Journal of The Korean Association
of Information Education Vol. 20, No. 6,
December 2016, pp. 553-562

http://dx.doi.org/10.14352/jkaie.2016.20.6.553

예비교원의 Computational Thinking(CT) 역량 계발 방안 : CT의 5가지 핵심 역량 분석

최형신

춘천교육대학교 컴퓨터교육과

요 약

소프트웨어 교육이 학습자의 Computational Thinking(CT) 역량 계발을 지향하고 있지만 CT역량 계발을 위해 수업을 설계하고 CT역량 계발 효과를 측정하는 것에는 아직 많은 연구가 필요한 실정이다. 본 연구의 목적은 예비교원의 CT 역량을 계발하기 위해 CT기반으로 설계된 교재를 활용하여 SW교육을 진행하고 본 수업이 예비교원의 CT 역량에 미치는 영향을 알아보는 것이다. 본 연구에서는 문헌 연구를 통해 CT의 다섯 가지 주요 세부 역량을 알고리즘적 사고, 평가, 분해, 추상화 및 일반화로 설정하였다. 본 연구의 대상은 국내 한 교육대학교에서 한 학기 동안 수업을 수강한 47명의 예비교원이며, CT관련 설문을 개발하여 실시하고 팀프로젝트를 CT의 세부 핵심 역량에 초점을 두고 질적으로 분석하여 연구 결과를 도출하였다. 본 연구의 결과는 예비교사가 지각한 CT경험과 프로젝트를 통해 나타난 CT세부 핵심 역량 측면의 질적 분석을 통해 향후 교원양성기관에서의 소프트웨어 교육의 교수설계 및 운영에 대해 시사점을 제공하고 있다는 점에서 의의가 있다.

키워드 : 컴퓨팅 사고력, 추상화, 알고리즘적 사고, 문제분해, 스크래치, 예비교원

Developing Pre-service Teachers' Computational Thinking : Analysis of the Five Core CT Competencies

Hyungshin Choi

Dept. of Computer Education, ChunCheon National University of Education

ABSTRACT

Although software education pursues developing learners' computational thinking skills, more studies are needed in terms of designing software education lessons to enhance CT skills and to measure the effects of the lessons. This study aims to investigate the effects of a course designed to enhance pre-service teachers' CT skills by using CT-based teaching materials. Through a literature review the study has selected the five core competencies of CT: algorithmic thinking, evaluation, problem decompositions, abstraction, and generalization. The participants of the study are 47 pre-service teachers who took the one-semester course in a national university of education. A survey was developed and conducted and qualitative analyses on the team projects were performed focusing on the core competencies of CT. The results revealed pre-service teachers' perceived degree of experiencing CT and their competencies represented in their projects. The present study provides important implications to future soft-

이 논문은 2015년도 춘천교육대학교 교내연구비 지원에 의하여 연구되었음.
논문투고 : 2016-11-07
논문심사 : 2016-11-07
심사완료 : 2016-11-30

ware education programs in terms of designing and implementing of software education.

Keywords : Computational Thinking, Abstraction, Algorithmic Thinking, Problem Decompositions, Scratch, Pre-service Teachers

1. 서론

소프트웨어 중심사회의 도래와 함께 프로그래밍 교육의 필요성이 크게 인식되었고 우리나라에서도 2019년부터 소프트웨어 교육이 초등현장에서 시행될 것이다[15]. 공교육에서 프로그래밍 교육을 도입하는 이유 중 하나는 미래에 더욱 더 요구되는 Computational Thinking(CT) 능력을 학생들에게 갖추게 하자는데 있다. CT는 큰 범주에서는 문제해결의 사고 과정이며, CT와 일반적인 문제해결과의 차이는 CT가 컴퓨터를 비롯한 정보처리체계가 가지는 컴퓨팅의 파워를 문제해결력에 연결시킬 수 있는 역량이라는 점이다[10]. CT는 Papert에 의해 처음 설명된 개념이며[11], Wing에 의해 학계에 부각되었다[13][14]. 학자들에게 따라 다소 다른 세부 요소를 포함시키거나 배제하는 차이를 보이고 있기는 하지만 근본적으로 프로그래밍 교육이 CT역량에 미치는 영향을 긍정적으로 평가하고 있다.

