JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 25 No. 4, pp. 213-222, April 2020
https://doi.org/10.9708/jksci.2020.25.04.213

Case Study on Problem-based Programming Classes in Software Education for Non-Computer Science Majors

Joo-Young Seo*, Seung-Hun Shin*

*Assistant Professor, Da-San Univertisy College, Ajou University, Suwon, Korea
*Assistant Professor, Da-San Univertisy College, Ajou University, Suwon, Korea

[Abstract]

Recently, as awareness of the need for software education has spread worldwide, the government of Korea has led compulsory software education also. Basic software education in universities has been stabilized through various trials and efforts. However, due to software classes are mandatory, students not only could not have motivation for learning but also have treated programming course as a difficult subject. In this paper, two programming classes, which were designed and managed as a problem-oriented programming class for the purpose of cultivating computational thinking for the non-computer science students, are compared using the lecture assessment results. As a result, in the case of expanding the use of the problem as a grammatical explanation aid and expanding the ratio of major-friendly problems, the student's responses were concentrated on higher scores and the response average improved by about 7%. It means that the level of difficulty experienced by learners is lowered.

▸ **Key words**: University Basic Software Education, Programming, Problem-based Education, Major-friendly Problems, Grammar Education

[요 약]

최근 소프트웨어 교육의 필요성에 대한 인식이 전 세계적으로 확산됨에 따라 우리나라도 정부 주도로 소프트웨어 의무 교육을 수행하고 있다. 대학에서의 소프트웨어 기초교육은 다양한 시행착오를 거쳐 안정화 되고는 있지만, 학생들의 비자발적 수강으로 인한 학습 동기 부족과 프로그래밍에 대한 높은 체감 난이도는 여전히 해결해야 할 문제로 남아 있다. 본 논문에서는 컴퓨팅 사고 역량 배양을 목적으로 문제 중심 프로그래밍 교과로 설계 및 운영된 컴퓨터과학 비전공 학생 대상 수업 사례를 수업 평가 결과를 이용해 비교하였다. 비교 결과, 문제의 용도를 문법 설명 보조재로 확대하고 전공 친화형 문제 비율을 확대 운영한 사례에서는 학습자들의 응답이 더 높은 점수에 집중되었고 응답 평균은 약 7% 향상되었다. 이는 학습자들이 느끼는 교과에 대한 체감 난도가 낮아졌음을 의미한다.

▸ **주제어**: 대학 소프트웨어 기초 교육, 프로그래밍, 문제 중심 학습법, 전공 친화형 문제, 문법 교육

• First Author: Joo-Young Seo, Corresponding Author: Seung-Hun Shin
*Joo-Young Seo (jyseo@ajou.ac.kr), Da-San University College, Ajou University
*Seung-Hun Shin (sihnsh@ajou.ac.kr), Da-San University College, Ajou University
• Received: 2020. 01. 20, Revised: 2020. 04. 14, Accepted: 2020. 04. 19.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

최근 지식 도메인 간 경계를 허물고 이들의 융합을 위한 기술로 소프트웨어가 지목되면서[1], 소프트웨어 교육의 중요성을 인지한 다수의 주요 국가들은 변화하는 시대를 선도하기 위한 우수 인재 양성을 목표로 컴퓨팅 사고 역량 배양을 통해 창의성을 키우는 소프트웨어 교육을 수행하고 있다[2]. 우리나라 또한 전 세계적인 흐름에 동참하여 정부 주도하에 소프트웨어 중심사회 실현을 위한 다양한 제도를 시행하고 있다. 대표 사례로 교육부는 '초중등 SW교육 활성화 방안'을 발표하고 이를 바탕으로 초중등학교에서 2018년부터 단계적으로 소프트웨어 교육을 의무적으로 이수하도록 했다[3]. 한편 과학기술정보통신부는 2015년에 시작된 소프트웨어중심대학사업을 통해 대학의 소프트웨어 교육을 주도하고 있다. 소프트웨어융합 인재 양성을 목적으로 하는 소프트웨어중심대학사업에는 2019년 현재 35개 대학이 참여하고 있으며 이들 대학에서는 모든 학생이 전공에 상관없이 의무적으로 소프트웨어 교육을 이수하고 있다[4-5].

초중등 정보 교육의 교과과정이 2015년에 개편된 이후 실제 운영까지 시간을 두고 교육과정 개발, 연구학교 운영 등의 준비 과정을 거치며 상대적으로 체계적인 도입 과정을 가졌던 반면, 대학의 소프트웨어 교육은 준비에 주어진 시간이 길지 않았던 탓에 교육에 대한 충분한 고민과 연구가 부족했고, 이에 따라 대학 소프트웨어 교육에 대한 철학이 명확히 정립되지 못한 상태에서 대학별로 수립한 목표에 따라 교육이 시작되었다. 이로 인해 소프트웨어중심대학사업 초기에는 대학 구성원 사이에 소프트웨어 교육에 대한 충분한 공감대를 형성할만한 여유가 없었을 뿐만 아니라 컴퓨터과학 비전공 학생들에 대한 이해 부족으로 효과적인 교육을 수행하기 어려웠다[1][6-7]. 하지만 사업이 진행되는 동안 다양한 시행착오를 겪으며 구성원의 소프트웨어 교육에 대한 인식은 점차 개선되었고 대학별로 학생들의 요구에 부응하기 위한 다양한 교육 프로그램을 운영하는 등의 노력을 통해 컴퓨터과학 비전공 학생의 학습 효과와 성취도에서 많은 성장을 이끌어내기도 했다[1][8-9]. 이러한 다양한 연구와 고민은 대학 소프트웨어 교육의 안정화에 상당한 진척을 이끌어냈지만 현재까지도 대학 소프트웨어 교육이 가져야 하는 철학에 대해서는 다양한 의견이 존재하여 교육 목표 및 교육 방법에 대한 다양한 형태의 연구가 이루어지고 있다[6][10-11].

대학의 컴퓨터과학 비전공자 대상 소프트웨어 교육 필요성과 인식은 크게 개선되었으나 이와 별개로 교육에 참여하는 학생들이 체감하는 어려움과 수업에 대한 만족도 개선은 쉽지 않은 것으로 알려져 있다. 실제로 소프트웨어 교육에 참여하는 학생들이 체감하는 어려움은 다수의 연구를 통해 확인되었으며[6][10][12-13], 이들은 다음과 같이 정리될 수 있다.

