JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 20, No. 5, pp. 947-954, May 2019

컴퓨팅 사고력 기반 소프트웨어 분석설계 교육 모델

강환수[1] · 조진형[1] · 김희천[2*]
[1]동양미래대학교 컴퓨터정보공학과
[2]한국방송통신대학교 컴퓨터과학과

A Study on Software Analysis and Design Education Model based on Computational Thinking

Hwan-Soo Kang[1] · Jin-Hyung Cho[1] · Hee-Chern Kim[2*]
[1]Department of Computer Information Engineering, Dongyang Mirae University, Seoul, Korea
[2]Department of Computer Science, Korea National Open University, Seoul, Korea

[요 약]

최근 컴퓨팅 사고력은 초중등 학생이 함양해야 할 기본 역량으로 인식되고 있다. 특히 제4차 산업혁명을 주도할 인재는 융합력과 창의력이 필요하며, 컴퓨팅 사고력은 창의·융합 사고능력과 협업 능력에 효과적이라고 알려져 있다. 이러한 배경으로 컴퓨팅 사고력과 함께 창의·융합 능력을 갖춘 미래 인재 양성이 대학교육에서도 강조되고 있다. 본 연구에서는 컴퓨팅 사고력의 개념과 요소를 살펴보고 시스템분석설계 운영 사례와 컴퓨팅 사고력 기반 소프트웨어 분석설계 교과에서의 교육 모델을 제시하였다. 본 연구에서 제안하는 소프트웨어 분석설계 교육 모델을 활용한다면 컴퓨팅 사고력을 배양하여 자기 주도 학습, 창의·융합 사고능력, 의사소통과 협업 능력 등을 키우는 데 효과적이며, 프로그래머로서 분석설계 능력을 지닌 소프트웨어 개발자로 성장하는 데 도움을 줄 것으로 기대한다.

[Abstract]

Computational Thinking(CT) has recently been recognized as a basic skill that all primary and secondary school students need to develop. In particular, those who will lead the 4th Industrial Revolution should have convergent thinking and creativity. CT is known to be effective in cultivating creative and convergent thinking ability and collaboration skills. In this context, the cultivation of future human resources with creativity and convergence ability along with CT is also emphasized in university education. In this study, we review the concepts and elements of CT and present an education model for CT-based software Analysis&Design course through the case study of Systems A&D class. We expect that the education model proposed in this study will help students effectively cultivate CT ability and develop skills for self-directed learning, creative and convergent thinking, and communication and collaboration. The model will allow students to grow as programmers and software developers with A&D capabilities.

색인어 : 컴퓨팅 사고력, 소프트웨어 분석설계, 자기 주도 학습, 의사소통, 협업
Key word : Computational thinking, Software analysis and design, Self-directed learning, Communication, Collaboration

http://dx.doi.org/10.9728/dcs.2019.20.5.947

Received 24 April 2019; **Revised** 15 May 2019
Accepted 27 May 2019

***Corresponding Author; Hee-Chern Kim**

Tel: +82-2-3668-4657
E-mail: hckim@knou.ac.kr

Ⅰ. 서 론

2016년부터 인공지능과 빅데이터, 사물인터넷(IoT) 등과 연관된 미래 유망 기술을 기반으로 하는 4차 산업혁명이 전 세계를 휩쓸고 있다. 4차 산업혁명 시대에 맞는 인재양성을 위해 세계가 컴퓨터 교육에 노력을 기울이고 있다. 미국과 영국 등은 2000년대 중반부터 미래의 창의적인 인재를 육성하려는 노력으로 소프트웨어 교육의 중요성을 강조해왔다. 영국은 2014년 9월부터 컴퓨팅 과목을 초중등 모든 학년에 필수과목으로 지정하였으며, 미국의 컴퓨터과학교사협회(CSTA)는 2011년 개정된 컴퓨터과학 표준(K-12 Computer Science Standards)을 제시하여 유치원부터 고등 교육과정에 컴퓨터과학교육을 필수로 지정하였다. 일본은 2000년부터 초등학교, 2012년부터 중학교 교육과정에 정보 교과를 필수로 지정하였다.