그러나 프로그래밍 교육이 CT역량에 긍정적 영향을 미친다는 가설을 입증하기 위해서는 프로그래밍 교육의 세부 내용이 CT의 세부 역량과 어떻게 연결되고 있는지 파악하고 이를 교육 및 평가해야 한다. 초등학생 대상의 프로그래밍 도구로 활용되고 있는 스크래치의 프로그래밍 교재가 많이 나오고 있지만 CT의 세부 요소에 초점을 두고 개발된 교재는 많지 않은 실정이다. 따라서 본 연구는 최근에 개발된 CT기반의 프로그래밍 교육 교재와 연구자가 개발한 교육 자료를 활용하여 CT의 세부 역량에 미치는 영향을 확인해 보고자 하였다[6]. 본 연구에서는 초등예비교원을 대상으로 한 학기 수업을 진행하고 본 수업이 예비교원의 CT역량에 미치는 영향을 양적 및 질적으로 조사하였다.

2. 이론적 배경

2.1 컴퓨팅 사고(CT)의 핵심 역량

Wing은 CT를 '문제를 구성하고 그 문제의 해결책을 정보처리 체계에 의해 효과적으로 수행될 수 있는 형태로 제시하는 사고 과정'으로 정의하였다[14]. 이러한 정의를 바탕으로 Selby는 문헌 연구를 통해 CT의 세부 요소를 다섯 가지로 제시하였는데 이는 '알고리즘적 사고(algorithmic thinking)', '평가'(evaluation), '문제분해'(problem decomposition), '추상화'(abstraction), 그리고 '일반화'(generalization)이다[12]. 알고리즘적 사고는 명확한 단계들을 통해 해결책에 이르는 방법이다. 따라서 사람이나 컴퓨터가 그 규칙이나 명령을 따라가면 그 문제나 유사한 문제의 답에 이르게 되는 것을 말한다. 평가란 알고리즘적 해결책이 목적에 적합한 것인지 판단하고 확인하는 과정을 말하는데 문제해결을 위해서 해당 상황에 따라 어떤 알고리즘이 적합한지 정확성, 속도, 자원 사용면 등 여러 측면에서 평가하는 역량을 말한다. 문제분해는 문제, 알고리즘, 산출물, 과정, 시스템 등에 대해 이를 그 부분으로 생각하는 방법이다. 분해된 문제는 개별적으로 이해되고 해결될 수 있으며 이는 복잡한 문제를 보다 쉽게 해결할 수 있는 방법이 된다. 추상화는 문제나 시스템을 보다 쉽게 생각하도록 하는 또 다른 방법으로서 불필요한 세부 사항을 숨기고 꼭 필요한 부분을 드러나게 표현하는 방법이다. 끝으로 일반화는 새로운 문제를 해결할 때 이전에 사용했던 알고리즘이나 해결책을 기반으로 보다 빨리 문제를 해결하는 것을 의미한다.

Barendsen 등은 CT능력을 측정하는 도구인 Bebras 검사[4]에서 CT 요소와 정의를 제시하였다[2]. 제시된 CT요소는 아홉 가지로 자료 수집, 자료 분석, 자료 표현, 문제 분해, 추상화, 알고리즘, 자동화, 병행화, 시뮬레이션을 포함한다. 자료 수집은 적절한 정보를 수집하

는 과정이며 자료 분석은 자료의 의미를 만들고 패턴을 발견하여 결론을 도출하는 것으로 정의하였다. 자료 표현은 자료를 적절한 그래프, 차트, 단어, 이미지로 묘사하고 조직하는 것으로 정의하였으며 문제 분해는 과제를 더 작고 관리 가능한 부분들로 나누는 것으로 정의하였다. 추상화는 복잡성을 줄이고 주된 아이디어를 정의하는 것으로 정의했으며 알고리즘은 문제를 해결하거나 어떤 목적을 달성하기 위해 순서화된 단계를 만드는 것으로 정의했다. 자동화는 컴퓨터나 기계가 반복적이거나 사소한 작업을 하는 것으로 정의했으며 Bebras 검사에서는 다뤄지고 있지 않다고 분석하였다. 병행화는 공통의 목적을 가지고 과제들이 동시에 수행되도록 자원을 조직하는 것으로 정의했으며 시뮬레이션은 과정의 제시나 모델로 정의하면서 모델을 사용하여 실험하는 것을 포함한다고 하였다. Selby[12]가 제시한 다섯 가지 요소에서 문제분해, 추상화, 알고리즘이 공통으로 포함되고 있음을 알 수 있다.

2.2 CT역량 평가 및 계발에 관한 연구

Lye와 Koh는 CT역량 평가관련 선행 연구들을 리뷰한 결과 대부분의 연구가 순차, 반복, 조건, 이벤트, 자료, 연산 등 CT 개념 평가에 국한되어 있다고 보고하였다[9]. 또한 CT의 주요 세부 요소로 여겨지고 있는 알고리즘적 사고, 평가, 추상화, 문제분해, 일반화 능력에 미치는 소프트웨어 교육의 영향에 초점을 둔 연구는 매우 부족한 실정이다.

