Journal of The Korean Association
of Information Education Vol. 20, No. 6,
December 2016, pp. 585-596

http://dx.doi.org/10.14352/jkaie.2016.20.6.585

Computational Thinking에서의 추상화 개념에 대한 고찰

정인기

춘천교육대학교 컴퓨터교육과

요 약

2018학년도부터 초중고등학교에서 소프트웨어 교육이 시행될 예정이다. 소프트웨어 교육의 목표는 학생들에게 Computational Thinking 능력을 길러주는 데 있다고 할 수 있다. Computational Thinking은 크게 추상화와 자동화로 구성되는데 추상화와 관련된 개념을 국가마다 전공마다 다르게 정의하고 있어 학생들과 교사들한테 혼란을 주고 있다. 따라서 본 논문에서는 추상화에 대한 여러 정의들을 비교하여 합리적으로 정의하였다. 합리적인 정의를 바탕으로 초등학교에서에서의 추상화 교육 방법과 평가 기준에 대하여 제안하였다. 본 논문에서 제시한 추상화에 대한 정의는 앞으로 소프트웨어 교육을 진행하면서 만나게 되는 문제에 대한 해결 방법을 제시할 수 있을 것으로 기대된다.

키워드 : 소프트웨어 교육, 컴퓨팅 사고, 추상화, 분해, 일반화, 평가 기준

Review of Concept of Abstraction of Computational Thinking

InKee Jeong

Dept. of Computer Education, ChunCheon National University of Education

ABSTRACT

Software Education will be implemented at elementary, middle and high schools starting in 2018. The goal of software education is to help students develop Computational thinking skills. Computational thinking is largely composed of abstraction and automation. However, the concepts related to abstraction are defined differently for each country, giving confusion to students and teachers. Therefore, in this paper several definitions of abstraction are compared and defined reasonably. And we proposed an abstraction teaching method and evaluation criteria in elementary school based on a reasonable definition. The definition of abstraction presented in this paper is expected to be able to present a solution to the problems encountered in the course of software education in the future.

Keywords : SW Education, Computational Thinking, Abstraction, Decomposition, Generalization, Evaluation Criteria

논문투고 : 2016-12-01
논문심사 : 2016-12-01
심사완료 : 2016-12-09

1. 서론

국가의 미래를 발전시킬 핵심 자원은 '정보'와 '사람'이라는 인식이 확산되어 이에 대한 국가 차원의 발전 전략을 세우고 있다. 주요 선진국들은 미래 사회를 이끌어가는 핵심 인재 육성에 힘을 기울이고 있는데 미래 인재 육성 정책은 바로 미래 인재가 가져야 할 핵심 역량을 찾아내어 교육 정책에 적용하는 것이 정책의 핵심이다. 즉, 국가의 교육 경쟁력이 국가 경쟁력의 원천이 된다는 점에서 미래 사회에 요구되는 핵심 역량을 학교 교육을 통해 실현하는 것은 국가 정책의 최우선이 되고 있는 실정이다[18].

미래 직업은 더욱 복잡한 문제를 효율적이고 창조적으로 해결하는 능력을 요구한다. 이를 대비하기 위해 우리 사회 전반에서 필요하다[12].

미래부는 '과학기술과 ICT를 통한 창조경제와 국민행복 실현'이라는 비전을 제시하고 추진 전략으로 21세기 언어인 소프트웨어를 핵심 산업으로 육성하고 누구나 소프트웨어를 개발 및 활용할 수 있는 교육 기반을 조성하며 '한국 스타일' 콘텐츠로 세계에 진출하는 과제를 제시하고 있다[10][12].

SW 교육의 목적은 프로그래머를 키우는 것이 아니다. 수학 공부를 하는 이유가 수학자를 만드는 이유가 아니고, 국어 공부를 하는 이유가 작가를 만들기 위함이 아니듯, SW 교육의 목적은 프로그래머를 키우기 위함이 아니다. SW 교육의 목적은 컴퓨터 과학의 원리와 개념을 이해하고, 그것을 바탕으로 일상생활에서 문제를 해결하는 능력, 즉 컴퓨터 과학적 사고력을 키우기 위함이다[28].

소프트웨어 교육의 방향은 그 거시적 목표 설정에 좌우되며 그 결과 또한 목표의 성취로 얻어지게 될 것이다. 이런 맥락에서 소프트웨어 교육의 주요 목표로 제안되고 있는 것이 Computational Thinking 역량 계발이다[13][19].

이전 시대의 SW 교육은 SW 개발자나 엔지니어를 양성하는 것이 목적이었다면 현대 시대의 SW 교육의 목적은 일반적이고 보편적인 교육으로서의 접근이 이루어지고 있다[20][24]. 보편 교육으로서의 SW 교육의 접근 교육 목표의 설정이 다르다. 보편 교육의 의미는 모든 학생들에게 SW 교육을 가르쳐야 하며 내용과 방법이 보편 교육에서 추구하는 기본 사고력, Computational Thinking의 향상에 초점이 맞추어져야 함을 의미한다[16][24].

Papert[20][13]에 의해 처음 사용된 Computational Thinking은 2006년 미국 카네기 멜론 대학교의 컴퓨터 과학과 교수 Jennette Wing의 논문을 통해 학계에 크게 부각되었다. Wing은 Computational Thinking이 단순히 컴퓨터 과학자들만이 아니라 모든 사람들이 배우고 사용해야 하는 보편적인 태도와 기술을 제시하고 있다고 주장하였다. 따라서, 읽기, 쓰기, 셈하기에 이어 모든 아동의 분석적 능력으로 Computational Thinking을 추가해야 한다고 하였다[12][23][25].