4차 산업혁명 시대에 새롭게 강조되는 인재 역량은 단순 지식의 습득 능력보다 컴퓨팅 사고력(Computational Thinking) 기반의 창의적 문제발견 및 해결 능력이다. 현재 많은 산업이 정보기술과 융합하고 있으며, 컴퓨터 중심사회로 변화하고 있는 상황에서 컴퓨팅 사고력 기반의 소프트웨어 교육의 중요성이 매우 커지고 있다. 컴퓨팅 사고력은 1980년 시무어 페퍼트(Seymour Papert)가 처음 언급한 이후, 2006년 자넷 윙(Jeannette Wing)이 발표한 논문[1]에 의해 대중화되었다. 이후 컴퓨터가 우리 사회에 없어서는 안 될 중요한 기기가 되면서 컴퓨팅 사고력은 더욱 관심을 받게 되었으며 컴퓨팅 사고력은 말하기, 읽기, 쓰기, 셈하기 등과 같이 일반인도 배워야 할 기술로 인식되기 시작했다. 현재 초중등 및 대학에서 컴퓨터 비전공자를 대상으로 컴퓨팅 사고력을 향상하기 위한 소프트웨어 교육에 관한 연구는 상대적으로 많이 연구되었다[2]-[6]. 컴퓨팅 사고력을 통한 문제 해결 능력을 갖춘 미래 인재양성이 대학교육에서 강조되고 있는 시점에서 컴퓨터공학 전공자를 위한 소프트웨어 교육 연구도 매우 필요한 상황이다. 컴퓨터 전공자에게 프로그래밍 능력 자체도 중요하지만, 소프트웨어 시나리오 개발과 기획력, 창의융합적 사고력, 논리력이 중요하며, 이를 위해 컴퓨터 전공자에게도 컴퓨팅 사고력 중심의 교육이 필요하다. 본 연구에서 컴퓨팅 사고력의 개념을 간략히 살펴보고 D 대학의 소프트웨어 분석설계 관련 교과목을 소개하고 이 운영 사례를 통하여 대학교육에서 컴퓨터 전공자에게 필요한 컴퓨팅 사고 기반 소프트웨어 분석설계 분야의 교수·학습 모델을 제안한다.

Ⅱ. 연구 배경

2-1 컴퓨팅 사고력 개념

자넷 윙은 컴퓨팅 사고력을 컴퓨터과학자뿐만 아니라 누구나 배워 활용할 수 있는 보편적인 사고이자 기술로 인지하여 모든 사람에게 필요한 읽기, 쓰기, 셈하기와 같은 근본적인 기술이 되어야 한다고 말하고 있다[1]. 컴퓨팅 사고력은 컴퓨터과학적 지식의 구조가 바탕이 되어 창의적 문제 해결 방법이나 다양한 교과에 적용될 수 있다고 주장하였다[1]. 또한, 컴퓨팅 사고력의 숙달은 일을 체계적이고 효율적으로 처리하는 데 도움을 준다고 한다[7]. 영국의 정보기술 전문교육 집행 기관인 BCS(British Computer Society)에서는 컴퓨팅 사고력을 '문제와 해결방법을 구조화하여 정보처리 에이전트(agent)가 효과적으로 일을 처리할 수 있도록 표현하는 사고과정'으로, 이런 일련의 과정을 컴퓨터가 이해할 수 있게 변환시키는 인간의 인지적 능력으로 정의하였다[8]. 또한, 영국의 BBC 컴퓨팅교육에서는 컴퓨팅 사고력의 주요 구성을 분해, 패턴인식, 추상화, 알고리즘 설계 등 4가지의 요소[9]로 표 1과 같이 제안하고 있다.

표 1. BBC의 컴퓨팅 사고력 구성 요소
Table 1. BBC's Components of Computational Thinking

Cornerstones	Content
Decomposition	breaking down a complex problem or system into smaller, more manageable parts
Pattern Recognition	looking for similarities among and within problems
Abstraction	focusing on the important information only, ignoring irrelevant detail
Algorithm	developing a step-by-step solution to the problem, or the rules to follow to solve the problem

2011년 미국의 컴퓨터과학교사협회(CSTA)는 국제교육공학협회(ISTE)와 공동으로 컴퓨팅 사고력의 구성요소를 자료수집(data collection), 데이터 분석(data analysis), 데이터 표현(data representation), 문제 분할(problem decomposition), 추상화(abstraction), 알고리즘 및 프로시저(algorithm & procedures), 자동화(automation), 시뮬레이션(simulation), 병렬화(parallelization) 등의 9가지로 요약[10]하였다. 컴퓨팅 사고력의 9단계 과정을 통해 학생들에게 문제 해결 과정 자체에 관해 사고하는 방법을 가르치며, 문제를 규정하고 해법을 찾고 이를 위해 타인과 협업하는 능력을 배양하도록 한다. 우리나라의 2015년 개정 교육과정에서 컴퓨팅 사고력을 '컴퓨터과학의 기본 개념과 원리 및 컴퓨팅 시스템을 활용하여 실생활 및 다양한 학문 분야의 문제를 이해하고 창의적 해법을 구현하여 적용할 수 있는 능력'으로 정의[11]하고 있다. 국내의 연구에서는 컴퓨팅 사고력을 컴퓨터과학자들이 가진 독특한 지식의 구조를 바

표 2. 컴퓨팅 사고력의 DPAAP 모델
Table 2. DPAAP Model of Computational Thinking

Component	Content
D: Decomposition	Breaking down of a system into smaller parts that are easier to understand, program and maintain.
P : Pattern Recognition	Searching or identifying patterns and regularities in data
A: Abstraction	Simplification and formalizing found principles with pattern recognition
A: Algorithm	A plan, a set of step-by-step instructions to solve a problem
P: Programming	The act of expressing an algorithm using a syntax that the computer can understand.

