Journal of Korea Game Society 2018 Feb; 18(1): 7-18
http://dx.doi.org/10.7583/JKGS.2018.18.1.7

게임 프로그래밍

코딩퍼즐게임의 코딩 성취도 평가 시스템의 설계와 구현

서범주, 조성현
홍익대학교 게임학부
{bseo, scho}@hongik.ac.kr

Design and Implementation of Students' Coding Assessment System for a Coding Puzzle Game

Beomjoo Seo, Sung Hyun Cho
School of Games, Hongik University

요 약

코딩 교육이 정규 교과과정에 편재됨에 따라 스크래치 혹은 엔트리로 대변되는 다양한 퍼즐 기반 코딩 학습 플랫폼 및 프로그램이 많이 배포되어 이용되고 있다. 본 논문에서는 학습자가 코딩 교구를 사용할 때 학생의 코딩 수행 능력의 수준을 정량적으로 평가할 수 있는 코딩 성취도 평가 시스템의 성취도 모델을 제안하고, 이 모델의 여러 가지 이슈에 대하여 논의한다. 또한 본 연구팀이 개발한 코딩 플랫폼인 "코딩퍼즐" 시스템의 사례 연구를 통하여 본 연구팀이 제안한 성취도 모델의 유용성을 보여준다. 그리고 현재 운영 중인 "코딩퍼즐" 플랫폼의 성취도 평가 시스템에 대한 상세 설계 및 구현을 기술한다.

ABSTRACT

As coding education is ubiquitous in elementary or higher school curriculum, puzzle-based coding platforms such as Scratch or Entry have been popularly deployed and employed by many Korean coding educational institutions. In this article, we propose a quantitative students' coding assessment methodology and discuss several issues of the method. Besides, we show its usefulness through case studies shown in our proprietary coding platform called "CodingPuzzle" system. Additionally, we describe detailed design and implementation issues of our coding assessment system that has already been ported to the CodingPuzzle system and is currently under operation.

Keywords : Assessment(성취도 평가), Coding(코딩), Computational Thinking(컴퓨팅 사고), Puzzle(퍼즐), Modelling(모델링)

Received: Jan. 2. 2018 Revised: Feb. 13. 2018
Accepted: Feb. 19. 2018
Corresponding Author: Sung Hyun Cho (Hongik University)
E-mail: scho@hongik.ac.kr

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

전 세계적으로 산업구조가 소프트웨어 기반기술을 중심으로 재편됨에 따라 학생들의 컴퓨팅 사고력[1]을 고양시키기 위한 다양한 노력이 시도되고 있다. 특히 국내의 경우 초·중·고등학교에서 단계적으로 소프트웨어 코딩교육이 의무화됨에 따라 게임과 같이 놀이를 통한 창의력과 문제해결력을 향상시키는 것을 목표로 하는 코딩교구 시장이 지속적으로 커지고 있다.

국외에서는 Kordable, Quetzal, LightBot, Cargo-Bot 등 다양한 코딩교구들이 활발하게 활용되고 있으며 스크래치의 기본 기능을 재구성한 엔트리로 대변되는 코딩 플랫폼이 널리 사용되고 있다[2-7].

본 연구의 평가 대상이 되는 코딩퍼즐류의 게임 제품들에서 제공하는 명령어 및 해당 명령어들의 기능을 살펴보면 다음과 같다. Kodable은 5세 이상 저연령층 아동을 대상으로 한 2D 기반 모바일 앱을 제공하기 때문에 상, 하, 좌, 우 네 방향으로 이동하는 명령어를 제공하며 반복문, 조건문, 함수 등을 제공하고 있다[2].

Quetzal은 레고사의 마인드스톰 브릭을 제어하기 위해 고안된 탠저블 프로그래밍 언어로서 브릭을 이용해 반복, 분기, 매개변수, 함수 등을 사용할 수 있게 고안되었다[3].

LightBot은 네 방향 이동 명령어 대신 전진 이동, 90도 좌회전, 90도 우회전 등 3개의 이동명령어를 사용한다[4]. 지원하는 명령어는 3차원 공간을 대상으로 하고 있으나, 실제로는 2차원 공간에서 구현되어 있다. 본 연구에서 사용하는 코딩퍼즐게임은 LightBot과 유사한 명령어 체계를 갖추고 있으나, 3차원 공간상에서 게임을 즐길 수 있다는 점이 다르다.

Cargo-Bot은 좌우 이동이 가능한 기중기로 색깔이 다른 화물상자들을 들어 올리거나 내리며 이동시키고 과제로 주어진 형태에 맞게 배치하는 고난이도 퍼즐이다[5]. 기존 퍼즐게임과는 달리 재귀용법을 구현하고 있기 때문에 난이도가 매우 높아서 초등학교 고학년부터 중등 교육과정 이수자에게 적합한 퍼즐게임이다.