최근에 예비교사의 CT역량 계발을 위한 연구가 이루어지기 시작하였다. Bean 등은 CT와 프로그래밍을 음악, 언어, 미술, 수학, 과학 등의 과목에서 교수적 도구로 활용하기 위한 교육 프로그램을 개발하여 수행한 결과 예비교사들의 CT 자기 효능감을 높이는 효과가 있음을 보고하였다[3]. 또한 An과 Lee는 예비교사의 CT 능력 향상을 위해 컴퓨터과학, 알고리즘, 프로그래밍 언어의 원리와 개념을 포함하여 60시간의 교육 프로그램을 개발하였다[1]. 한편 전수진과 한선관은 예비교사의 CT역량 평가를 위해 실험하기 및 반복하기, 테스팅과 디버깅, 재사용과 재구성, 추상화와 모듈화를 포함하는 서술형 수행평가 도구를 개발하였다[5]. 재사용과 재구성은 Selby가 제시한 다섯가지 핵심 역량 중 일반화에 근접하며, 추상화와 모듈화는 추상화와 문제분해와 유사한 역량으로 연결지을 수 있다[12].

3. 연구 방법

3.1 연구 대상 및 절차

본 연구의 연구 대상은 국내 한 교육대학교 'Computational Thinking을 활용한 문제해결'과목 수강자 47명의 예비교원이다. 한 학기의 수업에서 CT역량 계발 SW교육을 진행하고, 2인 1팀으로 스스로 기획한 프로젝트를 제작하여 발표하도록 하였다. 개발된 CT 설문은 학기말에 진행했으며, 학생들의 팀프로젝트(보고서, 스크래치 어플리케이션)를 CT의 세부 역량을 초점으로 분석하였다.

3.2 CT역량 계발 SW교육 프로그램

본 연구를 진행한 수업의 내용은 CT를 기반으로 개발된 소프트웨어 동아리 활동 교재와 교사지도서의 내용과 연구자가 개발한 교육 자료로 구성하였다[7][8]. 사용된 교재의 특징은 프로그래밍에 필요한 기능을 모두 설명하는 메뉴식 구성이 아니라 CT기반 사고를 할 수 있도록 학습할 부분을 문제의 형태로 제시하고 문제를 해결하기 위한 절차를 생각하고 이를 줄거리로 구성하고 추상화할 수 있도록 하는 것이다. 본 교재는 4개의 단원으로 구성되었는데 I 단원은 스크래치 프로그래밍 방법에 대한 설명, II 단원과 III 단원은 게임 등과 같이 문제 해결을 위한 프로그램을 작성하는 주제로 구성되어있다. IV 단원은 개발한 프로그램을 발표하는 방법에 대한 설명을 담고 있다[6]. 특히 교사지도서를 활용하여 향후 초등현장에서 SW교육을 할 때 초점을 두어야 할 사항과 유의점 등을 교육하였다.

수업의 후반부에는 2인 1팀으로 게임이나 애니메이션 등과 같은 디지털 콘텐츠를 프로젝트로 기획하고 구현하여 발표하는 과정으로 구성했다. 프로젝트를 발표하고 보고서를 제시할 때 CT의 주요 요소 즉 알고리즘적 사고, 알고리즘의 평가, 문제 분해, 추상화, 일반화 영역으로 구성하도록 함으로써 학생들의 사고과정을 외현적으로 드러나도록 하였다.

3.3 자료 수집

3.3.1 CT 설문지

본 연구에서는 선행 연구에서 사용된 다섯 가지 CT 개념-알고리즘적 사고, 평가, 문제 분해, 추상화, 일반화-을 가지고 설문을 구성하여 사용하였다[12]. 본 설문은 세 영역으로 구성되었는데 첫째는 본 수업 내용을 통해 다섯 가지 CT 세부 역량을 활용하는 경험의 정도를 질문하였으며, 두 번째 영역에서는 향후 초등학생에게 프로그래밍 교육을 수행할 때 다섯 가지 CT 세부 역량 교육에 대한 CT교육 자기효능감을 질문하였다. 끝으로 세 번째 영역에서는 본 수업에서의 경험이 다른 문제 해결 상황에서도 다섯 가지 CT역량을 활용할 수 있도록 할 것인가에 대한 질문으로 구성하였다. 설문 문항은 4점 척도(전혀 그렇지 않다, 약간 그렇지 않다, 약간 그렇다, 매우 그렇다)로 구성되었다.