- 교수자의 역량 부족 및 충분하지 못한 수업 환경
- 교육 내용과 학습되는 개념 자체
- 프로그래밍 중심의 학습 도구
- 비자발적 선택에 의한 학습에서 기인하는 학습 동기 부족

이 가운데 교수 역량이나 수업 환경은 적절한 자원의 투입을 통해 해결 가능한 문제라고 할 수 있다. 하지만 다른 문제들의 해결을 위해서는 교육 수요자인 학생들과 교육 방법에 대한 이해가 필요하다.

학생들이 소프트웨어 교과에서 다루는 내용을 높은 장벽으로 느끼고 있다고 해도 소프트웨어 교육이 추구하는 기본적인 목표를 고려하면 소프트웨어 교육에서 소프트웨어 문해(Software Literacy)와 프로그래밍을 완전히 배제하기는 어렵다. 또한, 학습 동기의 결여는 소프트웨어 교과에서 학습되는 수업 내용이 향후 전공 학습과 관련이 없다는 선입견에서 기인하였다고 보는 것이 타당하다. 따라서 개별 교과의 설계자 역량이나 강의를 개설하는 학과의 요구에 따라 차이가 있겠으나 소프트웨어 교육에서 소프트웨어 문해와 프로그래밍을 배제하기보다는 이들을 디지털 사회를 살아가는 사람으로서 반드시 알아야 하는 지식으로, 또한, 학습을 위한 도구로 충실히 활용하되 교육의 효과성을 확보할 수 있는 방법의 고안을 통해 학생들이 느끼는 어려움을 완화시켜 강의 만족도를 재고하는 것이 바람직하다. 따라서 본 논문에서는 학습자 강의 만족도를 높이기 위한 한 방법으로 문제 중심 소프트웨어 교과에서 문제의 유형 변화와 활용폭 확대에 주목하고 이들의 영향을 살펴본다.

II. Preliminaries

1. Difficulties of Basic SW Education for Non-Computer Science Majors

대학 SW교육의 필요성에 대한 공감대가 확산되고 교육의 효과성이 확인되고 있다고는 하지만 소프트웨어 교육과 관련된 모든 연구 결과가 긍정적인 것은 아니다. 실제로 학생들은 소프트웨어 교육에 대한 다양한 어려움을 호소하고 있는데 대표적인 것이 학습 의지의 빈약함과 교육 내용의 어려움이다.

박금주 등은 컴퓨터과학 비전공자 대상 소프트웨어 교육에 대한 강의 평가 결과를 바탕으로 대학 소프트웨어 교

육의 문제점들을 지적했다[6]. 이 연구에서는 문제점 가운데 학생들의 소프트웨어 교육에 관한 관심 결여와 교수-학생 간 소통의 부재에서 기인한 학습 동기 부재를 가장 먼저 언급하고, 컴퓨팅 사고와 프로그래밍에 대한 이해 부족이 소프트웨어 교육의 걸림돌이 되므로 교육에 앞서 정확한 정보 전달과 동기 부여 과정의 선행을 제안하였다. 한편 이수진은 소프트웨어 교육의 시작은 컴퓨팅 사고를 갖추는 것으로 정의하고 설문 조사를 통해 학생들의 컴퓨팅 사고에 대한 이해를 고찰했다[10]. 이를 바탕으로 컴퓨터과학 비전공자들의 학습 능력은 기대보다 높으나 프로그래밍 요소를 이해하고 알고리즘을 만들어 내는 과정에 문제가 있음을 보이고 소프트웨어 교육의 학습 효과를 효율적으로 증대시키기 위해서는 프로그래밍 기술이 아닌 컴퓨팅 사고 역량을 갖는 것이 선제 되어야 한다고 했다. 또한, 오경선 등은 프로그래밍을 처음 수강하는 학생을 대상으로 한 설문을 바탕으로 컴퓨터과학 비전공 학생들이 프로그래밍을 어려워하는 이유를 분석한 연구에서 프로그래밍을 처음 학습하는 학생의 경우 프로그램을 작성하는 전체 과정에서 어려움을 느끼므로 사고에 대한 학습을 프로그래밍 과목 학습 전에 수행하여 학생들의 인지적 부담을 줄여줄 것을 제안하였다[12]. 즉, 이들 연구에 따르면 프로그래밍이 학습 도구로써 필요한 것은 사실이지만 이를 효과적으로 활용하기 위해서는 학습 동기 부여를 위한 학생들과의 소통과 사고력 배양 등이 전제되어야 한다.

다수의 대학에서는 소프트웨어 기초교육을 위한 교과를 필수로 지정하고 여기에 대체로 3~6학점을 배정하고 있다[14]. 따라서 소프트웨어 교과는 학생들의 선택으로 수강이 이루어지는 것이 아니므로 학생들의 자발적인 수업 참여를 기대하기 어렵다. 아울러 소프트웨어 교육은 자신의 전공과 무관하다는 잘못된 인식과 교수자-학생 간 소통 부재는 소프트웨어 교육에 대한 동기 부여를 더욱 어렵게 하는 요소로 작용한다. 따라서 선행 연구에서 제시한 것과 같이 소프트웨어 교과 수강 이전에 소통을 통한 학습 동기 부여나 사고력 배양 등의 과정을 배치하는 것을 고려할 수 있다. 하지만 제한된 학점 안에서 이를 반영한 교과과정을 만드는 것에는 현실적인 어려움이 있으므로 기존의 틀 안에서 교육 효과를 개선시킬 수 있는 방법을 찾는 것이 바람직하다.

특히 프로그래밍을 다루는 소프트웨어 교과에서는 수강생들이 프로그램을 작성하는 과정 전체를 모두 어려워하고[12], 소프트웨어 교육의 목표가 프로그래밍 기술 교육이 아닌 컴퓨팅 사고 역량 배양에 있으므로[10] 프로그램 작성에 필요한 개념이나 문법 중심의 교육보다는 교육 본연의 목적 달성에 초점을 두어 교과 설계가 이루어져야 한다. 또한, 수업에 참여하는 학생들의 전공과 관련된 내용을 다루는 문제를 선택하여 학생들이 컴퓨팅 사고와 프로그래밍이 자신의 역량 개발에 도움이 된다는 사실을 이해하고 나아가 수업 내용에 호기심이나 재미를 느낄 수 있도록 해야 한다.