이러한 다양한 코딩 플랫폼이 활용되고 있으나 이에 대한 학습자의 컴퓨팅 사고력의 개선 정도를 평가하는 온라인 학습자 평가 시스템은 아직 초보단계이다.

국외의 경우에는 Dr.Scratch에서 컴퓨팅 사고력에 필요한 개념을 간단한 척도로 제시하는 기법이 활용되고 있으나, 아직 해당 척도는 매우 기초적인 제안 수준에 머무르고 있다[6]. 국내에서는 Dr. Scratch에서 적용한 방법론을 활용하는 수준의 평가 방법론이 최근 제시되고 있으나, 위와 같이 다양한 형태의 코딩블록 프로그래밍에 대한 컴퓨팅 사고력에 대한 측정 방법론은 전무하다[7].

본 연구에서 활용하는 코딩 플랫폼은 스크래치나 엔트리와 같이 범용성이 높은 시각적 프로그래밍도구는 아니지만, 저연령층 코딩교구인 Kodable, LightBot 등과 같은 아이콘 중심의 블록코딩 교구 등에 적합한 컴퓨팅 사고력 측정 방법론 및 이를 구현하는 시스템의 설계 및 구현을 다루고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 자동화된 코딩 품질 평가 방법론에 대해서 알아보고, 3장에서는 본 논문에서 다루는 코딩퍼즐류의 게임에 대해 소개하고, 코딩 평가를 위한 코드구조 비용, 코드성능 비용 모델링 및 이슈들을 토의한다. 4장에서는 제시한 비용 모델에 대한 사례 연구를 통해 비용 모델의 유용성을 검증한다. 5장에서는 본 논문에서 제안한 비용 모델을 적용한 시스템의 설계 및 구현 방법을 소개하고, 6장에서는 본 연구의 결과와 향후 과제를 기술한다.

2. 관련연구

자동화된 프로그래밍 평가 방법론은 실행 가능한 프로그램을 제출하고 그 프로그램의 동작 여부로 판별하는 방식으로 이미 1960년에 출발했다[8].

자동화된 평가 모델들은 소스 코드로부터 컴파일 여부, 실행 가능 여부, 실행 결과 동일 여부, 실행 시간 초과 등 실행 시간 평가를 중심으로 하고 있으며, 초창기 툴 중심의 평가 방법론에서 웹기반 평가 방법론으로 점진적으로 발전해 왔으나 평가 모델 자체는 크게 변화하지 않았다[9].

실행 측면을 강조한 평가 모델 대신 내부 코드 품질에 대한 자동화 평가 방식은 소프트웨어공학에서 주로 다루는 분야이다. 초기 평가 프레임워크인 Datrix에서는 소스코드를 트리형태의 추상 구문 트리(Abstract Syntax Tree)로 추상화시킨 후 아키텍처 그래프, 데이터 선언 그래프, 제어 흐름 그래프, 데이터 흐름 그래프 등으로 변환하여 21개의 함수 관련 평가 인자들을 추출한 후 유사 기능의 복제여부를 판별하는 용도 등 다양한 코드 품질 평가를 진행한 바 있다[10]. F. Fontana 등은 복사된 코드, 많은 기능을 담고 있는 클래스, 읽기 어렵고 유지보수가 어려운 메서드, 긴 함수 입력 인자 등을 가진 나쁜 코드(code smell)를 찾아내기 위한 방법으로 Lines of Code, 복사된 코드 비율, 외부 클래스 데이터 접근 횟수 등의 활용을 제안하고 있다[11]. 이들 연구 방법론들은 매우 복잡한 로직으로 평가 기준 및 측정법 때문에 본 연구가 목표로 하는 코딩퍼즐류의 게임에 직접 적용하기는 어렵다.

Dr.Scratch에서는 컴퓨팅 사고력 함양에 필요한 추상화, 논리적 사고, 동기화, 병렬, 흐름 제어, 상호 작용성, 데이터 표현 등의 측면에 대한 척도를 지표로 사용하고, 코드 분석을 통해 해당 지표는 0부터 3사이 값으로 평가한다[6]. 최종 스코어는 개별 지표의 합산으로 산출하고 있다. 하지만 지정된 기능을 수행하는지 여부를 판정하는데 국한되어 있을 뿐 코드구조를 고려한 분석을 하지는 않는다. 반면, 본 연구에서 사용하는 컴퓨팅 사고의 개념은 함수로 대변되는 추상화 및 분해능력, 무한 반복 사용에 국한되어 있으나, 코드 구조와 명령어 실행 여부에 따라 학습자의 컴퓨팅 사고력의 정도를 정량화한 지표를 제시한다.