정보 과학적 관점에서 Computational Thinking은 대표적인 문제 해결 방법으로 볼 수 있는데 핵심적인 의미는 인간이 실생활에서 직면할 수 있는 다양하고 복잡한 문제를 어떻게 해결할 것인지를 절차적으로 사고하고, 문제의 해결 과정을 컴퓨팅 기기가 제공하는 강력한 능력들을 통해 효과적이고 효율적으로 해결하고자 하는 종합적인 사고 과정이라고 할 수 있다. 기존에는 컴퓨팅 기기를 단순히 반복적이고 지루한 작업을 대신하는 일반적인 기기로 인식했다면 Computational Thinking의 개념에서는 컴퓨팅 시스템이 가지고 있는 다양한 능력을 인간의 사고 능력과 통합하여 새로운 형태의 인지 구조를 활용하는 것이라 할 수 있다. 즉, 복잡하고 다양한 문제를 해결하기 위해 필요한 자료를 수집・추출・분석하여 문제의 해결 모델을 구축하고, 이러한 모델을 컴퓨터가 이해할 수 있는 언어로 구현하여 문제의 해결 방법을 발견하는 것이다[27].

Computational Thinking의 구성 요소로는 크게 추상화(Abstraction)와 자동화(Automation)를 들 수 있다. 추상화는 '무엇을 단순화하고 관심을 집중시키기 위해 세부적인 것을 제거하는 과정'이며, '그것들의 공통적인 핵심과 본질을 찾기 위해 이를 일반화하는 과정'이다. 추상화는 실제 세계의 문제를 해결 가능한 형태로 표현하기 위한 사고 과정이라고 할 수 있다. 문제를 해결하기 위하여 필요한 자료를 수집 및 분석하고, 필요한 방법(도표, 그래프 등)을 활용하여 눈으로 보기 쉽게 나타내고, 복잡한 요소를 작은 단위로 분해하고, 해결에 필요한 변수들을 추출하여 적절한 해결 모델을 설계하는 과정이다[9][11][17][26].

Wing은 추상화를 정신적 도구라고 한다면 이러한 추상화의 기능은 기계적 도구인 자동화를 통해 더 증폭되어진다고 말한다. 자동화는 추상 개념과 추상 레이어, 이들 사이의 관계를 기계화하여 작동시킨다. 즉 이해하기 복잡하고 까다로운 추상 개념의 추상 레이어를 정교하게 해석할 수 있도록 기계적으로 자동화할 수 있다는 것은 컴퓨터 발전 역사의 핵심이라고 할 수 있다[24][26].

그런데, 추상화의 개념을 국가마다 학자마다 다르게 정의하고 있는 부분이 있어 Computational Thinking 교육에서 난맥상을 드러내고 있다. 따라서 본 논문에서는 추상화와 관련된 여러 개념들에 대한 정의를 비교·분석하여 합리적으로 개념을 정의하고자 한다.

2. Computational Thinking과 추상화에 대한 개념

2.1 영국 CAS[1]

영국의 Computing At School(이하 CAS)에서는 Computational Thinking을 문제를 해결하고, 제품, 프로시저 및 시스템을 좀 더 잘 이해할 수 있게 하는 논리적 추론을 포함한 인지적 혹은 사고 절차로 규정하고 있다. 또한 이것은 알고리즘적 사고, 분해, 일반화 혹은 패턴 인식, 추상화 혹은 표현 및 평가를 포함한다. 이 중에서 일반적인 추상화 절차와 관련된 개념은 추상화, 분해 및 일반화(패턴)으로 볼 수 있다.

CAS에서 정의하는 추상화는 문제나 시스템에 대한 사고를 보다 쉽게 할 수 있도록 만드는 것이며, 불필요한 상세 내용을 삭제하여 대상을 보다 이해하기 쉽게 만드는 절차라고 서술하고 있으며 추상화의 기술은 문제의 중요한 것은 그대로 두고, 상세한 내용을 적당하게 숨겨서 문제가 쉽게 되도록 하는 것이다. 즉, 추상화의 핵심 부분은 시스템을 가장 적절한 표현 방법을 선택하는 것이다.

분해는 대상물을 그것의 구성 요소의 관점에서 생각하는 방식으로 정의하고 있다. 구성 요소들은 개별적으로 이해되고, 해결되고, 개발되며 평가될 수 있다. 이러한 방법은 복잡한 문제를 보다 쉽게 해결하며, 소설을 보다 쉽게 이해하며, 큰 시스템을 보다 쉽게 설계할 수 있도록 해 준다. 이와 같은 하향식의 방법은 문제를 해결할 때 최초로 개발되어지는 방법이기도 하지만 해결 방법을 이해하는 좋은 방법이기도 하다.

또한 일반화를 패턴, 유사성 및 연관성을 식별하고, 그러한 면들을 활용하는 것으로 정의하고 있다. 이것은 유사한 문제에 대한 과거의 해결 방법과 경험에 기반하여 새로운 문제를 빠르게 해결하는 방법이다.

2.2 영국 CAS LONDON[8]

영국의 Computational At School LONDON(이하 CAS London)에서는 Computational Thinking이 컴퓨팅 교육의 새로운 정규 프로그램의 핵심이라고 정의하였으며 추상화 관련 개념을 다음과 같이 기술하고 있다.

CAS London에서는 분해를 문제, 알고리즘, 대상물, 절차 및 시스템을 부분별로 사고하는 방식으로 정의하고 있다. 각 분리된 부분은 독립적으로 이해하고, 해결하고 평가할 수 있다. 이것은 복잡한 문제를 보다 쉽게 해결하고 큰 시스템을 보다 쉽게 설계할 수 있게 해 준다고 하였다.

또한 추상화는 문제 혹은 시스템을 좀 더 쉽게 사고할 수 있는 방식으로 간단하게 상세함을 숨기는 것, 즉 불필요하고 복잡한 것을 제거하는 것으로 정의하고 있다. 이 기술은 적당한 상세 요소를 선택하여 숨김으로써 중요한 것을 잃어버리지 않으면서 문제를 쉽게 하는 것이다. 전체 시스템뿐만 아니라 복잡한 알고리즘을 작성할 때 그것을 쉽게 만들 수 있는 방식으로 사용된다.