2. Problem-based Learning in SW Education

문제중심학습(PBL)은 비구조적이고 실제적인 문제가 제시된 상황에서 자기주도적 개별학습과 효과적인 협동학습 및 전체학습을 통하여 학생들 스스로 문제를 발견하고 정의한 후, 자료와 경험의 재구성 과정을 거쳐 문제를 해결하는 과정에서 실제적인 문제해결능력을 배양할 수 있도록 하는 교육 방법이다. 또한, 제시된 문제 해결을 위한 다양한 정보 습득 과정에서 학습 동기가 유발될 수 있으며 이를 통해 학습자는 자기 주도적 학습 능력을 기를 수 있다[15]. 실제로 선행 연구에서는 PBL 기반의 프로그래밍 수업에서 실제감을 가지고 논리적인 해결 방안을 고민하는 경험을 통해 학습자의 논리적 사고력과 문제 해결력이 향상될 수 있음을 보였고[16], 교육용 프로그래밍 교육 활동에서도 PBL을 적용하는 경우, 학습자들의 논리적 사고력과 문제해결능력에 유의미한 영향을 미칠 수 있음[17]이 확인되었다.

이러한 연구 결과를 반영하듯 소프트웨어중심대학에서의 소프트웨어 기초교육을 포함한 실제 교육현장에서도 PBL을 적용한 학습 방법이 널리 사용되고 있다. 장은실 등의 분석에 따르면 소프트웨어중심대학의 기초교육에서 문제중심학습을 운영하는 대학은 2017년 현재 20개 대학 가운데 19개의 대학에 달한다. 이들 대학에서는 문제해결력, 융합적 사고력 등을 배양하기 위해 시범 실습법, PBL, 토의 토론법, 프로젝트 기반 학습, 플립드러닝 등의 교수학습법을 활용하고 있으며, 이 가운데 PBL은 모든 대학이 활용하는 시범 실습법 다음으로 많은 대학이 채택한 보편적 소프트웨어 교수법이다[18].

한편, 최형신은 프로그래밍 교육을 통한 컴퓨팅 사고력 개발에 대한 32개의 국내 연구를 분석하였는데, 조사 대상 가운데 문제해결적 접근 방법을 활용한 것이 16편으로 전체의 50%였으며, 14편(43.8%)는 대학 교육에 관련된 것이었다[19]. 하지만 대학생을 대상으로 문제해결적 접근을 한 연구는 4편(12.5%)으로 감소할 뿐만 아니라, 이 가운데 3편은 소프트웨어 교육을 전공하는 학생을 대상으로 하며, 다른 한 편은 교육 관련 학과 학생만을 대상으로 하고 있어 이를 통해 소프트웨어 기초교육에서의 PBL이 가지는 의미를 확인하기는 어렵다. 또한, 컴퓨터과학 비전공자를 대상으로 한 PBL 연구에서도 학습자의 전공과 관련된 것이 아닌 컴퓨팅 사고 교육에 일반적으로 활용되는 문제를

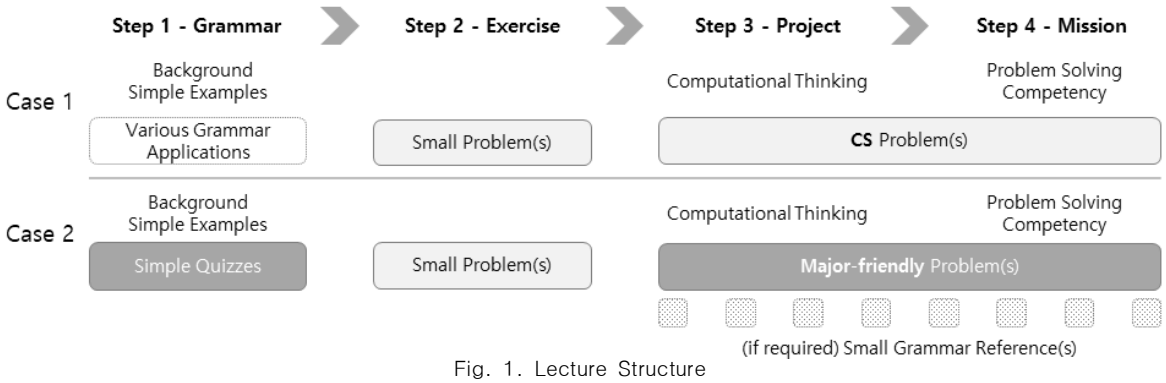

Fig. 1. Lecture Structure

활용하거나[20] 학습자의 전공을 고려할 수 없는 교양 수업에서의 사례[21] 위주로 연구가 이루어졌다. 따라서, 컴퓨팅 사고 역량 배양을 위한 프로그래밍 교육에서 학습자의 전공은 기존 연구에서 충분히 고려되지 못한 것이 사실이다. 따라서 본 논문에서는 이를 살펴보고자 한다.

III. The Case Study

1. Research Subjects

본 논문에서 살펴보는 교과는 컴퓨터과학 비전공 학과에서 1학년 1학기에 개설하는 교과로, 수강생들에게는 대학 입학 후 처음 접하는 대학 강의이자 소프트웨어 기초교육이다. 아울러 수강생 다수가 소프트웨어 교육이 의무화되기 이전의 교육과정을 거친 학생들임을 고려하면 정규 교육 과정을 통한 소프트웨어 교육 참여 경험이 전혀 없거나 적은 학생들을 대상으로 운영되는 교과이자, 기존 연구를 통해 알려진 것과 같이 학생들이 어려움을 호소하는 프로그래밍[12]을 학습하는 교과이다. 이와 같은 교과 개설 환경은 교수자와 학생 모두에게 많은 어려움을 유발할 것으로 예상되었고, 이에 따라 해당 교과는 설계 시점에 A대학의 소프트웨어 기초교육에서 정의하고 있는 핵심 역량인 컴퓨팅 사고 역량 배양을 목표로 하는 문제 중심 학습 기반의 프로그래밍 교과로 설계되었다.

사례의 교과는 소프트웨어 기초교육에 널리 활용되는 프로그래밍 언어인 Python에 대한 이해와 활용 능력 배양을 위한 교육을 수행하되 소프트웨어 기초교육의 취지에 부합되도록 문법 중심의 프로그래밍 교육 방법을 지양하고 문제 해결 과정에서 자연스럽게 컴퓨팅 사고 역량과 문제 해결 역량이 배양될 수 있도록 그림 1과 같은 구조로 설계되었다. 즉, 매 수업 시간은 다음의 구성 요소를 포함하는 프로세스에 따라 운영된다.