탕으로 여러 학문 분야의 지식을 결합하고 융합하여 컴퓨팅 시스템을 이용하여 창의적으로 문제를 해결할 수 있는 인지적 사고능력으로 정의[12]하고 있다.

2015년, 한국교육개발원의 정책 연구에서는 컴퓨팅 사고력 중심의 소프트웨어 학습 모델로서 분해(D), 패턴인식(P), 추상화(A), 알고리즘(A)의 4가지와 선택적인 프로그래밍(P) 항목을 포함한 DPAAP 모델인 표 2를 제시하였다[13].

2-2 국내의 컴퓨팅 사고력 교육 현황

우리 정부는 2015년도 교육과정 개정을 통해 초중등에서 소프트웨어 교육을 일부 교육과정에 포함하고, 2017년에는 소프트웨어 교육 과목을 초등학교 정규교과로 편성하고, 2018년에는 중・고등학교 정규교과로 편성하였다. 이러한 배경에서 미래창조과학부와 정보통신기술진흥센터에서는 'K-ICT 전략'의 목적으로, 대학교육을 소프트웨어 중심으로 혁신함으로써, 학생·기업·사회의 소프트웨어 경쟁력을 강화하고자 SW 중심대학 사업을 시작했다. 고려대, 성균관대, 충남대 등 8개 대학에서 시작한 SW 중심대학 사업은 산업현장에서 요구하는 문제 해결 역량을 갖춘 창의적 소프트웨어 인재를 양성하는 데 초점을 맞춰 2018년까지 30개 대학으로 확대하여 시행하고 있다.

국민대는 국내 최초로 전체 신입생에게 SW 교육을 의무화하였고, 프로그래밍언어와 컴퓨터개론 등 1년 동안 SW 과목을 6학점 이상 수강하도록 하였다. 연세대는 컴퓨팅 사고력 과목을 2015학년도에 신설하여 시범 운영하고, 2016학년도부터 신입생 전원에게 필수로 진행하고 있다. 동국대는 2016학년도부터 비전공자 신입생 전원을 대상으로 '소프트웨어와 미래사회' 수강의 의무화를 통한 SW 융합형 교육을 시행하고 있다. SW 중점대학으로 선정된 대부분 대학에서 신입생 전체를 대상으로 필수 교양 과목으로 컴퓨팅 사고력, 프로그래밍입문, SW 융합교육 등을 교육하고 있다[14]-[15]. 그러나 SW 중점대학이 아닌 대학은 컴퓨팅 사고력에 관한 정보가 부족하며, 관련 과목 한두 개 과정을 개설하여 시험 운영하고 기반 시설이나 환경 지원은 미비한 상황이다[16].

2-3 컴퓨팅 사고력 교육 방법

컴퓨팅 사고력 증진을 위한 코딩 교육 방법으로는 피지컬 컴퓨팅(physical computing), 언플러그드(unplugged), 블록 기반 교육용 프로그래밍언어(EPL: Educational Programming Language), 그리고 파이썬, C, 자바 등의 텍스트 기반 고급언어 교육 등이 있다[17]-[18]. 피지컬 컴퓨팅은 주로 초등학교에서 사용하는 방식으로 학생들이 직접 손으로 만질 수 있는 로봇이나 블록 또는 회로를 통해 배우는 코딩 학습이다. 언플러그드 교육은 뉴질랜드의 팀 벨(Tim Bell) 교수가 개발한 것으로 컴퓨터를 사용하지 않고 학생들이 직접 연필과 종이, 퍼즐이나 카드, 보드게임과 같은 다양한 도구 및 활동 방식을 통해 논리적으로 사고하는 방법을 학습하는 방식이다. 손쉬운 블록 프로그래밍 도구인 스크래치나 엔트리 등의 교육용 프로그래밍언어는 초등에서 대학까지 다양하게 사용하는 교육 방식[19-22]이다. 대학에서는 컴퓨팅 사고력을 위한 코딩 교육으로 파이썬이 주로 사용된다.