국내에서는 엔트리를 활용하여 학습에 참여한 학생들의 산출물을 평가하는 수행평가 도구 검증 방법을 제안하고 있다[7]. 이것은 Dr.Scratch에서 제안한 스크래치용 코드에 대한 분석규칙을 차용하여 유사한 플랫폼인 엔트리에 적용하고 그 결과를 웹을 통해 실시간으로 받을 수 있는 웹 서비스 형태로 제공하고 있다. 본 서비스도 이와 같은 실시간 웹서비스 형태로 제공되고 있으나, 대상이 되는 환경은 블록형 코딩프로그래밍 환경인 코딩퍼즐 게임이다.

3. 성취도 평가 모델링

본 장에서는 주어진 퍼즐코딩의 솔루션에 대한 성취도 평가 모델링 방법론을 소개한다.

3.1 코딩퍼즐 게임 개요 및 문제 정의

본 연구에서 사용하는 코딩퍼즐 게임은 본 연구진과 메이커스랩이 2014년부터 공동개발해온 코딩교육용 퍼즐게임으로서 코딩교육 확산을 위해 무료로 서비스 중인 웹 기반 플랫폼 서비스이다[12].

이 게임에서는 [Fig. 1]에서 예시한 바와 같이 로봇으로 대변되는 학습자가 출발점, 도착점, 수행과제가 제시되어 있는 등각투영된 3차원 그리드 공간을 기반으로 한다. 학습자는 [Fig.1] 화면 하단부에 있는 명령어 입력 창에 로봇이 수행할 제한된 개수의 코딩 명령어들을 입력한다. 실행버튼을 누르면 출발점에서 시작하여 일련의 과제 수행을 거쳐 도착점에 도달하고 원대 복귀 명령어를 통해 과제를 완수하는 것을 목표로 한다.

로봇은 [Fig.1] 왼쪽 하단부에서 제시된 MAIN이라는 명령어 입력창에 입력한 명령어를 순차적으로 실행한다. MAIN 명령어 입력창 옆에는 G1, G2와 같이 추가로 명령어들을 입력할 수 있는 입력창이 제공되고 있다. MAIN, G1, G2에 들어있는 명령어들을 명령어 시퀀스라 하며, 명령어 시퀀스는 원자 명령어와 명령어 시퀀스를 하나의 그룹으

로 만든 함수 명령어로 구성된다.

[Fig. 1] A Sample CodingPuzzle Game Publicly Available from https://codingpuzzle.org

명령어는 [Fig.1] 맨 하단부에 위치한 아이콘 형태로 표현된 명령어들로서 G1, G2 두 개의 함수 명령어와 원자 명령어들로 구성되어 있다. 원자 명령어들은 아이콘들이 왼쪽에서 오른쪽으로 한칸 전진, 90도 좌회전, 90도 우회전, 점프, 액션으로 배치되어 있다. 이들 명령어는 한 번에 하나의 기능만을 수행한다.

액션 명령어([Fig.1]에서 A)는 문맥에 따라 다른 기능을 수행한다. 예를 들어, 로봇이 다음에 이동할 위치에 있는 타일이 오염되어 있으면 정화시키는 기능을 수행하거나, 로봇이 위치한 주위에 적이 있으면 적을 공격하거나, 현재 로봇이 위치해 있는 타일이 순간이동이 가능한 타일(워프 타일)이면 이 타일과 짝 지워진 순간 이동 타일로 순간 전송되거나, 현재 로봇이 위치한 타일이 도착 지점이면 원대 복귀하는 등 다양한 기능들이 중첩되어 있다.

학습자는 이러한 명령어들을 활용하여 퍼즐에서 주어진 미션을 성공적으로 완수한 후 원대 복귀해야 한다. 이때 과제를 완수하는 MAIN에 입력된 명령어 시퀀스를 퍼즐 솔루션이라 한다.

학습자 개인별로 작성한 퍼즐 솔루션에 대한 성취도 평가를 통해 학습자의 프로그래밍 숙련도를 정량적으로 평가하고자 하는 것이 본 논문의 목표이다. 성취도 평가는 동일 코딩 퍼즐에 대하여 퍼즐 솔루션의 비용을 정량적으로 평가한다.