- Step 1. 수업 내용에 대한 이론적 배경 설명이 제공되며 핵심 문법 및 간단한 예제를 활용한 사용법이 학습된다.
- Step 2. 연습문제(Exercise) - Step 1에서 학습한 문법의 응용 사례가 필요에 따라 선택적으로 제공된다. 연습문제는 손쉽게 해결 가능한 문제를 다루되 학습된 문법을 점진적으로 확장하는 과정을 보임으로써 학생들이 이를 쉽게 이해하고 응용할 수 있도록 한다. 표 1은 연습문제 가운데 하나로 '선택(selection)'을 학습하는 과정에 사용된 것을 보인다. 주어진 문제는 두 개의 정수를 입력받아 이를 이용해 분수를 만들었을 때 생성되는 분수의 형태를 판단하는 것이다. 이 예제에서는 먼저 가분수 형태를 판단하는데 필요한 문법을 보인 후, 예외 상황 처리, 가분수 혹은 진분수 판단 등으로 기능을 점진적으로 확장해가며 자연스럽게 문법의 활용 방법과 프로그램 작성 시 고려해야 하는 내용 등을 제시하고 있다.
- Step 3. 프로젝트(Project) - 적절한 규모의 응용문제를 제시하고 해당 문제를 해결하는 과정을 요구 분석, 데이터 설계, 입출력 설계, 알고리즘 설계 단계로 나누어 단계별로 고려해야 하는 것과 만들어야 하는 것들의 의미와 이에 필요한 사고 과정을 상세 설명한다. 또한, 한 학기 동안 다루는 다수의 서로 다른 주제의 문제를 동일한 절차에 따라 해결하는 것을 보임으로써 문제를 바라보는 시각과 문제 해결에 필요한 사고 절차가 자연스럽게 반복 학습되도록 한다.
- Step 4. 미션(Mission) - 프로젝트와 난이도가 유사하나 추가 응용이 요구되는 문제를 제시하되 수업 시간 내에 해결 가능한 수준의 문제를 선별하여 활용한다. 문법, 연습문제 및 프로젝트에서 학습된 내용을 종합하여 주어진 문제를 학생 스스로의 역량으로 해결하도록 한다.

사례의 교과는 문제를 활용하는 용도와 문제의 유형에 차이를 두어 2개 학기 동안 운영되었다. 두 학기 모두 앞

Table 1. Exercise Sample

Subject - Selection

Question - Write a program that accepts two integers, each of which will be used as a numerator and denominator, and outputs whether this fraction is an improper or proper fraction.
 	
Step 1 - Improper fraction case
...
if numerator > denominator :
  ...
 	
Step 2 - Handling Exception(Div. by zero)
...
if denominator == 0 :
  ...
else :
  if numerator > denominator :
  ...
 	
Step 3 - Proper or Improper case
...
if denominator == 0 :
  ...
else :
  if numerator > denominator :
  ...
  else :
  ...

Table 2. Example of Grammar Description Style

Case 1
in the lecture note of week2 Function : print() Usage : print(string) or print(var) 1. Simple print() examples 2. String with format() function 3. Output formatting
Case 2
in the lecture note of week2 Function : print() Usage : print(string) or print(var) - Simple print() examples **in the lecture note of week3** - Simple reference of format() function **in the lecture note of week6** - Simple reference of output formatting

서 기술한 기본 구조를 따르고 있으나, 첫 번째 학기(사례 1)에는 수강생들의 전공에 대한 고려 없이 기존 컴퓨터과학 도메인에서 널리 활용되는 문제를 중심으로 교과를 구성하여 운영하였다. 한편, 두 번째 학기(사례 2)에는 사례 1과 교과의 기본적인 구조는 공유하되 수강생의 전공과 관련된 내용을 다루는 문제 비율을 사례 1에 비해 확대하고, 문법 학습 과정(Step 1)에도 문제를 활용한 교육을 수행하였다. 본 논문에서는 이 두 사례의 운영 결과를 수강생들의 수업 평가 결과를 이용해 비교하고, 이를 바탕으로 문제 중심 소프트웨어 기초교과에서 문제의 유형 변화와 활용폭 확대가 학습자의 강의 만족도에 미치는 영향을 살펴본다.

1.1 Grammar Description Style

컴퓨터과학 비전공자를 대상으로 한 기존 연구에서 학생들은 컴퓨팅 사고 역량 배양을 위한 학습에 활용되는 도구의 하나인 '프로그래밍 언어'에 대한 어려움을 표현하고 있다. 하지만 소프트웨어 기초교육의 기초적인 목표가 컴퓨팅 사고 역량을 바탕으로 한 문제 해결 능력 배양이자 학생들이 학습된 내용을 자신의 전공에 활용할 수 있도록 응용력을 기르는 것이라면 필요 수준 이상의 문법 학습은 학생들의 어려움만 가중할 뿐 교육적 의미가 크지 않다고 할 수 있다. 한편, 컴퓨팅 사고 역량이 프로그래밍 역량과 상관관계를 가지므로 컴퓨팅 사고 역량 배양에 집중하면 프로그래밍 역량 향상 또한 기대할 수 있을 것이다.

A대학에서는 소프트웨어 기초교육 수행 결과를 지속적으로 분석하고 이를 바탕으로 소프트웨어 기초 교과의 효과성을 확대하기 위한 다양한 노력을 수행하고 있다. 사례 분석 대상 교과는 최초 설계 시 컴퓨팅 사고 역량 배양을 위한 문제 중심 교과로 설계되어 프로그래밍 언어의 문법 설명은 상세 설명 대신 기본 형식과 이를 활용한 사례로 위주로 이루어졌다. 따라서 사례 1의 경우에는 표 2에 제시된 것과 같이 도입부에 해당 학기에 필요로 하는 범위로 제한된 문법 설명이 제공되었다. 하지만 사례 2 운영 시에는 문법의 설명을 더욱 축소하여 도입부에서 기본적으로 설명되었던 문법 분량을 기본 형식과 간단한 사례만으로 구성하고 여기에서 설명이 이루어지지 않은 내용은 추후 필요한 시기에 수업에 필요한 양만큼만 간략한 참조 형태로 제공하였다. 또한, 사례 2에서는 줄어든 문법 설명 부분을 문법에 대한 이해를 돕기 위한 목적의 문제들로 대신했다. 우선, 학습한 문법 중 대표적인 것을 이해하고 있는지 학생들 스스로 확인할 수 있도록 하기 위한 퀴즈 문제를 문법 설명 직후에 제공하였다. 퀴즈는 표 3에 제시된 것과 같이 짧은 시간 내에 해결할 수 있는 문제들로 구성되어 있으며 시험 형식이 아닌 교수자와 학생들이 함께 풀어보는 형식으로 운영하여 전반적인 학생들의 이해를 도모했다. 또한, 다양한 형태의 응용을 확인할 수 있도록 하기 위한 연습문제의 수를 확대해 운영하였다.