Ⅲ. 소프트웨어 분석설계 교과목 운영

3-1 교과목 개요

서울 소재 D 대학의 컴퓨터 정보공학과는 컴퓨터의 활용 분야인 인터넷 시스템과 모바일 앱 개발 분야의 기초지식 및 실무이론 교육을 바탕으로 소프트웨어 개발과 스마트웹, 모바일앱 개발 분야의 현장 실무형 인력 양성을 목표로 한다. 2학년 1학기에 개설되는 '시스템분석설계' 교과는 과정 마지막 2학년 2학기의 '졸업작품'과 연계되어 1년 과정으로 운영되고 있다. 본 교과는 학생이 직접 대학과정에서 배운 정보통신 지식을 기반으로 실무와 관련이 깊은 소프트웨어 중심 프로젝트를 선정하여 개발할 아이템의 시장 환경 이해, 시스템 분석과 설계의 실습으로 구성된다. 그 이후의 프로젝트 구현, 테스팅 및 배포 등의 개발과정은 '졸업작품'에서 수행한다.

3-2 교과목 내용

일반적으로 시스템분석설계는 소프트웨어 중심의 시스템이 수행해야 할 기능과 시스템 구성요소가 어떻게 구현되고 작동해야 하는지를 이해하고 구체적으로 설계함으로써 소프트웨어 개발을 계획하는 것을 다룬다. 프로젝트 수업인 본 교과목은 개발할 시스템 선정에서 시스템 분석·설계와 프로토타입 제작, 그리고 프로젝트 발표까지의 과정을 수행한다. 프로젝트의 내용은 소프트웨어 개발 중심의 개발이라면 제한이 없다.

3-3 소프트웨어 분석설계 운영 사례

1) 2018년 1학기 교과 운영

본 연구는 소프트웨어 개발과정에서 구현 이전 과정을 경험하는 프로젝트 수업으로, 2018년 2학년 1학기의 '시스템분석설계' 교과에서 진행되었다. 이 과목은 재직자를 위한 교육과정으로 야간에 진행되며, 대상 학생 수는 14명이었다. 재직자이긴 하나 정보통신 관련 재직자는 1명에 불과하며, 이러한 시스템 분석설계를 경험한 학생은 없었다. 본 교과에서는 다양한 산출물을 요구한다. 강의가 시작되는 첫 주에 학생 개인에게 개인별로 '팀 프로젝트 기획 제안' 과제가 제시되며, 바로 2주차에 과제를 제출하게 한다. 학기 초기에 본 교과의 내용을 파악하지 못한 학생에게 부여되는 '팀 프로젝트 기획 제안'은 선정 아이템에 대해 심사숙고할 기회를 주기 위한 과정으로 이후 팀 프로젝트 선정과 프로젝트 추진에 많은 도움을 준다.

2) 교과목 평가

평가는 팀 평가와 개인 평가로 나뉘며, 팀 평가는 수업 일정에서 부여되는 단계별 산출물과 발표로 구성된다. 개인 평가는 팀 프로젝트 기획 제안, 미니프로젝트 수행, 필기시험, 개인 포트폴리오 작성, 출석과 함께 팀 내 기여도도 포함하여 시행하였다. 개인 평가 중에서 필기시험은 프로젝트 분석설계를 위한 최소한의 소프트웨어 공학 기초와 분석설계 방법론, 그리고 컴퓨팅 사고력 절차에 대한 평가가 포함된다. 팀 활동에서 개인별 기여도는 교수와 팀 구성원이 2회 이상 평가하고 개인에게 피드백을 주어 학생의 지속적인 프로젝트 참여를 유도한다. 프로젝트 수행 교과이지만 개인 평가를 병행함으로써 수업 참여를 유도하였다. 최종평가에는 담당 교수 외 1명이 더 참가하여 평가하였고 학생들도 자기 팀을 포함하여 다른 팀도 평가하였다.

Ⅳ. 소프트웨어 분석설계 교육 모델

4-1 컴퓨팅 사고력 기반 분석설계 교육

본 연구에서 제안하는 소프트웨어 분석설계 교육모델은 다음을 고려하여 설계되었다. 본 사례에서는 대상인 학생이 주간에 회사에 근무하는 재직자이면서 2년제의 교육과정인 것을 고려하여 분석설계의 다양한 기법은 최소한으로 구성되었다. 본 교육 모델에서는 분석설계 교과의 기본 절차를 기반으로 팀 프로젝트에서 개인 참여를 활성화하기 방안으로 개인별로 '팀 프로젝트 기획 제안'을 시행하며, 소프트웨어 구현을 보완하고 코딩에서의 컴퓨팅 사고력 함양을 위해 3~4주 기간에 구현까지 수행하는 작은 규모의 개인별 '미니 프로젝트'를 수행하도록 한다.

