 타당도 비율 CVR 값은 0.89로 아주 높은 편이며, 안정도 값은 0.23으로 매우 높은 안정 수준으로 볼 수 있다. 세 번째 예체능계열에서 선정된 프로그래밍 언어는 스크래치이다. 스크래치와 앱인벤터의 3차 델파이 조사 결과 내용 타당도 비율 CVR 값이 0.79 이상이다. 두 언어의 안정도를 비교한 결과 스크래치의 안정도는 0.24이고 앱인벤터의 안정도는 0.25이다. 이 결과 예체능계열의 언어로 스크래치를 선택할 때, 전문가들의 응답이 더 일치하는 것으로 판단할 수 있다. 네 번째, 사회계열의 경우 선정된 프로그래밍 언어는 앱인벤터이다. 앱인벤터의 3차 델파이 조사 결과 내용 타당도 비율 CVR 값이 0.89 이상으로 아주 높은 편이며, 안정도 값은 0.22로 매우 높은 안정 수준으로 볼 수 있다. 다섯 번째, 공학계열에서 선정된 프로그래밍 언어는 파이썬이다. 파이썬과 C언어의 3차 델파이 조사 결과 내용 타당도 비율 CVR 값이 0.89 이상이다. 두 언어의 안정도를 비교한 결과 파이썬의 안정도는 0.14이고 C언어의 안정도는 0.17이다. 파이썬을 선택할 때, 전문가들의 응답이 더 일치하는 것으로 판단할 수 있다. 따라서 공학계열의 언어로 파이썬을 선택한다.

<표 13>은 계열별 델파이 3차 설문에 대한 전문가의 다양한 검토 의견이며, <표 14>는 델

<표 13> 계열별 델파이 3차 설문에 대한 개방형 전문가 의견

계열별	전문가 의견
인문계열	· 인문 학생들은 직관적으로 이해할 수 있으며, 실생활에서 활용이 가능한 앱을 중심으로 수업을 진행하는 것이 집중이 높음
	· GUI형 프로그래밍 언어는 인문계 전공자가 알고리즘을 표현하기에 적합한 언어로 판단됨
	· Blockly보다 실제 사용 및 흥미를 줄 수 있는 앱인벤터가 학습에 도움이 됨
자연계열	· 데이터를 다루기 위한 텍스트 언어나 실생활 밀접한 앱에 관한 내용이 적합함. 또한, 실험 데이터와 휴대전화와 연결을 할 수 있음
	· 코드형 프로그래밍 언어는 일부 자연계 전공자만 유용하다고 판단되어 척도가 낮음
	· 자연계열은 전공에 대한 응용까지 고려한다면 파이썬이 필요하다 생각함
사회과학계열	· 사회과학은 인문계열보다는 데이터를 다룰 줄 알아야 하나 저학년일수록 생활밀착형 툴이 좋음
	· 코드형 프로그래밍 언어는 일부 사회과학 전공자만 유용하다고 판단되어 척도가 낮음
	· Blockly보다 실제 사용 및 흥미를 줄 수 있는 앱인벤터가 학습에 도움이 됨
공학계열	· 공학 기초인 C언어나 최근 주목을 받는 파이썬이 좋음
	· 공학계 전공자라도 코드형 프로그래밍 언어를 쉽게 이해하기 어려울 수 있고 입문 과정에서는 적절하지 않다고 판단됨
	· 블록기반 언어는 전공 연계가 어려움
	· 실제 해보면 러플이 파이썬보다 더 복잡함
	· 공학계열에는 러플보다는 파이썬을 배우는 것이 더 활용도가 높음
	· JavaScript는 웹 프로그래밍을 할 때 주로 사용하는 언어로 창의적인 알고리즘을 작성하는 방법을 배우는 언어로는 적절해 보이지 않음
예체능계열	· 예체능 학생들 중 음악과 미술 계열은 음과 미디어 아트 등을 주제로 하는 스크래치 류의 툴이 적합함
	· GUI형 프로그래밍 언어는 예체능 전공자가 알고리즘을 표현하기에 적합한 언어로 판단됨

<표 14> 델파이 설문 최종 결과로 선정된 프로그래밍 언어

계열	인문계열	자연계열	사회과학계열	공학계열	예체능계열
프로그래밍 언어	앱인벤터	러플	앱인벤터	파이썬	스크래치

파이 설문 최종 결과로 계열별로 최종 선정된 프로그래밍 언어들이다. 해당 언어들의 내용 타당도 비율 CVR 값은 0.578947 ~ 0.894737로 모든 문항이 최소 요구값 0.42를 만족하는 범위 안에 있어 델파이 설문결과를 신뢰할 수 있다. 또한, 안정도의 값은 0.140187 ~ 0.259323으로 매우 안정적인 응답으로 판단된다.