Table 3. Quiz Samples

Subject - Assignment Operator
When you want to exchange the values stored in two variables, write the statement(s) you need. E.g, When a = 1 and b = 2, how do we swap the values stored in a and b so that b = 2 and a = 1?
Subject - Selection
1. Write a program that takes two integers and outputs whether two values are equal or not. 2. Write a program that takes two different integers and outputs the larger one.

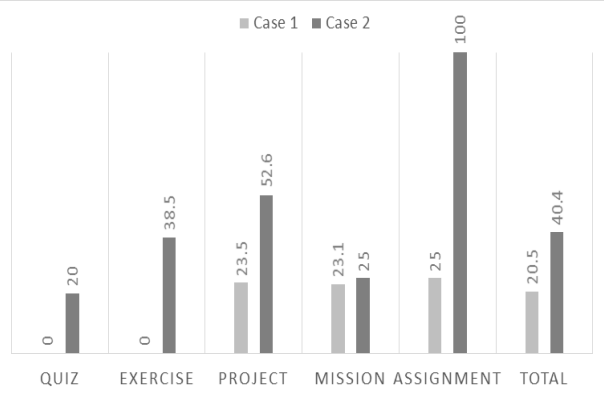

Fig. 2. Major-friendly Problem Ratio

Table 4. Questions for the Lecture

No	Question
Q4	The lecture was conducted with academic interest and curiosity
Q7	Class contents were organized and developed systematically.
Q8	Teaching methods(lecture, discussion, demonstration, etc.) were appropriate for this class.
Q10	The assignment appropriately supplemented the contents and helped to enhance the learning effect.
Q12	Lecture materials were appropriate and helpful for learning.
Q14	The difficulty and pace of the lecture were adequate.

1.2 Major-friendly Problems

사례 1과 사례 2는 수업에 활용한 문제의 유형에도 차이가 있다. 사례 1에서 활용된 문제의 유형은 기존 A 대학의 컴퓨터과학 교육에 기반을 두고 있으나, 사례 2에서는 학생들의 수업 참여 의지 재고를 위해 전공과 관련된 주제를 다루는 전공 친화형 문제 비율을 증가시켰다. 그림 2는 수업 시간에 사용된 전체 문제 수 대비 전공 친화형 문제 수의 비율을 구성 요소별로 제시하고 있다. 이 그림에서 사례 1의 문제 유형을 살펴보면 전체 문제 가운데 약 20%만이 수강생의 전공과 관련성이 있는 것을 알 수 있다. 이는 사례 1의 경우 과목 개발 시 전공에 대한 고려가 이루어지지 않았기 때문이며 이 비율은 순수하게 임의성에서 기인한 것이다. 문제 수를 정량적으로 살펴보면 사례 2에서 다룬 전체 문제 가운데 전공 친화형 문제의 비율은 약 40%로 사례 1에 비해 약 2배 증가되었다. 세부적 살펴보면 구성하는 모든 항목에서 전공 친화형 문제 비율이 증가되었고 특히 과제에서는 모두 전공 친화형 문제를 다루었으며, 프로젝트의 경우 사례 2가 사례 1의 2배 이상이었다. 또한, 퀴즈와 연습문제는 문법 학습을 위한 보조 도구이므로 이를 배제하고 프로젝트와 미션만을 고려하면 전공 친화형 문제 비율은 사례 1이 23.5%, 사례 2는 47.1%로 2배 이상 증가되었다. 한편 퀴즈와 연습문제에도 전공 친화형 문제를 각각 20%와 38.5% 활용하였다.

1.3 Questionnaire for the Lecture Assessment

A 대학의 수업 평가를 위한 설문 문항 중 강의 만족도 분석에 사용한 문항 구성은 표 4와 같다. 설문지 문항은 교수자의 태도, 수업 내용, 평가 및 수업 방식 그룹으로 구분되고 총 14개의 세부 문항들이 배분되어 있다. 수업 평가는 종강 후에 시행되고 모든 수강생은 의무적으로 설문에 참여한다. 모든 문항은 5점 리커트 척도를 사용하며, 1부터 5까지 각각 '매우 불만족', '불만족', '보통', '만족', '매우 만족'을 의미한다. 이에 추가로 학생들의 주관적 의견을 표현할 수 있도록 개방형 문항 1개가 포함되어 있다. 본 논문에서는 이 문항들 가운데 학생들의 체감 수업 난이도와 교과 구성 만족도를 확인하기 위한 문항 Q4, Q7, Q8, Q10, Q12, Q14에 대한 응답을 비교한다. 개방형 문항에 대한 응답 또한 수업 관련 의견만 수집하여 비교한다.

한편, 강의 만족도는 교수자의 준비와 태도에 의해서도 영향을 받을 수 있다. 따라서 강의 만족도 분석 결과에서 교수자에 의한 영향이 배제되어야 하므로 교수에 대한 평가 문항을 별도로 분석하여 강의 만족도와 비교한다. 교수 만족도 분석을 위해서는 동일 설문지의 Q2, Q5, Q6에 대한 응답을 비교한다. 교수 만족도 분석에 사용한 문항 구성은 표 5와 같다.

Table 5. Questions for the Professor

No	Question
Q2	Lecture preparation of the professor was faithful
Q5	The students were respected and treated them respectfully
Q6	Student's questions were accepted and answered appropriately

1.4 Method of Analysis

먼저 사례 1과 사례 2의 강의 만족도를 문항별로 분석하여 비교하고, 교수자 만족도 분석 결과를 확인한다. 한편, 사례 1의 학습자 수는 42명인데 반해, 사례 2의 학습자 수는 22명으로 응답 분포 확인 결과 이는 정규분포를 따르지 않았다. 따라서 두 사례의 비교를 위해 윌콕슨 부호 순위 검정(Wilcoxon signed-rank test)을 수행하였으며, 분포 확인과 검정은 R에서 통계 패키지 'stats'를 활용해 진행하였다. 또한, 검정 결과를 응답 분포와 비교하여 분석 결과가 갖는 의미를 명확히 확인하기 위해 응답의 첨도(kurtosis)를 'fBasics' 패키지를 활용해 파악하였다.