3.3.2 팀프로젝트

본 연구에서 수집한 팀프로젝트 관련 자료는 2인 1팀으로 개발한 스크래치 프로젝트, CT기반으로 구성된 보고서, 개인별 1장 분량의 성찰일지이다. CT기반으로 구성된 보고서 템플릿의 내용은 (1) 문제 정의 (2) 스프라이트, 배경, 변수 설명 (3) 문제 해결 절차 (4) 문제 해결 절차 평가 (5) 문제 분해(추상화) (6) 분해된 문제의 세부 내용 (7) 일반화 (8) 팀 역할 분담 및 성찰일지이다.

4. 연구 결과

4.1 예비교원 CT 설문 결과

수업의 수강자 62명 중 본 CT설문에 참가한 예비교원은 총 47명이었으며, 여학생 37명(79%)과 남학생 10명(21%)으로 구성되었다. 사전 프로그래밍 경험 유무를 확인한 결과 사전 경험이 있는 학생이 10명(21%)이었고, 사전 경험이 없는 학생이 37명(79%)이었다. 또한 연령별로 보았을 때 20세~25세가 39명(83%)이었고, 25세~30세가 8명(17%)이었다.

한 학기 동안 CT 역량 계발을 위한 소프트웨어 교육을 진행한 뒤 학생들에게 자기 평가 CT설문을 실시한 결과는 다음과 같이 나타났다.

4.1.1 CT 경험 정도

첫째는 본 수업에서 다섯 가지 CT 세부 역량을 활용한 경험의 정도를 질문하였는데 전반적으로 긍정적인 반응을 하였다<Table 1>. 97.9%(46명)의 응답자가 알고리즘적 사고를 경험하였다고 보고하였으며, 100%(47명)의 응답자가 평가와 일반화를 경험한 것으로 보고하였다. 91.5%(43명)의 응답자가 문제분해를 경험하였다고 하였으며, 89.4%(42명)의 응답자가 추상화를 경험하였다고 보고하였다. 한편 10.6%(5명)의 응답자는 추상화를

<Table 1> Degree of CT Experiences (n=47)

In this course, I've experienced the following:	*Strongly disagree*	*Disagree*	*Agree*	*Strongly agree*	*Mean*	*SD*
I thought algorithmically as I solved the problems.	-	1 (2.1%)	19 (40.4%)	27 (57.4%)	3.55	0.54
I thought in terms of evaluation as I checked to see if my algorithms are appropriate to solve the problems.	-	-	25 (53.2%)	22 (46.8%)	3.47	0.50
I thought in terms of decomposition as I broke problems into smaller problems.	-	4 (8.5%)	23 (48.9%)	20 (42.6%)	3.34	0.64
By hiding complex parts and using abstraction solving problems was getting easier.	-	5 (10.6%)	27 (57.4%)	15 (31.9%)	3.21	0.62
I solved new problems by incorporating the algorithms that I used for previous problems.	-	-	18 (38.3%)	29 (61.7%)	3.62	0.49

잘 경험하지 못했다고 보고하였고 8.5%(4명)는 문제분해를 잘 경험하지 못했다고 보고하였다. 알고리즘적 사고(Mean=3.55, SD=0.54), 평가(Mean=3.47, SD=0.50), 일반화(Mean=3.62, SD=0.49)에 비해 추상화(Mean=3.21, SD=0.62)와 문제분해(Mean=3.34, SD=0.64)가 상대적으로 학생들에게 잘 수용되지 않는 것으로 나타났다.

4.1.2 CT교육 자기효능감

두 번째 영역에서는 향후 초등학생에게 소프트웨어 교육을 수행할 때 다섯 가지 CT 세부 역량을 교육할 수 있다고 생각하는지 즉, CT교육 자기효능감을 질문하였다. 97.9%(46명)의 응답자가 일반화를 교육할 수 있다고 보고하였으며, 95.7%(45명)의 응답자가 알고리즘 평가를 교육할 수 있다고 보고하였다<Table 2>. 또한 87.2%(41명)가 알고리즘적 사고를, 89.4%(42명)가 추상화를, 85.1%(40명)가 문제분해를 교육할 수 있다고 보고하였다. 한편 14.9%(7명)의 응답자는 문제분해를 잘 교육하지 못할 것 같다고 응답했으며, 12.8%(6명)는 알고리즘적 사고를, 10.6%(5명)는 추상화를 잘 교육하지 못할 것 같다고 보고 하였다. 전반적으로 볼 때, 일반화(Mean=3.38, SD=0.53), 평가(Mean=3.28, SD=0.54)에 비해 문제분해(Mean=3.15, SD=0.66), 추상화(Mean=3.19, SD=0.68), 알고리즘적 사고(Mean=3.19, SD=0.65) 영역에서 상대적으로 예비교사의 CT교육 자기효능감이 낮은 것으로 나타났다. 일반화는 기존에 있는 코드나 알고리즘을 다른 목적을 위해 수정하여 사용하는 것으로 예비교사들이 이를 지도하는데 상대적으로 높은 자기효능감을 보였고 알고리즘을 평가하는 것도 프로그램 수행 후 오류를 디버깅할 수 있기 때문에 평가 지도에 상대적으로 자신감이 있었다고 판단된다. 그러나 문제 분해와 추상화에서 CT교육 자기효능감이 상대적으로 낮게 나왔는데 그 이유는 예비교사 자신에게도 상대적으로 어려운 영역이었기 때문에 교육하는 것에서도 낮은 자기효능감을 보인 것으로 판단된다.