Ⅴ. 결론 및 향후 계획

본 연구는 소프트웨어 코딩으로 가치가 있는 소프트웨어 중심사회인 4차 산업혁명의 시대에서 필요로 하는 인재 양성을 위해 최근 주목받고 있는 비전공자 소프트웨어기초 교육을 위한 최적의 프로그래밍 언어 결정을 목적으로 한다. 이는 현실세계의 비정형화된 문제들을 소프트웨어로 해결하는 과정, 즉 문제를 분석하고 분해, 패턴을 인식하고 개체를 추상화하여 알고리즘으로 표현할 수 있는 알고리즘씽킹 기반의 교육과정 수립을 위한 최적의 소프트웨어 개발 언어 선정을 목표로 하는 것이며, 전공 영역별로 알고리즘씽킹 역량 향상을 위한 계열별 프로그래밍 언어를 제시하여 지능정보사회에 대비한 창의적이며 논리적인 역량 강화를 이끌 소프트웨어 교육에 활용할 수 있도록 한다.

경기도 D 대학에서는 비전공자의 소프트웨어에 대한 흥미 유발과 전문적인 소프트웨어 교육을 위하여 소프트웨어 비전공자를 인문, 자연, 사회, 공학, 예체능 5개 계열로 구분하고, ADD씽킹의 알고리즘씽킹, 디자인씽킹, 딥러닝씽킹의 3가지 사고들을 기반으로 소프트웨어 기초교육과정을 3단계로 나누었다. 각 단계에서는 ‘창의적 사고’, ‘논리적 사고’. ‘융합적 사고’에 초첨을 맞춰 학습하도록 설계되었다. D 대학에서는 2018년 1학기부터 1단계 교육인 ‘창의적 사고와 코딩’ 교과목을 운영하였으며, 2단계 교육인 ‘대학기초 SW 입문’ 교과목에서는 다양한 영역에서 창의적 사고를 표현하는 방법을 학습한 학생들이 ‘알고리즘씽킹’ 역량을 이용하여 자신만의 소프트웨어를 구현하는 방법을 학습하는 것을 목표로 한다.

본 연구에서는 ‘무엇을 가르쳐야 하는가?’에 초점을 두고 알고리즘씽킹 역량을 키우기 위한 비전공자 소프트웨어 기초교육에서 전공 및 계열별 특성에 맞는 프로그래밍 언어를 선정하기 위해서 21명의 전문가를 대상으로 3회에 걸쳐 델파이 조사를 실시하고, 양적 분석(CVR)값과 안정도를 사용하여 그 결과를 분석하였다. 델파이 기법은 전문가의 경험적 지식을 통한 문제해결을 및 미래예측을 위한 기법으로 다양한 방면에서 전문가들로 하여금 보다 객관적인 의견일치를 이끌어 내도록 지원해준다. 이에 해당 기법은 본 연구의 최종 결과인 프로그래밍 언어 도출에 최적의 기법이라 할 수 있으며, 정보교과 교사, 대학 강사 또는 교수, 연구원, 개발자 총 21명의 전문가들의 반복적인 피드백을 통해 최종 연구 결과를 도출하였다. 이때, 21명의 전문가이기 때문에 20명 기준의 CVR 값을 사용하여 0.42 이상이면 타당하다고 보았다. 1차 델파이 설문결과를 분석하여 소프트웨어 기초교육에 적합한 9개의 프로그래밍 언어를 선정하고, 선정된 언어를 기반으로 2차 델파이 조사 분석을 통해 CVR 값과 안정도를 근거로 계열별 프로그래밍 언어를 2~5개 선정하였다. 2차 델파이 조사 결과 선정된 언어를 기반으로 3차 델파이 조사를 실시하였으며, 3차 조사 결과의 CVR 값이 큰 언어를 해당 계열의 최종 프로그래밍 언어로 선정하였다. 만약 조사 결과에서 같은 CVR 값을 가지고 있다면 안정도가 낮은 프로그래밍 언어를 최종 언어로 선정하였다. 최종 결정된 계열별 가장 적합한 프로그래밍 언어로 인문계열은 앱인벤터, 자연계열은 러플, 사회과학계열은 앱인벤터, 공학계열은 파이썬, 예체능계열은 스크래치가 선정되었다. 해당