2. Lecture Assessment Results

표 6과 그림 3은 각각 강의에 대한 응답을 분석한 결과와 응답의 분포를 보인다. 사례 1은 문항별로 평균 최저 4.05에서 최고 4.225를, 사례 2는 4.36에서 4.59의 분포를 보이고, 모든 문항에서 사례 2의 점수가 사례 1에 비해 높았으며, 두 사례 간의 차이는 최소 0.19(Q10, 과제의 적절성), 최대 0.50(Q14, 난이도와 속도의 적정성)을 보였다. 하지만 윌콕슨 검정 결과에서는 모든 문항에서 두 사례의 차이가 확인되지 않았다(p>.05). 그러나, 응답 분포를 살펴보면 사례 1은 대다수의 선택이 3~5점(Q4, Q7) 혹은 1~5점(Q8, Q10, Q12, Q14)에서 이루어졌지만, 사례 2의 경우 3점 이상이 선택되었다. 이러한 결과는 각 문항에 대한 응답의 첨도를 통해서도 확인이 되는데, 모든 문항에서 사례 2의 첨도가 사례 1보다 큰 값을 가지는 것으로 확인되었다. 이는 비록 두 사례의 평균에서는 유의한 차이가 확인되지 않았지만, 사례 2 학습자들의 응답이 사례 1에 비해 평균과 첨도가 높고, 다수의 응답이 상대적으로 더 좁은 영역에 분포되어 있으므로 사례 2의 응답은 상대적으로 높은 점수에 집중되어 있음을 의미한다. 한편, 표7과 그림 4는 교수자에 대한 평가 문항의 분석 결과를 보인다. 교수자에 대한 평가는 강의에 대한 평가와 유사하게 모든 문항에서 사례 2의 평균 점수가 사례 1보다 큰 값이 나왔지만 윌콕슨 검정 결과에서는 유의한 차이가 확인되지 않았다(p>.05). 또한, 두 사례의 응답 분포가 서로 유사한 형태이므로 두 사례의 강의 만족도 비교에서 교수자의 영향은 배제 가능함을 알 수 있다.

사례 2 학습자의 응답이 높은 점수로 수렴되었음이 확인됨에 따라 평균 점수를 기준으로 두 사례 사이에서 가장 큰 편차를 보이는 문항을 살펴보면, Q14와 Q7로 각각 '강의의 난이도와 속도'와 '체계적인 수업 내용의 구성 및 전개'를 묻는 문항이다. 사례 1과 사례 2는 다루는 문제의 규모와 유형을 제외하면 학습 내용에는 거의 차이가 없다. 따라서, 이는 문제 중심 학습이 강화되고 전공 친화형 문

Table 6. Lecture Assessment Result of the Lecture

No	Computer Science Problem(n=40)			Major-friendly Problem (n=22)			W of Wilcoxon test	p-value
	Mean	SD	Kurtosis	Mean	SD	Kurtosis		
Q4	4.225	0.851102	-1.55026	4.454545	1.075651	2.715279	348	.1299
Q7	4.2	1.029563	0.654272	4.590909	0.937069	6.320325	330	.0645
Q8	4.175	1.092875	0.11814	4.409091	1.154402	1.768379	368	.2286
Q10	4.175	1.069755	0.11814	4.363636	1.149919	1.570591	382.5	.3422
Q12	4.175	0.997184	0.180375	4.454545	1.075651	2.715279	353.5	.151
Q14	4.05	1.223724	0.222782	4.545455	0.78203	2.400386	348.5	.133

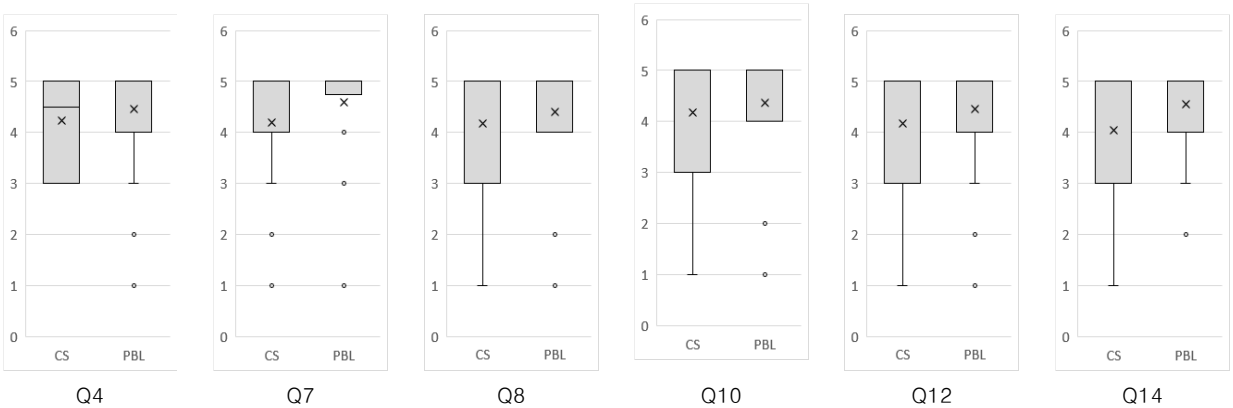

Fig. 3. Response Distribution of the Lecture Assessment

Table 7. Lecture Assessment Result of the Professor

No	Computer Science Problem (n=40)			Major-friendly Problem (n=22)			W of Wilcoxon test	p-value
	Mean	SD	Kurtosis	Mean	SD	Kurtosis		
Q2	4.325	0.95884	1.272186	4.545455	0.655555	-0.18835	403	.537
Q5	4.325	0.95884	1.272186	4.545455	0.940371	5.725827	376.5	.2757
Q6	4.25	0.99373	0.930601	4.5	0.941469	5.243545	371.5	.2555

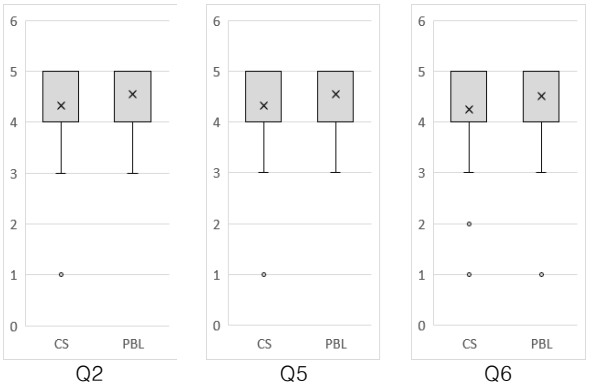

Fig. 4. Response Distribution of the Professor Assessment

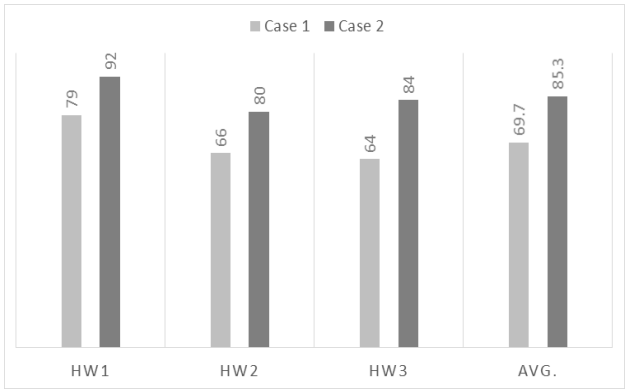

Fig. 5. Assignment Submission Rate (%)