일반화는 해결되었던 이전에 해결하였던 문제에 기반하여 새로운 문제를 빠르게 해결하는 방식으로 정의하고 있다. 어떤 특정한 문제를 해결한 알고리즘을 취하여 그와 유사한 문제에 적용하여 문제를 해결할 수 있다. 따라서 새로운 문제를 해결할 때마다 이러한 일반화된 해결 방법을 적용할 수 있다.

2.3 영국 BBC Bitesize[2][3][4][5]

영국의 BBC에서는 교육 관련 콘텐츠를 제공하고 있는데 이곳에서 제공하는 Computational Thinking 관련 내용은 다음과 같다.

BBC에서는 Computational Thinking이 복잡한 문제

를 문제가 무엇인지 이해할 수 있도록 하고, 해결책을 개발할 수 있도록 해준다고 설명하면서 Computational Thinking이란 컴퓨터 과학 기술을 사용하여 사람과 컴퓨터가 이해할 수 있는 방법으로 해결책을 개발하고 표현할 수 있도록 하는 문제-해결 방법이라고 정의하고 있다[4].

분해는 복잡한 문제나 시스템을 작은 부분으로 분해하여 좀 더 관리하기 쉽고, 이해하기 쉽게 하도록 하는 것이라고 정의하고 있으며, 좀 더 작은 부분들은 개별적으로 검사되고, 해결될 수 있다고 하였다[3].

패턴 인식은 좀 더 효율적으로 복잡한 문제를 해결하기 위하여 유사성이나 패턴을 찾아내는 것으로 정의하였다[5].

또한 추상화는 필요한 것에 집중하기 위하여 필요하지 않은 아이디어나 상세한 것을 분리하거나 제거하는 프로세스라고 정의하고 있다[2].

2.4 영국 컴퓨터 과학 교육과정[6]

영국의 컴퓨터 과학 교육 과정에서는 컴퓨터 과학의 키 프로세스를 Computational Thinking으로 정의하고 있다. 즉, Computational Thinking은 우리 주변의 세계를 computation의 관점에서 인식하고, 자연적 혹은 인공적인 시스템과 프로세스에 관하여 Computing으로부터 이해하고 원인을 찾는 도구이자 기술이며, Computational Thinking은 컴퓨터보다는 사람이 행하는 것으로 논리적, 알고리즘적, 재귀적, 추상적으로 사고하는 능력을 포함한다고 기술하고 있다. Computational Thinking을 추상화와 프로그래밍으로 크게 구성하였으며, 이 중에서 본 연구와 관련되는 영역은 추상화이다.

Computational Thinking의 핵심 과제는 연구하거나 구성하고자 하는 시스템의 크기와 복잡도에 관한 것으로 설명하고 있다. 즉, 복잡도를 관리하는데 사용되는 기술이 추상화이다. 추상화의 절차를 모델화, 분해 및 일반화와 같은 여러 형태를 가지고 있으며 각 경우에 복잡한 상세 내용을 숨김으로써 복잡도를 관리한다.

모델화는 실세계의 이슈, 시스템 혹은 상황의 표현을 개발하는 프로세스이다. 이것은 특정 목적에 관한 관점에서 중요한 것을 포착하고 나머지는 생략하는 것이다. 이것은 서로 다른 목적에는 다른 모형을 필요로 한다. 또한 특정 상황에서는 하나 이상의 모형을 필요로 하기도 한다.

그리고 문제는 부-문제로 분해되고, 해결하여 해결 방법들을 합성함으로써 원래 문제를 해결할 수 있다.

또한 복잡함은 특정한 예를 일반화함으로써 회피할 수 있는데 이것은 예제들 사이의 공통점과 차이점을 명시함으로써 수행할 수 있다.

2.5 미국 ISTE - Computation Thinking 교사용 자료 2판[14]

미국 ISTE에서 Computational Thinking 교육을 위해 개발한 교사용 자료에서는 Computational Thinking을 구성 요소로 데이터 수집, 데이터 분석, 데이터 표현, 문제 분해, 추상화, 알고리즘 및 프로시저, 자동화, 시뮬레이션, 병렬화 등 9개를 언급하고 있다.

이중에서 추상화와 관련된 개념에는 문제 분해와 추상화가 있는데 문제 분해는 문제를 해결 가능한 수준의 작은 문제로 나누는 것으로 정의하였다.

또한 추상화는 문제 해결을 위해 반드시 필요한 핵심 요소를 파악하고, 복잡함을 단순화하는 것으로 정의하였다.

2.6 미국 CSTA[22]

미국 CSTA에서는 K-12 컴퓨터 과학 표준의 strand를 Computers and Communications Devices, Computational Thinking, Computing Practice & Programming, Collaboration, Community and Global, and Ethical Impacts 등 5개로 구성하였다. 특히, Computational Thinking은 컴퓨터로 해결할 수 있는 문제에 대하여 분석하고 해결책을 개발하는 탁월한 도구를 제공하는 컴퓨터 과학의 모든 분야를 섞을 수 있는 문제-해결 방법이라고 하였다. Computational Thinking Strand에서 추상화와 관련된 내용을 살펴보면 다음과 같다.

- Grade 3-6
  - 보다 큰 문제를 얘기하는 동안 고려해야 하는 부-문제들의 목록을 만든다.
- Level 2
  - 문제를 부-문제들로 분해하는 추상화를 사용한다.
  - 고수준 언어, 번역, 명령 집합 및 논리 회로들을

포함하는 컴퓨팅에서의 계층 및 추상화 개념에 대하여 이해한다.

- Level 3A
  - 문제의 복잡도를 관리하는 추상화의 가치에 대하여 토론한다.
  - 커다란 문제를 해결하는 전략으로서의 병렬 프로세싱의 개념에 대하여 서술한다.
- Level 3B
  - 새로운 함수와 클래스를 정의하여 문제를 분해한다.
  - 프로세스들을 스레드로 분리하고 데이터를 병렬 스트림으로 나누어 동시성을 보여준다.