- 프로젝트팀 구성과 프로젝트 아이템 선정
- 요구사항 분석과 소프트웨어 기능 분석과 설계
- 소프트웨어 프로토타입 개발
- 개인별 팀 프로젝트 기획 제안
- 미니 프로젝트 수행

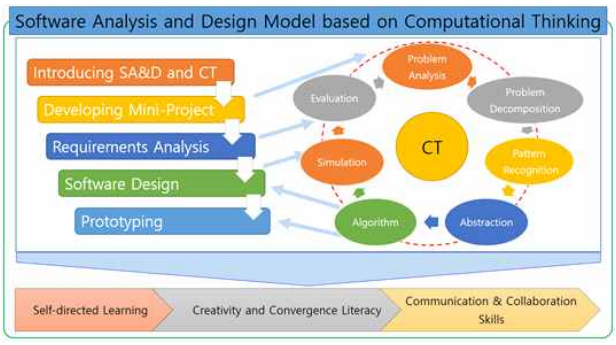

그림 1. 소프트웨어 분석설계 교육 모델 설계 개념
Fig. 1. Software A&D Education Model Design Concept

분석설계의 단계별 활동 내에 컴퓨팅 사고력 함양을 위한 요소를 추가하는 방식으로 모델을 제안한다. 그림 1과 같이 학생들은 미니프로젝트 수행과 소프트웨어 분석설계에서 컴퓨팅 사고력의 절차인 문제분석, 문제분해, 패턴인식, 추상화, 알고리즘, 시뮬레이션, 평가 과정을 수행하여 컴퓨팅 사고력 교육을 기반으로 한다.

본 컴퓨팅 사고력 기반 소프트웨어 분석설계 교육모델을 통해 학생의 자기 주도 학습과 창의융합, 의사소통과 협업 능력을 배양하도록 하며, 분석설계에서의 다음의 구체적인 성과도 목표로 한다.

- 졸업 후 현장에서 필요한 아이디어 도출을 위한 프로젝트 아이템 선정
- 소프트웨어 개발에 필요한 시스템 요구분석 및 설계 과정의 경험 습득
- 프로젝트 수행과정에서의 문서 작성과 발표 능력 습득

4-2 컴퓨팅 사고력 기반 교육 요소

본 연구에서 제안하는 '시스템분석설계(SA&D)' 교수·학습 교육 모델은 4차 산업혁명 패러다임이 요구하는 인재교육을 위해 컴퓨팅 사고력의 이해와 증진을 통한 학생의 자기 주도 학습, 창의·융합 사고능력, 의사소통과 협업 능력 등의 핵심 능력 향상에 주안점을 두었다. 이를 위하여 본 연구의 모델은 그림 2의 15주 학습 내용과 교육 효과를 위해 진행하였으며, 컴퓨팅 사고력 향상 목표를 달성하기 위해 7개의 컴퓨팅 사고력 구성요소를 포함하도록 구성하였다.

1) 자기 주도 학습

프로젝트팀은 기본적으로 3~5명의 학생으로 구성된다. 팀 구성 방법은 자기 주도 학습 차원에서 학생들 스스로가 구성하도록 한다. 다만 형평성을 위해 최소한의 조정이 있을 수 있다. 또한, 자기 주도적으로 3~4주에 수행되는 '미니 프로젝트'는 간단한 모바일 웹 및 앱 프로그램 또는 간단한 게임 및 콘솔 프로그래밍 등의 과제로 본 프로젝트에 앞서 설계에서 구현까지의 과정을 먼저 경험하도록 하며, 코딩을 통해 컴퓨팅 사고력을 함양한다. 팀 프로젝트 수업의 가장 큰 문제는 학생의 참여 부족으로 팀이 구성되면 소수의 학생이 주도한다는 점이다. 본 교과에서 자기 주도적인 학습을 위해 개인에게 첫 주부터 팀 프로젝트 기획 제안 과제와 몇 주 후에 미니프로젝트를 부여하고, 학습 내용을 정리한 자료와 자신과 팀에서 작성한 발표자료 및 형상자료의 모음인 개인 포트폴리오 과제도 부여하여 적극적으로 수업에 참여하도록 유도한다. 또한, 격주마다 시행하는 '프로젝트 진행 회의'를 통해 팀원의 역할을 주지시키고 프로젝트에 적극적으로 동참하도록 지속적인 관리를 한다.