4.1.3 CT 전이

세 번째 영역은 본 수업에서의 경험이 다른 문제 해결 상황에서도 다섯 가지 CT역량을 활용할 수 있는가 즉, 전이(transfer)에 대한 질문이었다. 설문 결과를 보면, 95.7%(45명)의 응답자가 일반화를 적용할 것이라고 보고하였으며, 93.6%(44명)가 알고리즘적 사고와 평가를 적용할 것이라고 보고하였다<Table 3>. 91.5%(43명)의 응답자가 문제분해를 적용할 것이라고 보고하였으며, 80.9%(38명)의 응답자가 추상화를 적용할 것이라고 보고하였다. 한편 19.1%(9명)는 추상화를 잘 적용하지 못할 것 같다고 했으며, 8.5%(4명)는 문제분해를 잘 적용하지 못할 것 같다고 보고하였다. 전반적으로 보면, 일반화(Mean=3.49, SD=0.59), 알고리즘적 사고(Mean=3.30, SD=0.59), 평가(Mean=3.28, SD=0.58)에 비해 추상화(Mean=3.13, SD=0.71)와 문제분해(Mean=3.26, SD=0.58)가 상대적으로 예비교사가 적용하기 힘든 영역으로 생각하는 것으로 나타났다. 이러한 결과는 일반화와 평가에 비해 추상화나 문제분해에 대한 교육이나 훈련이 더 필요하다는 것을 보이고 있다고 할 수 있다.

<Table 2> CT Teaching Self-Efficacy (n=47)

If I teach elementary students Scratch programming in the future, I would be able to help them:	*Strongly disagree*	*Disagree*	*Agree*	*Strongly agree*	*Mean*	*SD*
To solve problems with algorithmic thinking.	-	6 (12.8%)	26 (55.3%)	15 (31.9%)	3.19	0.65
To check to see if their algorithms are appropriate to solve the problems.	-	2 (4.3%)	30 (63.8%)	15 (31.9%)	3.28	0.54
To brake problems into smaller problems so that they could solve the problems easier.	-	7 (14.9%)	26 (55.3%)	14 (29.8%)	3.15	0.66
To use abstraction by hiding complex parts.	1 (2.1%)	4 (8.5%)	27 (57.4%)	15 (31.9%)	3.19	0.68
To solve new problems by incorporating the algorithms that they used for previous problems.	-	1 (2.1%)	27 (57.4%)	19 (40.4%)	3.38	0.53

<Table 3> CT Tansfer (n=47)

I would do the following when I solve other problems as well as I would solve the programming problems:	*Strongly disagree*	*Disagree*	*Agree*	*Strongly agree*	*Mean*	*SD*
I would solve problems thinking algorithmically.	-	3 (6.4%)	27 (57.4%)	17 (36.2%)	3.30	0.59
I would think in terms of evaluation as I check to see if my algorithms are appropriate to solve the problems.	-	3 (6.4%)	28 (59.6%)	16 (34.0%)	3.28	0.58
I would think in terms of decomposition as I brake problems into smaller problems.	-	4 (8.5%)	27 (57.4%)	16 (34.0%)	3.26	0.61
I would use abstraction by hiding complex parts to solve problems easier.	-	9 (19.1%)	23 (48.9%)	15 (31.9%)	3.13	0.71
I would solve new problems by incorporating the algorithms that I used for previous problems.	-	2 (4.3%)	20 (42.6%)	25 (53.2%)	3.49	0.59