이와 같이 CSTA에서는 추상화를 분해와 일반화 모두를 포괄하는 개념으로 보고 있는 것이 특징이다.

2.7 대한민국 정보과 교육과정

대한민국의 중학교 정보과 교육과정에서는 Computational Thinking을 컴퓨터과학의 기본 개념과 원리 및 컴퓨팅 시스템을 활용하여 실생활과 다양한 학문 분야의 문제를 이해하고 창의적으로 해법을 구현하여 작용할 수 있는 능력으로 정의하고 있으며, 추상화 능력과 프로그래밍으로 대표되는 자동화 능력, 창의-융합 능력을 포함하는 것으로 서술하고 있다. 특히, 추상화는 문제를 이해하고 분석하여 문제 해결을 위해 불필요한 요소를 제거하거나 작은 문제로 나누는 과정이라고 정의하고 있다.

또한, 소프트웨어 교육 운영 지침에서는 추상화를 문제를 작은 단위로 분해하고, 문제 해결에 필요한 요소를 추출하는 것으로 설명하고 있다.

이처럼 대한민국의 정보과 교육과정에서는 모델화와 분해를 모두 포괄하는 것으로 보고 있다.

2.8 데이터베이스에서의 추상화[15]

데이터베이스 시스템의 추상화는 상세한 것을 신중하게 생략한 그 시스템의 모델이다. 삭제할 것을 선택하는 것은 추상화의 의도된 어플리케이션과 사용자에 의해 만들어진다. 목적은 사용자가 어플리케이션과 관련된 시스템의 상세한 것에 주의하고 그 외의 상세한 것은 무시할 수 있게 한다. 집단화는 객체들 사이의 관계를 고수준의 객체로 간주하는 추상화를 말하며 일반화는 유사한 객체들의 집합을 일반적인 객체로 간주하는 추상화를 말한다[15].

3. Computational Thinking의 추상화 개념

3.1 추상화 용어 및 개념

앞에서 언급한 내용들을 살펴보면 추상화가 복잡한 상태를 줄여서 이해하기 쉽도록 단순하게 하는 과정이라는 것은 동일하다. 그러나, 용어와 내용에 있어서는 다음과 같이 약간의 다른 점이 있음을 알 수 있다.

먼저, 추상화에 대한 정의이다. K-12 학생들에게 Computational Thinking을 교육하기 위한 자료인 미국 ISTE의 교사용 자료와 영국의 CAS 자료에서는 추상화를 불필요한 상세함을 제거하여 복잡함을 감소시키기 위한 과정이라고 정의하고 있다. 즉, 추상화를 분해와 같은 수준의 용어로 사용하고 있다. 반면에 컴퓨터 과학을 교육하기 위한 자료인 영국의 컴퓨터 과학 교육과정과 미국 CSTA의 K-12 컴퓨터 과학 표준, 대한민국의 정보과 교육과정 그리고 데이터베이스 추상화 관련 논문에서는 추상화라는 용어를 복잡함을 감소시키기 위한 일련의 과정으로 정의하고 하위의 개념으로 집단화, 일반화 및 모델화 등으로 세분하고 있다. 따라서 추상화 개념에 대한 명확한 체계를 확립하는 것이 중요하다.

컴퓨터 과학에서 추상화(Abstraction)는 불필요한 것을 제거하여 복잡함을 감소시키는 프로세스로 볼 수 있다. 따라서 공통적인 요소를 통합하여 정의하는 일반화(Generalization) (혹은 패턴 인식)와 여러 요소들을 모아서 정의하는 집단화(Aggregation)가 이에 속한다고 할 수 있다. 또한 영국의 컴퓨터 과학 교육과정에서는 불필요한 상세한 것들을 제거하는 모델화(Modeling)도 이에 포함시키고 있다.

한편 일반화된 사항들을 각각의 개체에 맞게 적용한 특수화(Specialization)이나 개별 요소로 분리하는 분해(Decomposition)는 추상화의 반대 방향이라고 볼 수 있다. 또한 상세한 사항들을 모두 언급하는 것도 추상화의 반대 방향의 프로세스라고 볼 수 있는데 본 논문에서는 이를 실체화(Substantialization)로 정의하고자 한다. 또한 추상화와 반대되는 방향의 특수화, 분해, 실체화의

프로세스를 구체화(materialization)로 정의하고자 한다. 이를 도표로 표현하면 <Table 1>과 같다.

<Table 1> Abstraction Processes

Abstraction		
Modeling	Aggregation	Generalization
↕	↕	↕
Substantialization	Decomposition	Specialization
Materialization		

3.2 효율적인 문제 해결 방법

앞에서 언급한 바와 같이 추상화는 필요하지 않은 것을 제거하여 압축시키는 프로세스라고 볼 수 있다. 그런데, 이와 같이 압축시키면 좀 더 쉽게 문제를 해결할 수 있을까? 반드시 그렇지는 않다. 이를 각 개념 별로 분석해 보면 다음과 같다.

먼저, 집단화 프로세스에 대한 분석이다. 일반적으로 제시되는 문제는 처음에는 의미가 압축되어 표현되는 경우가 많이 있다. 따라서 문제를 직접 해결하기에는 너무 복잡한 문제인 경우가 많이 있다. 이러한 경우에 개발자들은 문제를 작은 문제로 분해하여 해결하려고 한다. 이와 같이 집단화 프로세스의 경우에는 집단화의 반대 방향인 분해 프로세스가 문제를 좀 이해하고 해결하기 쉽게 하는 방향이라고 볼 수 있다. 특히 학생의 경우에는 많은 경우에 문제를 분해해야 해결할 수 있다.