2) 창의융합 사고능력

창의성은 '새롭고, 독창적이고, 유용한 것을 만들어 내는 능력' 또는 '전통적인 사고방식을 벗어나 새로운 관계를 창출하

Semester Schedule	Software A&D with CT	Lecture Plan with CT	Effects of Education
Introducing to SA&D with CT Week 1 ~ 4	• Project Teaming • Choosing Project Items • Data Collection(CT) • Data Analysis(CT) • Data Representation(CT)	• Introducing SA&D and CT • Creating Individual Proposal for Project • Assigning Individual Mini-Project for 3~4 weeks • Forming a few teams of three to five students • Presentation Project Plan • Team Meeting	• Understanding SA&D and CT • Understanding the Importance of CT in the 4th Industrial Revolution • Learning data collection, data analysis, data representation of CT • Experiencing Self-directed learning with proposal and Mini-Project
Preparing Project & Developing Mini-Project Week 5 ~ 8	• Requirements Analysis • Decomposition(CT) • Abstraction(CT) • Pattern Recognition(CT) • Algorithm(CT)	• Learning theoretical techniques of SA&D • Preparing Project with SA&D and CT • Implementing Individual Mini-Project • Presentation Requirements Analysis of Project • Team Meeting	• Understanding processes of SA&D • Experiencing the whole processes of SA&D with the Mimi-Project • Experiencing Communication and Collaboration skills
Software Analysis Week 9 ~ 10	• Software Analysis • Decomposition(CT) • Abstraction(CT) • Pattern Recognition(CT) • Algorithm(CT)	• Presentation Mini-Project • Learning theoretical techniques of Software Analysis • Creating Software Analysis of Project • Presentation Software Analysis of Project • Team Meeting	• Experience Collaboration of team • Understanding Software Analysis • Understanding Decomposition, Abstraction, Pattern Recognition Algorithm process of CT
Software Design & Prototyping Week 11 ~ 13	• Software Design • Prototyping • Automation(CT) • Simulation(CT)	• Making personal portfolio for students • Creating Software Design • Software Prototyping • Team Meeting	• Understanding Software Design • Understanding Generalization, Abstraction, Algorithm and Procedure process of CT
Presentation Final Report of SA&D & Assessment Week 14 ~ 15	• Presentation • Implementation Preparation • Evaluation(CT)	• Presentation final Report of SA&D with Prototype • Assessment of personal portfolio • Assessment of Team and Team fidelity for a student	• Cultivating Creativity and Convergence literacy through presentation • Opportunity for Introspection

그림 2. 컴퓨팅 사고 기반의 소프트웨어 분석설계 교육 모델의 교육 일정
Fig. 2. Schedule of Software Analysis and Design Education Model based on Computational Thinking

거나, 비일상적인 아이디어를 산출하는 능력'을 말한다. 이러한 창의성은 의식적 사고, 노력뿐만 아니라 무의식적인 사고와 노력의 영향을 받아 일어난다고도 한다. 창의성은 다양한 사고 유형이 총체적으로 결합하여 나타나는 가장 고차적인 사고능력으로 간주한다. 그러므로 창의성이란 단번에 어떤 기발한 아이디어를 내는 것이 아니라, 개인과 더불어 여러 사람이 모여서 다양한 문제들을 해결해 나가는 탐구 활동이라고 볼 수 있다. 연구에 의하면 창의성의 첫 번째는 문제, 정보의 격차, 빠진 요소, 무언가 잘못된 것을 감지하는 과정이며, 두 번째는 이러한 부족한 점이나 빠진 요소에 대하여 추측해 보거나 가설을 세우는 단계이고, 세 번째는 추측이나 가설을 검증하거나 수정한 다음 다시 검증하고, 네 번째로 그 결과를 다른 사람에게 전달하는 의사소통 과정의 네 가지 과정으로 설명한다[23]. 바로 소프트웨어 분석설계의 전 과정이 바로 창의성의 과정과도 일치한다. 작게는 미니프로젝트 진행 과정도 창의성의 과정과도 일치하며, 팀원과의 회의를 통한 의사소통과 격주마다 발표와 '프로젝트 진행 회의'도 창의성에 긍정적인 영향을 미친다.

3) 의사소통과 협업 능력

교과과정에서 팀이 구성된 약 3주차 이후, 교수자는 팀별로 격주마다 '프로젝트 진행 회의'를 갖는다. 이 회의에서는 프로젝트의 진척 보고, 구성원의 역할, 소프트웨어 공학 및 구현 기술, 프로젝트 진행의 애로점 등을 학생과 함께 논의한다. 그리고 개인 프로젝트 기획 제안 발표, 팀 프로젝트 주제 발표, 개인 미니프로젝트 계획 및 결과 발표, 프로젝트 수행 계획서, 시스템 환경 및 요구사항 분석서, 시스템 분석 및 설계서, 시스템 프로토타입 개발 발표 등 프로젝트 수행과 거의 격주로 시행되는 발표를 통하여 팀원 구성원 간의 의사소통과 협업이 이루어지도록 한다.

4-3 교육 효과를 위한 평가

본 교과목은 팀 프로젝트가 주요 내용이나 개인의 참여와 자기 주도 학습의 효과성을 높이고자 팀 프로젝트에도 개인의 기여도를 평가하고 개인 제안서와 미니프로젝트 평가, 중간고사 등도 실시하여 팀 평가 45%와 개인 평가 55%로 구성된다. 구체적인 평가의 항목은 표 3과 같다.