3.2 비용 모델

정량화된 비용 계산을 위해 퍼즐 솔루션의 정보를 정량적으로 평가할 수 있는 비용 지표들을 먼저 논의하고, 명령어들을 반복 사용할 때 고려사항을 논의하며, 함수를 무한 반복 사용할 때 고려사항을 알아본다. 이를 바탕으로 최종적인 비용 모델을 제시한다.

3.2.1 비용 지표

주어진 퍼즐 솔루션은 크게 두 개의 상이한 정보를 담고 있다. 하나는 코드의 간결성, 추상성 정도를 대변하는 코드의 구조에 대한 정보로서 학습자의 코드에 대한 이해도를 평가할 수 있는 지표이다. 다른 하나는 코드가 실제 실행될 때의 성능에 관련한 지표이다. 가장 이상적인 코드는 코드의 구조가 간결하고, 미션 수행에 필요한 기능만을 수행하는 코드이다.

코드 구조 지표

본 연구에서는 코드의 구조를 정량적으로 평가할 수 있는 지표로 **'솔루션 길이'** 혹은 **'코드길이'**를 제안한다. 코드 구조를 가장 단순하게 표현할 수 있는 지표로서 주어진 솔루션의 명령어 시퀀스 안에 명령어들의 수를 의미한다. 이때 명령어가 함수인 경우 해당 함수의 길이만큼 솔루션 길이에 추가된다.

n개 명령어들로 구성된 솔루션 $S(=c_1 \cdots c_n)$와 S에 의해 혹은 그 내부에 있는 함수에서 사용된 함수 명령어 집합 G가 주어질 때, 솔루션 길이 |S|는 다음과 같이 정의된다.

$$|S| = n + \sum_{g_i \in G} |g_i|$$

이때, 함수 명령어의 코드길이 $|g_i|$는 g_i안에 있는 명령어 시퀀스의 길이이다.

동일 기능을 수행하는 코드는 코드길이가 작을수록 코드 효율성이 높다. 예를 들어, 6칸 앞으로 전진하는 두 개의 서로 다른 퍼즐 솔루션 $S_1 = ffffff$, $S_2 = G_2G_2(\ G_2 = fff)$ 의 경우, $|S_1|$=6, $|S_2|$=5 이므로 코드 간결성 측면에서 S_2가 비교우위에 있다고 평가할 수 있다. 이는 반복 사용되는 원자단위 명령어 사용 대신 함수로 추상화하고, 추상화된 함수를 반복 적용함으로써 동일한 미션 수행에 대하여 코드 길이가 작게 된다. 그래서 학습자의 코드에 대한 이해도가 높은 것으로 평가할 수 있다. 학습자는 코드길이를 줄이는 노력을 통해 자신의 추상화 능력, 반복 패턴 탐색 및 적용 등 컴퓨팅 사고력에서 중요하게 생각하는 요소를 향상시킬 수 있다.

코드 성능 지표

코드길이가 코드 구조의 효율성을 평가하는 지표라면 코드의 성능을 평가하는 지표로 다음 두 가지 성능 지표 모델을 제안한다. 먼저 주어진 퍼즐 솔루션이 미션을 완수하기까지 호출되어 실행되는 (중복을 포함한) 명령어들의 '**총 호출횟수**'이다. 그리고 호출된 원자단위 명령어들 중에서 정상적으로 실행되지 않는 명령어들의 총 호출 횟수를 지칭하는 '**무효 명령어 수**'가 있다.

일반적으로 호출된 모든 명령어는 정상적으로 실행되지만 본 연구에서 사용하는 코딩퍼즐 게임에서는 명령어가 호출되지만 아무런 동작을 하지 않는 경우가 발생할 수 있다. 이는 코딩퍼즐 게임에서 의도적으로 제안하고 있는 방식이다. 예를 들어 로봇이 막다른 골목 혹은 절벽 앞에 위치해 있을 때 전진 명령어를 호출하게 되면 막다른 골목 앞을 전진할 수 없거나 절벽에서 떨어지게 된다. 이는 프로그래밍에서 흔히 발생하는 오류이다. 하지만 이 경우 오류로 판정하고 미션 수행 실패를 선언하게 되면 많은 학습자들은 좌절하고 학습 의욕이 저하될 수 있다. 따라서 부정적 영향을 줄이기 위해 비정상적인 상황에서 아무런 동작을 하지 않도록 설계 및 구현되었다. 아무런 동작을 하지 않는 명령어를 No Operation (NOP)라고 한다.