다음으로 일반화 혹은 패턴 인식의 경우에는 공통적인 요소를 공용으로 활용하게 되므로 통합적으로 관리하는 것이 문제 해결을 쉽게 하는 경우가 많다고 볼 수 있다. 특히, 과거의 문제 해결 방법을 일반화하여 현재에 활용하는 것은 문제를 보다 쉽게 해결할 수 있는 방법이라고 할 수 있다. 영국의 CAS, BBC의 Bitesize와 컴퓨터 과학 교육과정에서는 Computational Thinking 교육 자료에서는 일반화를 패턴, 유사성 및 연관성을 식별하고, 그러한 면들을 활용하는 거나 특정한 예를 일반화함으로써 복잡함을 회피할 수 있는 과정으로 정의하고 있다. 다만, 영국의 컴퓨터 과학 교육과정에서는 일반화를 예제들 사이의 공통점과 차이점을 명시함으로써 수행할 수 있다고 설명하고 있는데 비하여 CAS와 BBC의 Bitesize에서는 유사한 문제에 대한 과거의 해결 방법과 경험에 기반하여 새로운 문제를 빠르게 해결하는 방법으로 설명하고 있는 것이 차이점이다. 이러한 차이점은 문제 해결의 공통 패턴이 존재하는 시점의 차이라고 볼 수 있다. 즉, 영국의 컴퓨터 과학 교육과정에서 언급하는 것은 동일한 시점에 존재하는 공통된 문제 해결 방법에 대한 것이며, 영국의 CAS나 BBC의 Bitesize 등에서 언급하는 것은 과거와 현재에 존재하는 공통된 문제 해결 방법을 연결하는 것이라고 볼 수 있다.

모델화의 경우는 필요하지 않은 것들을 제거하는 것이므로 모든 것을 상세하게 표현하는 것보다는 문제 해결에 필요한 것만 활용하는 것이 문제를 쉽게 해결할 수 있는 방법이라고 볼 수 있다. 특히 모델화는 영국의 컴퓨터 과학에서 사용하고 있는 용어이며, 영국 CAS와 미국의 CSTA의 자료에서는 추상화로 표현하고 있다. 즉, Computational Thinking 교육 자료에서는 추상화를 “불필요한 상세 내용을 삭제하여 대상을 보다 이해하기 쉽게 만드는 과정”이라고 정의하고 있으며, 컴퓨터 과학 교육 과정에서는 모델화를 “특정 목적에 관한 관점에서 중요한 것을 포착하고 나머지는 생략하는 것”으로 정의하고 있다. 특히, 복잡한 런던의 지하철 노선도를 필요에 따라 단순하게 표현하는 것으로 동일한 예를 기술하고 있기 때문에 둘은 동일한 개념을 지칭하고 있는 것으로 간주할 수 있다. 또한, 미국 CSTA의 Computational Thinking 교육 자료에서는 추상화를 “문제 해결을 위해 반드시 필요한 핵심 요소를 파악하고, 복잡함을 단순화하는 것”으로 정의하고 있으므로 이들 3개의 개념은 동일한 의미를 내포하고 있는 것으로 간주할 수 있다.

따라서 문제를 쉽게 이해하고 해결할 수 있는 추상화 관련 프로세스는 일반화(혹은 패턴 인식), 분해, 모델화 프로세스라고 할 수 있다.

3.3 교실에서의 추상화 활동

Computational Thinking를 위한 활동 중에서 교실에서 수행할 수 있는 추상화 관련 활동은 정리하면 다음과 같다.

3.3.1 모델화

영국의 컴퓨터 과학 교육과정에서는 모델화로, 영국의 CAS와 미국의 ISTE에서 언급하는 추상화 활동으로 부르고 있는 활동으로 학교에서 수행할 수 있는 활동은

ISTE[14]에서는 수행할 수 있는 학년별 활동의 예를 다음과 같이 제시하고 있다.

- [K-2] 다양한 크기와 색깔의 세 변을 가진 도형을 추상화하면 삼각형이라고 함
- [3-5] 이야기를 듣고, 핵심 항목을 반영하여 적당한 제목을 결정하라.
- [6-8] 역사의 한 시대를 공부한 후에 그 시대를 대표하는 기호, 테마, 사건, 주요 인물 및 가치를 식별하라.
- [9-12] 현 시대의 본질적인 특성을 분석하여 정치적으로 현재와 가장 비슷한 시대를 선택하라.

또한, 과목별로 수행할 수 있는 예는 다음과 같다[14].

- [컴퓨터 과학] 함수를 수행하는 명령을 반복하는 집합을 캡슐화하기 위한 프로시저 사용, 조건, 루프, 재귀 등 사용
- [수학] 대수학에서 변수의 사용, 프로그래밍에서의 함수와 비교하여 대수학의 함수 공부
- [과학] 물리적 개체의 모델 구성
- [사회] 사실의 요약, 사실로부터 결론의 추론
- [어학] 직유와 은유의 사용, 균형 있는 스토리의 작성

영국의 CAS에서 제시하고 있는 교실에서 관찰할 수 있는 활동은 다음과 같다[1].

- 불필요한 상세 요소를 제거함으로써 복잡도를 감소
- 대상물을 표현하는 방법을 선택하여 유용한 방법으로 관리할 수 있도록 함
- 대상물의 복잡함을 숨김 (기능적 복잡도를 숨김)
- 자료 구조 등을 사용하여 자료의 복잡함을 숨김
- 추상화 사이의 관계를 식별
- 해결 방법을 개발할 때 정보를 여과

3.3.2 분해

모든 자료에서 분해라는 동일한 용어를 사용하고 있으며, 학교에서 수행할 수 있는 활동을 ISTE[14]에서는 수행할 수 있는 학년별 활동 예를 다음과 같이 제시하고 있다.

- [K-2] 학교가 있는 지역을 방향별로 분해하라. 방향의 구역들을 합쳐서 전체가 되게 해 본다.
- [3-5] 학교를 “녹색화” 시키기 위한 계획을 개발하라. 폐지나 캔 등의 재활용, 전기 절약, 음식 쓰레기의 퇴비 사용과 같이 전략을 분리한다.
- [6-8] 월간 신문의 발행을 계획하라. 역할, 책임, 일정 및 프로젝트를 완료하는데 필요한 자원 등을 식별한다.
- [9-12] “록스타가 되기 위해 무엇을 해야 하는가?”와 같은 큰 문제를 고려하라. 그것을 좀 더 작은 부분으로 분해한다. 학생의 통제 하에 있는 변인과 외부의 요인에 의해 결정되는 변인들에 대하여 토론한다.