표 3. 평가 요소
Table 3. Evaluation Unit

Evaluation unit	Team	Individual	Schedule
Individual proposal for Project	-	5	week 2
Individual proposal for Mini-Project	-	3	week 4
Individual presentation for Mini-Project	-	7	week 9
Team proposal for Project	5	10	week 3
Team presentation for Project Plan	5		week 2
Team presentation for Requirements Analysis of Project	10		week 9
Team presentation for Software Analysis and Design of Project	10		week 12
Team final presentation for Software Analysis and Design of Project	15		week 15
Midterm Examination	-	10	week 8
Attendance Marks	-	20	-
Subtotal	45	55	-
Total	100		-

Ⅴ. 결 론

세계는 지금 4차 산업혁명 시대를 맞아 소프트웨어가 혁신과 성장, 가치창출이 중심이 되고 개인·기업·국가의 경쟁력을 좌우하는 소프트웨어 중심사회로 급속히 발전하고 있다. 이러한 배경에서 미래의 창의적인 인재에게 필요한 교육 중의 하나로 소프트웨어 교육을 꼽을 수 있다. 특히 과거의 컴퓨터 및 사무자동화 소프트웨어의 활용을 넘어 코딩 교육에 의한 컴퓨팅 사고력 교육의 중요성이 커지고 있다. 컴퓨팅 사고력이란 컴퓨팅의 기본 개념과 원리를 기반으로 문제를 효율적으로 해결하는 사고능력을 말한다. 이러한 컴퓨팅 사고력을 함양하는 여러 방법의 하나는 직접 프로그래밍을 하는 코딩 교육이다.

본 연구에서는 연구 배경으로 컴퓨팅 사고력에 개념과 요소를 살펴보고 대학교육에서 컴퓨터 전공자에게 필요한 컴퓨팅 사고력 기반의 소프트웨어 분석설계 교수·학습 모델을 제시하였다. 컴퓨터공학 분야 정규교과목 사례 연구를 통해 제안된 '시스템분석설계(SA&D)' 교수·학습 교육 모델은 프로젝트팀 구성과 프로젝트 아이템 선정, 요구사항 분석과 소프트웨어 기능 분석과 설계, 소프트웨어 프로토타입 개발 등의 소프트웨어 개발 전 과정을 학습하고 개인별 미니 프로젝트 수행뿐만 아니라 팀 활동에 의한 자신들의 프로젝트 수행을 통하여 학생 스스로 소프트웨어 개발과정을 체험하는 소프트웨어 분석설계 교육 모델이다. 본 교육 모델에서 개인에게 부여된 팀 프로젝트 기획 제안과 미니프로젝트 수행, 개인 포트폴리오 수행은 학생에게 적극적인 수업 참여를 유도하고 자기 주도 학습에 효과적이다. 또한, 팀 프로젝트와 미니프로젝트 진행 과정, 그리고 팀원과의 회의는 개인과 더불어 여러 사람이 모여서 다양한 문제들을 해결해 나가는 탐구 활동으로 다양한 사고 유형이 총체적으로 결합하여 나타나는 사고능력인 창의융합 사고능력에도 효과적이며, 팀 프로젝트 수행과정과 격주로 시행되는 발표는 팀원 구성원 간의 의사소통과 협업 능력 향상에도 효과적이다. 결론적으로 본 연구의 교수·학습 교육 모델은 컴퓨팅 사고력을 배양하여 자기 주도 학습, 창의·융합 사고능력, 의사소통과 협업 능력 등을 키우는 데 효과적이며, 프로그래머로서 분석설계 능력을 지닌 소프트웨어 개발자로 성장하는 데 도움을 줄 것으로 기대한다. 본 연구의 교수·학습 모델이 미래 인재의 핵심역량인 창의·융합 사고능력과 의사소통 및 협업 능력을 확보하는 데 중요한 전환점이 되기를 기대한다.

감사의 글

본 연구는 2017년도 동양미래대학교의 학술연구비 지원에 의하여 이루어진 연구로서, 관계부처에 감사드립니다.

참고문헌

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

[2] S. J. Lee, M. J. Lee and Y. S. Park, "A Study on a Educational Model for Computational Thinking Development with Excel Program", *Journal of Digital Contents Society*, Vol. 20, No 1, pp. 65-74, 2019.

[3] S. H. Kim, "Analysis of Non-Computer Majors` Difficulties in Computational Thinking Education," *The Journal of Korean Association of Computer Education*, Vol. 18, No. 3, pp. 49-57, 2015.

[4] M. J. Lee, "Exploring the Effect of SW Programming Curriculum and Content Development Model for Non-majors College Students: focusing on Visual Representation of SW Solutions," *The Journal of Digital Contents Society*, Vol. 18, No. 7, pp. 1313-1321, 2017.