제의 비율이증가됨에 따라 학습자들이 교육 내용에 상대적으로 잘 적응하였음을 의미한다. 이러한 결과를 기존 프로그래밍 언어를 학습하는 소프트웨어 기초교육에 대한 연구 결과에 비추어보면 문법 위주의 교육보다는 문제 중심의 학습을 학생들이 더 편하게 받아들이고 있으며, 이는 학생들의 수업 내용에 대한 이해 수준 향상으로 확장되어 전체적인 수업의 구성을 인지하는 데까지 영향을 미친다는 것을 알 수 있다.

한편 가장 작은 편차를 보인 문항은 Q10으로 전공 친화형 문제로 모두 전환되었던 과제와 관련된 것이다. 과제의 경우 두 사례에서 출제된 과제의 내용과 난이도가 서로 달라 직접적인 비교가 불가능하나 사례 2에서는 문제 해결 역량 배양에 초점을 두어 사례 1의 과제보다 문제의 길이가 상대적으로 길고 요구 사항이 더 복잡한 문제를 다루었다. 하지만 그림 5에 제시된 과제 제출 학생의 비율을 살펴보면 사례 2에서는 사례 1보다 제출 비율이 약 16% 향상되었으며, 특히 난도가 높아지는 후반 과제 제출 비율이 사례 1에 비해 높은 수준으로 유지되는 결과를 보였다. 이를 종합해 보면 전공 친화형 문제와 문제 중심 학습 방법이 상대적으로 높은 학습 효과성을 유도할 수 있으며 나아가 학생들의 중도 이탈 확률을 낮추는 데에도 기여함을 알 수 있다.

표 8은 설문에 참여한 학생들이 강의 평가 시 제출한 개방형 질문에 대한 응답 가운데 교수자에 대한 의견을 제외하고 수업에 관련된 것들만 정리한 것이다. 부정적인 답변을 살펴보면 사례 1의 경우 강의가 너무 '어려움'이나 '재수강' 등의 표현을 통해 수업에 적응하지 못했다는 응답을 확인할 수 있다. 한편 사례 2는 '어려움'이나 '재수강' 등의 키워드는 사라지고 '강의 노트에 제공된 콘텐츠의 부족'을 언급하고 있다. 이들을 정리하면 사례 2의 경우 사례 1에 비해 학생들이 체감하는 난도는 낮아졌지만, 강의 노트가 직접적인 설명 중심이 아닌 문제와 문제의 해결 방법 위주로 구성되어 있어 자습용으로 활용하기에는 부족하다고 판단한 것으로 보인다. 하지만 강의 노트에 대한 수업 평가 문항 Q12가 비교 문항 중 세 번째로 높은 상승폭을 보인 것을 고려하면 자습을 위한 용도로서의 강의 노트의 부족함은 수업의 전체적인 만족도에는 큰 영향을 미치지는 않았을 것으로 추정된다.

Table 8. Student Comments for the Lecture

Case	Comments
Case 1	• I was able to get closer to information easily • It was too difficult • It was useful lecture as a basic programming class • I will take the class again • It was suitable lecture for the freshmen
Case 2	• It was appropriate class for the novices • Programming was interesting even though it was my first programming experience • I became interested in programming • The contents of the lecture notes were not sufficient

IV. Conclusions

본 논문에서는 문제 중심으로 설계된 컴퓨터과학 비전공자 대상 프로그래밍 교과의 두 사례를 수업 평가 결과를 이용해 비교하였다. 사례 1은 학습자의 이해를 돕기 위한 목적의 '배경 지식 및 문법 설명'이 제공된 후, 응용 사례 제시를 위한 '연습문제', 컴퓨팅 사고 역량 배양과 프로그래밍 절차 교육을 위한 '프로젝트' 및 학습된 역량을 바탕으로 학생 스스로 주어진 문제를 해결하는 '미션'으로 구성해 운영되었으며, 이 과정에서 활용된 문제의 약 80%는 A 대학에서 기존 컴퓨터과학 프로그래밍 교과에 활용되었던 주제를 다루었다. 한편 사례 2에서는 사례 1과 기본적인 학습 절차는 공유하되 문제의 활용 폭을 확장하고 전공 친화형 문제의 비율을 높여 운영하였다. 세부적으로 살펴보면 '퀴즈'를 문법 설명을 보조하기 위한 용도로 도입하고, '연습문제'의 수를 확대하였으며, '프로젝트'와 '미션'에 사용된 전공 친화형 문제는 첫 번째 사례의 약 2배 수준으로 구성하였다.

수업 평가 문항 중 교과 구성과 난이도에 관련된 6개 문항에 대한 응답만 선별하여 비교한 결과 학습자들은 선별된 모든 문항에서 사례 2가 더 긍정적인 것으로 응답했다. 이 가운데 '강의의 난이도와 속도'와 '체계적인 수업 내용의 구성 및 전개'의 향상도가 가장 컸으며, '과제의 수업 내용 보완 및 학습 효과'의 향상도가 가장 작았다. 두 사례의 응답 평균은 유의한 차이를 보이지 않았으나 사례 2의 경우 사례 1에 비해 응답이 높은 점수로 집중되었고 이는 학습자들이 사례 2를 더 적절하게 판단하고 있음을 의미한다. 한편, 응답 평균의 정량적인 차이를 살펴보면 사례 2가 사례 1보다 약 7% 향상된 결과를 보였다. 개방형 질문에 대한 응답은 사례 1의 경우 학습과 수업 내용의 '어려움'이 표현되었으나, 사례 2에서는 강의 노트를 활용한 학습의 '불편함'이 언급되었다.

이와 같은 비교 결과를 정리하면 문제를 문법 설명을 위한 도구로 활용하고 전공 친화형 문제를 사용해 학습자의 수업 참여 의지를 확보하는 경우 학생들의 수업에 대한 체감 난도를 낮게 하고 수업에서 다루는 내용에 대한 이해를 높여 전반적인 강의 만족도 향상에 도움이 된다는 것을 알 수 있다. 따라서 그간 많은 어려움이 표출되었던 컴퓨터과학 비전공 학생 대상 프로그래밍 교과의 어려움을 완화시키는 방법의 하나로 전공 친화형 문제 중심 교과 운영을 고려할 수 있을 것이다.