또한, 과목별로 수행할 수 있는 예는 다음과 같다[14].

- [컴퓨터 과학] 객체와 메소드 정의 - 메인과 함수 정의
- [수학] 수식에서 연산의 순서 적용
- [과학] 종류별로 분류
- [어학] 개요의 작성

영국의 CAS에서 제시하고 있는 교실에서 관찰할 수 있는 활동은 다음과 같다[1].

- 좀 더 쉽게 작업할 수 있게 대상물을 구성 요소로 분해
- 같은 방법으로 해결될 수 있는 같은 문제의 좀 더 간단한 버전으로 문제를 분해 (재귀 및 분해 정복 전략)

3.3.3 일반화 (패턴 인식)

영국의 컴퓨터 과학 교육과정 및 CAS, BBC 등에서 사용하고 있는 개념인 일반화(패턴 인식) 프로세스를 교실에서 관찰할 수 있는 활동은 다음과 같다[1].

- 대상물의 패턴 및 공통점 식별
- 해결 방법 혹은 해결 방법의 부분을 적응하여 유사한 문제의 전체에 적용
- 아이디어나 해결 방법을 하나의 문제 영역으로부터 다른 것으로 전이

3.4 추상화 개념의 구현

학생들이 효율적으로 문제를 해결하기 위해서는 추상

화 프로세스 중에서 모델화, 분해 및 일반화 프로세스를 활용하는 것이 일반적이다. 그런데 이러한 사고 과정은 표현하지 않으면 교육하기 매우 어려워진다. 학생들의 경우에는 사고력을 훈련하는 단계에 있으므로 교사 혹은 전문가의 조언이 필요한데 사고 과정을 표현하지 않으면 이러한 조언을 구하기가 매우 어려워지게 된다. 따라서 학생들이 수행하는 추상화 프로세스는 반드시 표현해야 할 필요가 있다. 그런데, 이를 프로그램과 별도로 문서화하게 되면 학생들에게는 많은 부담이 되며 결국 생략하는 경우가 빈번하게 발생할 수 있다. 그러므로 알고리즘을 설계하고 프로그램을 작성하는 단계에서도 그대로 표현하는 것이 가장 좋은 방법이라 할 수 있다. 특히, 효율적인 문제 해결 방법에 사용되는 모델화, 분해 및 일반화 프로세스를 프로그램에서 표현하게 된다면 학생들의 사고 훈련과 평가에 많은 도움을 줄 수 있다.

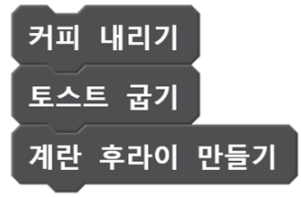
추상화 개념 중에서 모델화, 분해 및 일반화를 교육용 프로그래밍 도구인 스크래치에서 표현하는 방법을 제안하면 다음과 같다.

3.4.1 모델화

모델화는 실세계의 이슈, 시스템 혹은 상황의 표현을 개발하는 프로세스로서 어떤 목적을 위하여 중요한 상황의 관점을 포착하고 나머지는 생략하는 것이다. 즉, 지하철의 노선도 혹은 애니메이션을 위한 스토리보드 등이 이에 속한다. 또한 다른 목적을 위해서는 다른 모형이 필요하며, 어떤 경우에는 하나 이상의 모형이 필요하기 한다[6].

따라서, 모델화의 구현은 차별화된 방법이 없으며, 개발자들이 상황을 인식하고 이를 올바르게 구현하느냐에 달려 있다고 할 수 있다. 이는 개발자 혹은 학생들이 상황을 인식하는 관점에 따라 달라질 수 있으므로 창의성 개발에 도움을 줄 수 있다. 따라서어떻게 구현하느냐 보다는 목적에 맞게 상황을 인식하고 모델화하였느냐에 초점을 맞추어야 한다.

예를 들어, 아침 식사를 준비한다고 가정해 보자. 아침 식사를 준비하는 것에는 여러 가지가 있을 수 있다. 국을 끓이기도 하고, 밥도 해야 하고, 반찬도 준비해야 한다. 또한 토스트를 구울 수도 있고, 커피, 주스 혹은 우유를 준비할 수도 있다. 그 외에 과일을 준비할 수도 있고 물을 준비하기도 한다. 그러나 어느 하루의 아침을 준비하기 위해서 앞에서 언급한 모든 것을 해야 하는 것은 아니다. 그날 아침에 필요한 것만 준비하면 되는 것이다. 즉, 불필요한 것은 삭제하고 필요한 것만을 준비하면 되는 것이다. 이와 같이 아침 식사 준비를 모델화하여 표현하면 (Fig. 1)과 같다[1].



(Fig. 1) Example of Modeling

3.4.2 분해

분해는 복잡하고 큰 문제를 해결할 수 있는 크기의 작은 문제로 분해하여 문제를 해결하는 방식이다. 따라서, 문제를 해결하는 사고 방식이라고 할 수 있다. 이러한 방식의 문제 해결은 컴퓨터 과학에서의 분해 및 정복 (Divide and Conquer) 방식으로 효과가 입증된 바 있다. 그러나, 이러한 사고 활동은 학생들에 대한 교육의 결과인 프로그램에 제대로 드러나지 않는다는 문제점을 가지고 있다.