4.2 팀프로젝트 및 성찰일지 분석 결과

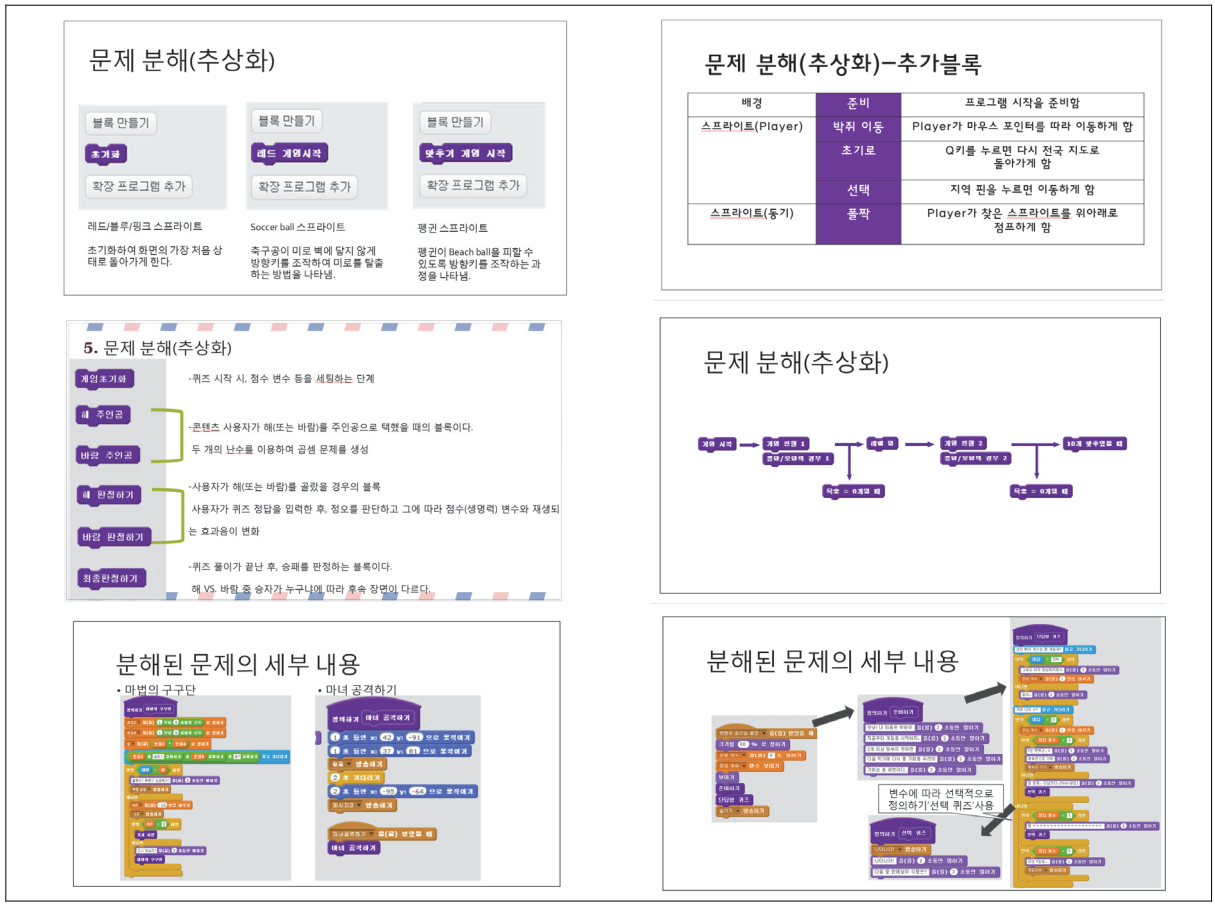
본 연구에서는 예비교사들의 CT세부 역량이 팀프로젝트 과제물을 완성해가는 과정에서 어떻게 발현되는지를 확인하기 위해 프로젝트 템플릿을 제공하고 템플릿에 팀과제를 정리해서 발표하도록 하였다. 분석 대상은 총 27팀의 스크래치 프로젝트와 CT기반 프로젝트 보고서와 성찰일지였다. CT의 다섯 가지 요소 중에서 설문 응답에서 다소 낮게 나타난 추상화와 문제분해 영역에 초점을 두고 분석해 보았다.

분석 결과, 33%(9팀)의 프로젝트에는 문제분해 및 추상화 작업이 나타나있지 않았다. 67%(18팀)의 프로젝트에서는 문제 분해와 추상화가 구현되었는데 그 정도는 다양하게 나타났다. 반복적으로 사용되는 블록을 묶어 한 두 개의 추가블록으로 만들어 재사용하는 용도로 활용한 프로젝트가 있는 반면 전체 프로젝트를 논리적인 덩어리로 분해하여 각 덩어리를 모두 추상화하여 구현한 프로젝트도 있었다. (Fig. 1)은 분석 프로젝트에서 문제분해 및 추상화를 구현한 사례를 보여주고 있다.