언어들은 비전공자 대상의 대학기초 SW 입문 과정에서 설계한 내용을 소프트웨어로 구현함에 있어 가장 적합하며 이후 전공계열과 연계에도 활용할 수 있는 프로그래밍 언어이다.

본 연구에서 제안한 계열별 프로그래밍 언어를 통해 소프트웨어 기초교육을 시작하는 대학의 교육과정과 운영 방향을 결정하는 기초자료로 활용될 것을 기대한다. 나아가 창의적으로 사고하여 이를 소프트웨어로 문제를 해결하는 방식에 익숙하게 하고, 소프트웨어를 자신의 영역에서 내재화하여 새로운 가치를 만들어 낼 수 있는 능력을 갖춤으로써 21세기를 살아갈 경쟁력을 가진 인재 육성에 보탬이 되고자 한다. 또한, 본 연구의 결론 및 도출한 결과를 바탕으로 향후 비전공자 소프트웨어 교육에서 다양한 프로그래밍 언어를 활용한 사례와 이를 교육과정에 적용한 결과 학습자들의 변화로부터 다양한 요구사항을 파악하고 교육과정에 재반영한다면 소프트웨어 기초교육에 보다 큰 도움이 될 것으로 사료된다.

참고문헌

강용주, “델파이 기법의 이해와 적용사례”, 한국장애인고용공단 고용개발원, 수시과제보고서, 2008, pp. 1-17.

김시정, 조도은, “효과적인 코딩교육을 위한 학습 모델에 대한 연구”, 한국융합학회논문지, 제9권 제2호, 2018, pp. 7-12.

김영민, 이민정, “비전공자를 위한 교육용 프로그래밍 언어의 비교 연구:프로그래밍 언어 설계 원칙의 관점으로”, 컴퓨터교육학회 논문지, 제22권 제1호, 2019, pp. 47-61.

서응교, “플립러닝과 디자인 씽킹에 기반을 둔 창의적사고 강화와 코딩교육을 위한 강좌 개발”, 학습자중심교과교육연구, 제17권 제16호, 2017, pp. 173-199.

서응교, 오경선, 정혜진, “창의적 사고와 코딩(인문)”, NOSVOS, 용인시, 2018.

서현주, 김진아, 김연정, 문남미, 김효근, “예술교육 매칭 플랫폼 구축을 위한 서비스 모델 개발”, 정보시스템연구, 제28권 제3호, 2019, pp. 227-247

성정숙, 김현철, “국외 컴퓨터 교육과정의 변화 분석”, 컴퓨터교육학회논문지, 제18권 제1호, 2015, pp. 45-54.

신수범, 구진희, “교육용 프로그래밍 언어의 선택 기준 개발”, 컴퓨터교육학회 논문, 제17권 제4호, 2014, pp. 13-21.

신윤희, 정효정, 서응교, “비공학계열 학생을 위한 디자인 씽킹 기반 코딩교육 프로그램 효과분석 및 지원전략 연구”, 학습자중심교과교육학회, 제19권 제10호, 2019, pp. 361-373.

신희성, 서응교, 정혜진, 박소현, “비전공자 SW 기초교육을 위한 디자인 씽킹 기반의 SW 교육 프로그램 개발 사례 연구”, 한국컴퓨터교육학회 학술대회논문집, 제22권 제2호, 2018, pp. 125-128.

안상진, 서영민, 이영준, “교육용 프로그래밍 언어 연구 동향”, 한국컴퓨터정보학회 학술발표논문집, 제20권 제1호, 2012, pp. 139 - 142.