따라서 학생들이 분해 프로세스를 수행하는 것을 표현할 수 있도록 하는 것이 중요하다. 일반적인 프로그램 개발 프로세스에서는 설계 문서를 작성하고 이를 바탕으로 프로그래밍하므로 설계 문서를 통하여 문제를 분해한 결과를 알 수 있으나 학교에서는 여러 제약 사항으로 인하여 이를 수행할 수 없는 경우도 많이 발생한다. 그러므로 문제의 분해 작업을 프로그램 자체에 표현하는 것이 중요하다. 이러한 작업을 하기 위해서는 프로그래밍 언어의 부프로그램인 프로시져 혹은 함수의 기능을 이용하면 가능하다. 예를 들어, 대표적인 교육용 프로그래밍 언어인 스크래치에서는 “추가 블록”을 사용하여 분해된 문제를 표현할 수 있다.

예를 들어, 대포가 포탄을 발사하는 과정은 여러 단계를 거치게 된다. 이와 같은 발사 단계를 처음부터 상세하게 구현하려면 작업이 복잡해지게 된다. 그런데 이

를 “장전하기”, “발사하기” 및 “폭발하기” 등으로 분해하여 표현하면 각각에 대해서는 쉽게 구현할 수 있다. 이와 같이 대포를 발사하는 과정을 분해하여 표현하면 (Fig. 2)과 같다.

(Fig. 2) Example of Decomposition

3.4.3 일반화

일반화는 유사한 패턴을 가지는 문제 해결 방법을 별도의 모듈로 구현하고 이를 필요한 때 사용하는 문제 해결 방법이다. 특히, 유사한 문제를 해결한 과거의 방법을 사용하기도 하고, 문제를 해결하는 과정에서 유사한 모듈이 필요한 경우에 만들어 사용하기도 한다. 특히, 교육용 프로그래밍 도구에서는 사용자들의 프로그램을 공유하는 기능이 있으므로 유사한 문제를 해결한 다른 사람의 해결 방법을 참고하여 문제를 해결할 수도 있다.

예를 들어, 숫자의 제곱을 계산하는 문제는 숫자가 다르더라도 연산하는 방식은 비슷하다. 따라서 공통적인 연산 방식을 별도로 구현하고 이를 필요한 경우에 불러서 사용하면 매우 효과적으로 문제를 해결하 룻 있다. 이와 같이 숫자의 제곱을 구하는 과정을 일반화하여 표현하면 (Fig. 3)과 같다.

(Fig. 3) Example of Generalization

4. 평가 기준

앞에서 언급한 바와 같이 Computational Thinking을 활용한 문제 해결에서 추상화와 관련된 문제 해결 과정은 모델화, 분해 및 일반화 과정이라고 할 수 있다. 이러한 사고 과정은 Computational Thinking을 사용하여 문제를 해결하는 과정 중에서 매우 중요한 부분이라고 할 수 있다. 따라서, 이를 학생들이 올바르게 수행하고 있는지 평가하는 것도 매우 중요한 활동이라고 할 수 있다. 학생들이 Computational Thinking을 활용하여 문제를 해결하고 있는지에 대한 평가 기준을 제시하면 다음과 같다.

모델화는 문제 해결 방법 중에서 불필요한 것을 제거하고 필요한 것만을 선택하는 과정으로 학생들의 모델화 활동에 대한 평가 기준을 제시하면 <Table 2>와 같다.

<Table 2> Evaluation of Modeling Activities

성취 기준	문제 해결 방법에 모델화를 적용하여 효과적으로 표현할 수 있다.
평가 기준	
상	수집된 문제 해결 방법들 중에서 필요한 것은 선택하고 불필요한 것을 제거하여 표현할 수 있다.
중	수집된 문제 해결 방법들 중에서 필요한 것이 무엇인지 판단하여 표현할 수 있다.
하	수집된 문제 해결 방법들 중에서 필요한 것과 그렇지 않은 것을 판단하지 못한다.

분해는 복잡한 문제를 작은 문제들로 분해하여 문제를 해결하는 방식으로 학생들의 분해 활동에 대한 평가 기준을 제시하면 <Table 3>과 같다.

<Table 3> Evaluation of Decomposition Activities

성취 기준	복잡한 문제를 분해하여 문제를 해결할 수 있다.
평가 기준	
상	복잡한 문제를 의미가 있는 작은 문제들로 중복이나 손실 없이 분해할 수 있다.
중	복잡한 문제를 작은 문제들로 분해할 수 있다.
하	복잡한 문제를 작은 문제들로 쪼개지 못한다.

일반화는 문제의 공통 패턴을 인지하여 공통으로 활용할 수 있도록 하거나 이전의 방법을 일반화하거나 문제 해결 과정에 활용하는 것으로 학생들의 일반화 활동에 대한 평가 기준을 제시하면 <Table 4>와 같다.

<Table 4> Evaluation of Generalization Activities

성취 기준	문제의 공통 패턴을 인지하여 일반화된 문제 해결 방법으로 확장할 수 있다.
평가 기준	
상	부여된 문제와 유사한 문제에 대한 이전의 문제 해결 방법을 활용하거나 개발하고 있는 문제 해결 방법들 중에서 공통 요소를 찾아 일반화하여 별도로 관리할 수 있다.
중	부여된 문제와 유사한 문제에 대한 이전의 문제 해결 방법을 탐색하고 일반화하여 문제 해결에 활용할 수 있다.
하	문제의 공통 패턴을 인지하지 못한다.

5. 결론

4차 산업 혁명을 맞이하면서 소프트웨어의 파워는 점점 더 강해질 것으로 예측된다. 따라서 각국에서는 학생들에게 소프트웨어 교육을 하기 시작하였다. 우리나라도 이런 추세에 맞추어 2018학년도부터 초중고등학교에서 소프트웨어 교육을 시행하기로 한 바 있다.

소프트웨어 교육은 Computational Thinking 능력의 배양이라고 볼 수 있으며, Computational Thinking은 학자마다 국가마다 차이를 보이기도 하지만 추상화와 자동화로 이루어진다고 보는 것이 일반적이다. 그런데, 추상화의 경우에 국가마다 분야마다 약간의 차이를 보이고 있다. 이러한 차이는 학생과 교사들에게 혼란을 야기할 수도 있다. 따라서 본 논문에서는 추상화의 개념에 대한 문헌 연구를 통하여 합리적인 정의를 도출하고 추상화 개념에 대한 위계성을 확보하였다. 또한 추상화 개념을 Computational Thinking 교육에서 활용할 수 있도록 구현 방법과 평가 기준을 설정하였다.