한편 프로젝트 성찰일지의 내용을 분석해 본 결과 스크래치 프로그래밍 프로젝트 제작을 통해 문제분해와 추상화의 사고 과정을 경험하고 그러한 경험이 사고력 향상과 문제해결을 쉽고 빠르게 한다는 것을 인식하는 것으로 나타났다(예비교사 성찰일지 스크립트 참조). 또한 이번에 사용한 프로젝트 템플릿은 예비교사들로 하여금 스크래치 프로젝트 완성 후에 다시 한 번 문제해결 과정 및 코딩 내용을 반성할 수 있는 기회를 가지게 하였고 논리적이고 알고리즘적 사고를 하는데 도움이 된 것으로 나타났다.

> “과제를 수행하며 내가 원하는 프로그램을 코딩하기 위해 스스로 영역별 항목을 적절히 조립하여 체계를 구축해나가는 것이 정말 힘들었지만 한편으로는 즐거웠다. 실패를 반복해서 코드를 따져 나의 실패원인을 분석하고 추상화, 일반화하는 동시에 더욱 간단하게 단순화 시키며 나의 뇌가 깨어나는 감각이었다. 또한 시중에 만들어져 있는 게임도 간단한 조작으로 사용하지만 이를 만들기 위해서는 복잡한 코드를 추상화시키는 사고 과정이 필요하다는 것을 깨달았다. (중략) 또한 CT는 컴퓨터 프로그래밍 뿐만 아니라 모든 삶의 문제들의 해결 과정에서 합리적이고 이상적인 해결책으로 도움이 될 것이라고 생각한다.”(성찰일지 스크립트, 예비교사1)

> “이번에 구상부터 프로그램의 구체화에 이르기까지 계획-실행-평가의 단계를 무한 반복해보며, 각 단계별 오류가 해소되어감에 따라 큰 성취감을 느낄 수 있었고 추상화 작업과 일반화 작업을 반복하며 ‘논리성’이 개발되는 것을 몸소 체험하는 계기가 되었다.”(성찰일지 스크립트, 예비교사2)



(Fig. 1) Team project report describing abstractions and problem decompositions

"문제를 작은 단위로 쪼개고, 그 문제를 해결하기 위한 순서를 생각하면서 실마리를 찾아가고 그 과정에서 문제를 하나씩 해결할 수 있어서 문제 풀이 과정 중에 성취감을 지속적으로 느낄 수 있었다. 지속적인 성취감은 문제해결에 좀 더 몰입할 수 있는 원동력이 되었다. 이런 문제해결과정을 거치다 보니, 처음에는 어렵고 힘들 것이라 생각한 큰 덩어리의 문제가 생각보다 빠르게 해결 되었다. 이번 과제를 통해 얻은 이런 문제해결방법을 다른 교과학습, 일상생활 더 나아가 나중에 학생을 지도 할 때도 쓸 수 있을 것이라는 생각을 할 수 있었다."(성찰일지 스크립트, 예비교사3)

"또한 피피티(프로젝트 템플릿)를 만드는 과정에서 스크래치 코드의 순서를 한번 더 생각하고 정리할 수 있었던 계기가 되었다. 스크래치 만들고 정리하는 과정을 통해 논리적이고 절차적인 사고력 향상에 도움이 되었다."(성찰일지 스크립트, 예비교사4)

"스크래치를 이용하여 어찌 어찌하여 만들고 나서, PPT(프로젝트 템플릿)를 만들기 위해서 다시 한 번 검토해 보는 과정을 통해서 프로그램에서 쓰인 코딩을 반성할 수 있어서 좋았다. '아, 이렇게 하면 더 좋지 않았을까?'하는 아쉬움이 드는 부분이 있기도 했다."(성찰일지 스크립트, 예비교사5)

5. 결론 및 제언

향후 초등교육현장에서 필수로 진행될 소프트웨어 교육의 실질적인 효과는 이를 담당하게 될 교사의 역량에 달려있다. 본 연구는 예비교사를 대상으로 진행되는 한 교육대학교의 소프트웨어교육 관련 교육이 예비교사의 CT 핵심 역량에 어떤 영향을 미치는지 조사해 보고자 하였다.

본 연구의 결과를 통해 향후 교사양성기관의 소프트웨어 교육 교수설계 및 운영에 대한 시사점을 제시하면 다음과 같다. 첫째, 본 연구에서 진행한 소프트웨어 교육은 CT를 기반으로 제작된 교재를 활용하여 소프트웨어의 기능에 대한 학습보다는 문제를 구성하고 문제를 분해하여 추상화하고 만들어진 알고리즘을 평가하고 효율성을 위해 대안적 알고리즘을 생성하게 하는데 초점을 두었다. 이러한 접근은 예비교사 설문 결과를 통해 확인한 결과 긍정적인 효과를 내는 것으로 나타났다. 향후 이러한 접근의 교재가 예비교사에게 맞춤화되어 더 보강될 필요가 있다고 사료된다.