[5] S. J. Lee and M. J. Lee, "Study of computer programming education paradigm for non-majors," *The Proceeding of Korean Association of Computer Education*, Vol. 21, No. 2, pp. 161-164, 2017.

[6] Y. S. Lee, "Python-based Software Education Model for Non-Computer Majors," *Journal of the Korea Convergence Society* Vol. 9, No. 3, pp. 73-78, 2018.

[7] Lu J. J, Thinking about computational thinking. ACM SIGCSE Bulletin, Vol. 41, No. 1, pp. 260-264, 2009.

[8] BCS, Call for evidence-UK Digital Skills Taskforce, BCS, http://policy.bcs.org.

[9] BBC. Introduction to computational thinking. BBC Bitesize. https://www.bbc.com/bitesize/guides/zp92mp3/revision/1.

[10] CSTA & ISTE, Computational thjnking in K-12 education teacher resource second edition, CSTA & ISTE, https://www.iste.org/docs/ct-documents/ct-teacher-resources_2ed-pdf.pdf?sfvrsn=2.

[11] KERIS, Training Textbooks for Software Leading Teachers, Elementary Schools KERIS, http://lib.keris.or.kr/search/detail/CAT000000013040.

[12] D. M. Kim and T. W. Lee, "A Meta-Analysis on the Effects of Software Education on Computational Thinking," *Journal of The Korea Society of Computer and Information*, Vol. 23, No. 11, pp. 239-246, 2018.

[13] J. S. Kim and S. K. Han, Development of SW Education Teaching and Learning Model, Technical Report CR 2015-35. Seoul : Korea Education Development Institute, Korea Education and Research Information Service, 2015.

[14] M. S. Lee, "A Study on Creative and Convergent SW Education Programs for improving Computational Thinking," *Journal of the Korea Society of Computer and*

Information, Vol. 22, No. 8, pp. 93-100, 2017.

[15] S. H. Park, "Study of SW Education in University to enhance Computational Thinking," *Journal of Digital Convergence*, Vol. 14, No. 4, pp. 1-10, 2016.

[16] S. Y. Choi, "A Study on the Digital Competency for the Fourth Industrial Revolution," *The Journal of Korean Association of Computer Education*, Vol. 21, No. 5, pp. 25-35, 2018.

[17] J. H. Park, S. Park. and Y. G. Seo, "Software Education and Business Model for Enforcing the Computational Thinking Ability," *Journal of Digital Contents Society*, Vol. 19, No. 12, pp. 2297-2304, 2018.

[18] J. Seo, "A Case Study on Programming Learning of Non-SW Majors for SW Convergence Education," *Journal of Digital Convergence*, Vol. 15, No. 7, pp. 123-132, 2017.

[19] S. B. Shin, "The Improvement Effectiveness of Computational Thinking through Scratch Education," *Journal of the Korea Society of Computer and Information*, Vol. 20, No. 11, pp. 191-197, 2015.

[20] Y. Sung, "Development of SW Education Model based on HVC Learning Strategy for Improving Computational Thinking," *Journal of The Korean Association of Information Education*, Vol. 21, No. 5, pp. 583-593, 2017.

[21] Y. N. Song, "Design and Implementation of Reflection-based Coding Education: Case Study of "SW and Computational Thinking Courses at H University," *Journal of Educational Technology*, Vol. 33, No. 3, pp. 709-736, 2017.

[22] Kwangil Ko, "A Study on the EPL using Instructional Model of SW Major's Programming Class," *The Journal of Digital Contents Society*, Vol. 19, No 5, 891-898, 2018.

[23] E. Paul Torrance, "Understanding Creativity: Where to Start?," *Psychological Inquiry*, Vol. 4, No. 3, pp. 232-234, 1993.

강환수(Kang, Hwan Soo)

1991년 : 서울대학교 대학원 (이학석사)
2002년 : 서울대학교 대학원 (공학박사수료-컴퓨터그래픽스)

1992년~1998년: 삼성에스디에스
1998년~현 재: 동양미래대학교 컴퓨터정보공학과 교수
※관심분야 : 컴퓨터교육, 객체지향, 프로그래밍언어

조진형(Cho, Jin Hyung)

1999년 : 서울대학교 대학원 (공학석사)
2007년 : 서울대학교 대학원 (공학박사-기술경영)

1999년~현 재: 동양미래대학교 컴퓨터정보공학과 교수
※관심분야 : 협업 필터링(Collaborative Filtering), 추천 시스템(Recommender System)

김희천(Kim, Hee Chern)

1991년 : 서울대학교 대학원 (이학석사)
1998년 : 서울대학교 대학원 (이학박사-소프트웨어 공학)

2004년~현 재: 한국방송통신대학교 전산과학과 교수
※관심분야 : 컴퓨터교육, 소프트웨어 공학