이러한 추상화에 대한 연구는 앞으로 소프트웨어 교육을 혼란 없이 효과적으로 진행하는데 도움이 될 것으로 기대된다.

참고문헌

[1] Andrew Csizmadia, Paul Curzon, Mark Dorling, Simon Humphreys, Thomas Ng, Cynthia Selby, John Woollard (2015). Computational Thinking - A Guide for Teachers. Computing At School. [http://computingatschool.org.uk/ computationalthinking]

[2] BBC Bitesize (2016a). Abstraction. [http://www.bbc.co.uk/education/guides/zttrcdm/revision]. (2016. 11. 2)

[3] _____ (2016b). Decomposition. [http://www.bbc.co.uk/education/guides/zqqfyrd/revision] (2016. 11. 2)

[4] _____ (2016c). Introduction to Computational Thinking. [http://www.bbc.co.uk/education/guides/zp92mp3/revision] (2016. 11. 2)

[5] _____ (2016d). Pattern Recognition [http://www.bbc.co.uk/education/guides/zxxbgk7/revision]. (2016. 11. 2)

[6] Computing at School Working Group (2012). Computer Science: A Curriculum for Schools [http://www.computingatschool.org.uk]

[7] Cynthia C Selby, John Woollard (2010). Computational Thinking: The Developing Definition

[8] Developing computational thinking. Teaching London Computing : A RESOURCE HUB from CAS LONDON [https://teachinglondoncomputing.org/resources/developing-computational-thinking] 2016. 9. 20

[9] Devlin, K. (2003, September). Why universities Require Computer Science Students to Take Math. *Communications of the ACM, 46*(9), pp.37-39.

[10] Digital Daily (2013. 4. 18). MISP.Pursuing Computer Programming Education from Elementary and Middle Schools. URL:http://www.ddaily.co.kr/news/news_view.php?uid=103537.

[11] Hazzan, O (1999). Reducing Abstraction Level When Learning Abstract Algebra Concepts. *Educational Studies in Mathematics, 40*(1), pp.71-90.

[12] Hyungshin Choi (2014). Developing Lessons and Rubrics to Promote Computational Thinking. *Journal of The Korean Association of Information Education. 18*(1), pp. 57-64.

[13] Hyungshin Choi · Inkee Jeong · Hyojeong So(2014). Computational Thinking Framework-based Analysis of Afterschool Scratch Team Project Experiences. *Journal of The Korean Association of Information Education. 18*(4), pp. 549-558.

[14] ISTE, CSTA, NSF (2011). Computational Thinking - Teacher Resources 2nd edition.

[15] John Miles Smith and Diane C.P. Smith (1977). Database Abstraction: Aggregation and Generalization. *ACM Transactions on Database Systems. 2*(2), pp.105-133.

[16] Kim. Y. A. et al. (2015). A Study of Development of the Operation Guideline for SW Education. KERIS. CR 2015-3.

[17] Kramer, J. (2007). Is Abstraction the Key to Computing?. *Communications of the ACM, 50*(4), pp.36-42.

[18] Kyung-Hoon Kim · Oh-Han Kang · Yung-Sik Kim · Yoon Young Kim · Seo In Soon · Seong Jin Ahn · Soon Young Jung · Hyun Jong Choe(2012). A Study on the Direction of Informatics Education Strategies Based on the Creative Problem Solving to Improve Core Competencies. KICE Report RRC 2012-7.

[19] National Academies of Sciences (2010). Report workshop on the Scope and Nature of Computational Thinking. Washington DC: National Academies Press.

[20] Papert, S. & Resnick, M. (1995). Technological Fluency and the Representation of Knowledge. Proposal to the National Science Foundation. MIT Media Laboratory.

[21] Soohwan Kim (2015). Effects of Teaching and Learning Strategies of Learner-Centered Learning for Improving Computational Thinking. *Journal of The Korean Association of Information Education. 19*(3), pp. 323-332.

[22] The CSTA Standards Task Force (2011). K-12 Computer Science Standards Revised 2011, CSTA.

[23] Wing, J. M. (2006). Computational Thinking. Communications of the ACM. 19-3. pp.33-35.

[24] _____ (2008). Computational Thinking and Thinking about Computing. Philosophical Transactions of the Royal Society. 366. pp. 3717-3725.

[25] _____ (2011). Computational Thinking - What and Why?. CMU Research Notebook. Retrieved from http://lin-k.cs.cmu.edu/article.php?a=600.

[26] Youngho Seo · Miryeong Yeom · Jonghoon Kin (2016). Analysis of Effect that Pair Programming Develop of Computational Thinking and Creativity in Elementary Software Education. *Journal of The Korean Association of Information Education. 20*(3), pp. 219-234.

[27] Young Jun Lee · Seoung Hye Paik · Shin Je Hon g · HeonChang Yu · Inkee Jeong · SangJin An · JeongWon Choi · SeongKyun Jeon(2014). Research for Introducing Computational Thinking into Proimary and Secondary Education. Korea Foundation for the Advancement of Science and Creativity.

[28] Youngsik Jeong (2015). Proposal of Standard Model for Software Education in the Elementary School. Proceeding of Public Hearing for SW Education in the Elementary and Middle School. pp.3-40.

저자소개

정 인 기

1988 고려대학교 전산과학과 (이학사)

1990 고려대학교 대학원 수학과 전산학전공(이학석사)

1996 고려대학교 대학원 전산과학과 전산학전공 (이학박사)

1997～현재 춘천교육대학교 컴퓨터교육과 교수

관심분야 : 컴퓨터과학교육, 프로그래밍 교육

E-Mail : inkey@cnue.ac.kr