둘째, CT의 핵심 역량 중 문제분해와 추상화가 예비교사에게 보다 심층적 교육 및 훈련이 필요한 영역으로 나타났다. 스크래치 교육용 프로그래밍 언어를 SW교육 도구로 활용할 경우 '추가 블록'이라는 기능이 최신 버전에 제공되고 있어서 이를 통해 문제 분해와 추상화를 구현할 수 있는 잠재력이 있다. 그러나 문제분해와 추상화를 SW교육 속에 어떻게 설계하느냐에 따라 교육 효과는 크게 달라질 수 있으므로 이에 대한 의도적 설계가 필요하다고 판단된다.

셋째, 본 연구에서는 예비교사가 스크래치로 팀프로젝트를 구현한 뒤 CT 핵심 요소가 포함된 템플릿에 내용을 정리하여 발표하도록 하였다. 연구대상의 67% 예비교사는 질적인 차이는 있으나 문제분해와 추상화의 구현 내용을 템플릿에 제시하였지만 33%는 문제분해와 추상화 영역에 대한 내용을 담아내지 못했다. 향후 SW교육 수업에서는 이에 따른 교수적 처방이 필요하다고 판단된다. 향후 이 부분에 대한 스캐폴딩 방안 연구가 더 진행되어야 할 것이다.

이상을 종합해 볼 때, 본 연구는 소프트웨어 교육을 받고 있는 예비교사가 지각한 CT경험과 프로젝트를 통해 나타난 CT세부 핵심 역량 측면의 질적 분석을 통해 향후 교원양성기관에서의 소프트웨어 교육의 교수설계 및 운영에 대해 시사점을 제공하고 있다는 점에서 의의가 있다.

참고문헌

[1] An, S., & Lee, Y. (2014). Development of pre-service teacher education program for computational thinking. In M. Searson & m. Ochoa(Eds), *Proceedings of Society for Information Technology & Teacher Education International Conference 2014* (pp. 2055-2059).

[2] Barendsen, et. al (2015). Concepts in K-9 computer science education. ITICSE-WGR conference, Vilnius, Lithuania.

[3] Bean, N., Weese, J., Feldhausen, R., & Bell, S. (2015). Starting from scratch : Developing a pre-service teacher training program in computational thinking. Frontiers in Education Conference (FIE), IEEE, El Paso, TX, 2015, pp. 1-8. doi: 10.1109/FIE.2015.7344237.

[4] Bebras international challenge on informatics and computational thinking. http://bebras.org

[5] Jeon, S., & Han, S. (2016). Descriptive assessment tool for computational thinking competencies. *Journal of The Korean Association of Information Education, 20*(3), 255-262.

[6] Jeong, I. (2015). Development of materials for programing education based on computational thinking for club activities of elementary school. *Journal of The Korean Association of Information Education, 19*(2), 243-252.

[7] Jeong, et al. (2015). Dukdak dukdak coding gongjak-so - elementary software club activities. Seoul: Seoul Books.

[8] Jeong, et al. (2015). Dukdak dukdak coding gongjak-so - elementary software club activities. Teachers manual. Seoul: Seoul Books.

[9] Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through pro-

gramming: What is next for K-12?. *Computers in Human Behavior, 41*, 51-61.

[10] National Academies of Sciences. (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington DC: National Academies Press.

[11] Papert, S. (1993). Mindstorms: Children, computers, and powerful ideas. Cambridge, MA: Perseus Publishing.

[12] Selby, C. C. (2013). Computational thinking: The developing definition. ITiCSE Conference, University of Kent, Canterbury, England, July 1-3.

[13] Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 19*(3), 33-35.

[14] Wing, J. (2011). Research Notebook: Computational Thinking - What and Why? The Link. Pittsburgh, PA: Carneige Mellon.

[15] Yonhap News (2014.7.22). Mandatory SW education starting from the first-year junior high school students next year.

저자소개

최 형 신

1988 이화여자대학교(전자계산학 학사)

1993 (미)New Jersey Institute of Technology (컴퓨터정보과학 석사)

2007 이화여자대학교(교육공학 박사)

2009-현재, 춘천교육대학교 컴퓨터교육과 교수

관심분야: 컴퓨팅 사고력(Computational Thinking), 피지컬 컴퓨팅, 뉴미디어기반 학습

e-mail : hschoi@cnue.ac.kr