수업 평가 결과를 비교한 본 사례 연구에서는 제한된 수업 평가 문항을 활용하여 학습자들의 강의에 대한 전반적인 인식의 변화를 확인하였다. 하지만 수집된 설문 결과에서 개별 학습자 식별이 불가능하여 세부 구성 요소 간의 상관성 및 연관성 등 세부 분석이 수행될 수 없었다는 데 한계가 있다. 또한, 컴퓨팅 사고 역량 배양을 목표로 운영되는 교과이지만 강의의 효과성에 대한 면밀한 분석은 이루어지지 못했다. 이에 따라 소프트웨어 기초교육을 수강하는 모든 학생을 대상으로 컴퓨팅 사고 역량, 소프트웨어 문해력 등을 종합적으로 살펴보기 위한 소프트웨어 교육 효과성 분석 연구가 진행 중이다.

REFERENCES

[1] Jeong-Eun Nah, "Software Education Needs Analysis in Liberal Arts," Korean Journal of General Education, Vol. 11, No. 3, pp. 63-89, June 2017.

[2] Ho-Sung Woo, Ja-Mee Kim, Won-Gyu Lee, "A Comparative Analysis of domestic universities curriculum based on overseas higher Informatics standard curriculum," The Journal of Korean Association of Computer Education, Vol. 20, No. 1, pp. 27-38, January 2017.

[3] Ministry of Education, Ministry of Science, ICT and Future Planning, Human Resource Development Plan for SW-centric Society. https://www.msit.go.kr/SYNAP/skin/doc.html?fn=075964f3c916f54dc5b36afbd4bfd256&rs=/SYNAP/sn3hcv/result/201908/

[4] IITP, Korea. Human Resource Development through the National Program for Excellence in SW. https://www.iitp.kr/kr/1/notice/reportAndClarify/view.it?ArticleIdx=3561&count=true

[5] Ministry of Science, ICT and Future Planning. Plan for the National Program for Excellence in SW. https://www. msit.go.kr/web/msipContents/contentsView.do?cateId=mssw11211&artId=1272942

[6] Geum-Ju Park and Young-Joon Choi, "Exploratory study on the direction of software education for the non-major undergraduate students," Journal of Education & Culture, Vol. 24, No. 4, pp. 273-292, August 2018. DOI: 10.24159/joec.2 018.24.4.273

[7] Woo-Yong Kim. University Software Education. Is it good? https://www.zdnet.co.kr/view/?no=20180209110815

[8] Min-Seok Lee. Status and Crisis of University SW Education. https://spri.kr/posts/view/22496

[9] Min-Ja Kim and Hyeon-Cheol Kim, "Effectiveness analysis based on computational thinking of a computing course for non-computer majors," The Journal of Korean Association of Computer Education, Vol. 21, No. 1, pp. 11-21, January 2018.

[10] Su-Jin Lee, "A Study on Designing a Class of Convergence Thinking based on Computational Thinking," The Korean Society of Science & Art, Vol. 36, pp. 255-263, December 2018. DOI: 10.17548/ksaf.2018.12.30.255

[11] Wan-Seop Kim, "A Study on the Students Perceptions Trend for Software Essentials Subject in University," Korean Journal of General Education, Vol. 13, No. 4, pp. 161-180, August 2019.
[12] Kyung-Sun Oh and Seong-Jin Ahn, "A study on the relationship between difficulty in learning to program and Computational Thinking," The Journal of Korean Association of Computer Education, Vol. 18, No. 5, pp. 55-62, September 2015.
[13] Soo-Hwan 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, May 2015.
[14] Joo-Young 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, Jul. 2017.
[15] Young-Shin Han, "Effectiveness of problem-based learning based programming education : Focus on Computational Thinking," Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology, Vol.8, No.7, pp. 433-445, Jul. 2018. DOI:10.21742/AJMAHS.2018.07.74
[16] Hak-Jin Bae, "The Elementary school Programming teaching model by Problem-based learning," Masters's Thesis, Korean National University of Education, 2009.
[17] Shin-Jong Paik, "The Effects of Educational Programming Language with PBL(Problem Based Learning) on logical thinking ability and Problem Solving ability in elementary school environments," Masters' Thesis, Korea National University of Education, 2017
[18] Eun-Sill Jang, Jae-Hyoun Kim, "Contents Analysis of Basic Software Education of Non-majors Students for Problem Solving Ability Improvement – Focus on SW-oriented University in Korea," Journal of Internet Computing and Services, Vol. 20, Issue 4, pp. 81-90. Aug. 2019.
[19] Hyung-Shin Choi, "Domestic Literature Review on Computational Thinking Development through Software Programming Education," Journal of Educational Technology, Vol 34, No 3, pp.743-774, Sep. 2018. DOI:10.17232/KSET.34.3.743
[20] Kwangil Ko, "A Study on the Effectiveness of EPL Utilizing Programming Education based on Problem Based Learning (PBL) for Non-SW Major," Jouranl of Information and Security, Vol. 19, No. 2, pp. 105-111, Jun. 2019. DOI:10.33778/kcsa.2019.19.2.105
[21] Eui-Sun Kang, Sun-Im Shin, and Kwang-Jin Lee, "Education Model Using PBL for IT Convergence Education of Non-Major in Liberal Arts Class: Focusing on Computing Thinking," Journal of Digital Contents Society, Vol. 20, No. 11, pp. 2159-2168, Nov. 2019. DOI:10.9728/dcs.2019.20.11.2159

Authors

Joo-Young Seo received the B.S., M.S. and Ph.D. degrees in Computer Science and Engineering from Ewha Womans University, Korea, in 1993, 2001 and 2009, respectively. Dr. Seo joined the faculty of the Department of Information and Computer Engineering at Ajou University, Suwon, Korea, in 2009. She is currently a Professor in the Dasan College University, Ajou University. She is interested in software education and software engineering with particular emphasis on software testing, embedded software testing and test automation.

Seung-Hun Shin received a B.S. degree in Information & Computer Engineering from Ajou University, Suwon, Korea, in 2000, and M.S. and Ph.D. degrees in Information & Communication Engineering from Ajou University, Suwon, Korea, in 2002 and 2011, respectively. From September 2011 to February 2016, he was with the department of Software Convergence Technology, Ajou University as a Lecture Professor. Since March 2016, he has been with the Da-san University College, Ajou University as an assistant professor. His research interests include software testing algorithm, network intrusion detection, and mobile multimedia networking.