안인회, “코딩교육의 현황과 미래”, 미디어와

교육, 제6권 제1호, 2016, pp. 76-87.

오경선, 서응교, 정혜진, “창의・컴퓨팅사고 교육내용 기본 설계 연구”, 디지털융복합연구, 제16권 제5호, 2018, pp. 65-73.

오경선, 안성진, “소프트웨어 교육을 위한 컴퓨팅사고 교육내용 설계 기본 연구”, 컴퓨터교육학회 논문지, 제19권 제2호, 2016, pp. 11-20.

오창규, 이홍걸, 김성후, “경영정보 관련 학과의 교육과정 설계와 운영 방안: K대학 e-비즈니스학과 사례를 중심으로”, 정보시스템연구, 제24권 제4호, 2015, pp. 117-138

이명숙, “컴퓨팅 사고력 향상을 위한 창의 융합적 SW교육 프로그램 연구”, 한국컴퓨터정보학회논문지, 제22권 제8호, 2017, pp. 93-100.

Bennett, V., Koh, K., and Repenning, A., “Computing Creativity: Divergence in Computational Thinking.” Proceeding of the 44th ACM Technical Symposium on Computer Science Education, 2013, 359-364.

Cook, D. D., “Flowgorithm: Principles for Teaching Introductory Programming Using Flowcharts.,” In Proc. American Society of Engineering Education Pacific Southwest Conf.(ASEE/PSW), 2015, pp. 158-167.

Crews, T., Ziegler, U., “The Flowchart Interpreter for Introductory Programming Courses.,” in Frontiers in Education Conference(FIE’98), Vol. 28, No. 1, 2018, pp. 307-312.

Denning, P. J., “Remaining trouble spots with computational thinking” Communications of the ACM, Vol. 60, No. 6, 2017, pp. 33-39.

Gerber, D.J., Khashe, S. and Smith, I.F.C., “Surveying the Evolution of Computing in Architecture, Engineering, and Construction Education”, Journal of Computing in Civil Engineering, Vol. 29, No. 5, 2015, Article number 4014060.

Kuen, Kwan Chi., “Learning Programming Concepts Using Flowcharting Software.,” Proceedings of the Global Chinese Conference on Computers in Education (GCCCE). 2011.

Lawshe, C. H., “A quantitative approach to content validity.,” Personel Psychology, Vol. 28, No. 4, 1975, pp. 563-575.

Senske, N., “Evaluation and impact of a required computational thinking course for architecture students”, Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE, 2017, pp. 525-530.

Streiner, D. L., & Norman, G. R., Health measurement scales: A practical guide to their development and use, 5 edition, Oxford, UK, 2015.

박 소 현 (Park, So Hyun)

단국대학교에서 소프트웨어 공학을 전공하여 공학박사 학위를 취득하고, 현재 단국대학교 SW중심대학 강의교수로 재직하고 있으며, 주요 관심 분야는 정보시스템, 비전공자 컴퓨터 교육 등이다.

<Abstract>

A Study on the Determination of Programming Language for Software Basic Education of Non-majors

Park, So Hyun

Purpose

The objective of this study is to determine the programming language for improving algorithmic thinking of basic software education for non-majors, which has recently been receiving attention to nurture talents needed in the era of the Fourth Industrial Revolution.

Design/methodology/approach

In this study, Delphi method was used to select the suitable programming language for the features of each of five departments for basic software education for non-majors in order to develop the capability of algorithmic thinking. The survey was conducted three times to 21 experts, and the results were analyzed using quantitative analysis (CVR) values and stability.

Findings

For the most suitable programming language for each department determined in this study, App Inventor was selected for humanities department, RUR-PLE for natural science department, App Inventor for social science department, Python for engineering department, and Scratch for fine arts department. This is expected to be used as the basis for determining the direction of curriculum and operation of universities starting basic software education through programming language by department proposed in this study.

***Keyword*:** Software Basic Education, Coding Education, ADD Thinking, Design Thinking, Computational Thinking, Algorithm Thinking, Programming Language for Non-majors

* 이 논문은 2019년 11월 11일 접수, 2019년 11월 12일 1차 심사, 2019년 12월 28일 2차 심사, 2019년 12월 29일 게재 확정되었습니다.