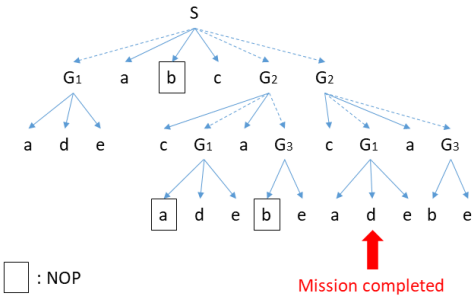
이러한 의도된 코딩퍼즐 게임 구현 방식을 악용하는 경우가 있을 수 있기 때문에(4장에서 상술) 무효 명령어 수를 코드의 성능 평가 지표로 포함하고 있다.

이렇게 코드구조 지표와 코드성능 지표 두 가지를 제안하며 이들 값에 대한 이해를 돕기 위해 다음과 같은 퍼즐 솔루션 예제1을 통해 알아본다.

(퍼즐 솔루션 예제1)

$$S = G_1abcG_2G_2$$
$$G_1 = ade$$
$$G_2 = cG_1aG_3$$
$$G_3 = be$$

본 예제는 임의의 퍼즐에 대한 미션을 완수하는 퍼즐 솔루션 S의 예를 보여주고 있다. 솔루션 S를 트리 형태로 표현하면 [Fig. 2]와 같다.



[Fig. 2] Tree Representation of a Sample Puzzle Solution 1.

여기에서 리프 노드는 원자단위 명령어, 그 이외 노드들은 함수로 구성되어 있다. 함수 호출의 경우에는 점선, 원자 단위 명령어의 경우는 실선으로 표시하고 있으며, 호출되었으나 실행되지 않은 NOP 명령어들은 네모박스로 표현하고 있다. 이 예제에서 솔루션의 코드 길이 |S|는 6 + |G1| +|G2|

+ |G_3| = 15 이다.

총 호출 횟수는 [Fig. 2]에서 표현한 간선의 개수 중에서 미션 완료까지 사용된 간선의 개수와 일치한다. [Fig. 2]에서 명령어 d 실행 후 미션이 종료된다고 가정하면, 총 호출 횟수는 함수 호출수 6, 원자단위 명령어 호출수 16으로 총 호출수는 22이다. 이때 16개의 원자단위 명령어 호출 중에 3회가 실행되지 않았다.

3.2.2 반복 축약 변환 규칙

본 연구에서 사용하는 코딩퍼즐 게임에서는 반복문을 구현하지는 않았지만, 반복문의 기본 개념을 사용하는 것은 비용 계산 모델링에 미치는 영향이 매우 크기 때문에 반복문을 사용할 경우에 비용 계산 모델링에서 고려해야 할 이슈를 알아본다. 다음과 같은 퍼즐 솔루션 예제2를 중심으로 반복 축약에 대해 알아본다.

(퍼즐 솔루션 예제2)

$$
\begin{aligned}
&S_0 = ffffff \\
&S_1 = G_1, && G_1 = ffffff \\
&S_2 = G_2 f, && G_2 = fffff \\
&S_3 = G_3 ff, && G_3 = ffff \\
&S_4 = G_4 G_4, && G_4 = fff \\
&S_5 = G_5 G_5 G_5, && G_5 = ff \\
&S_6 = G_6 G_6 G_6 G_6 G_6 G_6, && G_6 = f
\end{aligned}
$$

예제2는 6칸 앞으로 전진하는 미션을 수행하기 위해, 원자단위 명령어 한 칸 전진을 6회 반복 시행하여 미션을 완수하는 퍼즐을 고려한다. 본 예제에서는 함수를 반복하는 호출이 비용모델에 어떤 영향을 미치는지 조사하기 위해 총 7개의 상이한 퍼즐 솔루션들을 제시하고 있다.

S_0의 경우 6회 반복하는 전진 명령어를 사용하고 있다. 1개의 원자단위 명령어를 6회 사용하고 있으며, 총 호출수와 유효 명령어 수는 동일하다. 코드 성능 측면에서는 최적의 코드 조합이나 코드 구조 측면에서는 개선의 여지가 있다.

S_1에서는 G_1이라는 함수 개념을 도입하여 S_0에서 직접 보이는 명령어 시퀀스를 줄였으나 코드길이는 7, 무효 명령어는 없는 총 호출 수가 7회로 S_0보다 열등하다. 함수를 사용하는 것이 추상화 정도를 높인다고 가정할 수 있으나, 실제 반복 사용하고 있지 않기 때문에 추상화의 당위성이 떨어진다고 평가할 수 있다.

S_2와 S_3는 S_1와 동일한 지표를 가지고 있다. 하지만 S_3를 아래와 같은 S_3'으로 변형하여 함수를 반복 호출하게 된다면 코드길이 측면에서는 S_0와 동등하다고 할 수 있다. 다만 성능 측면에서 총 호출 수 8회(무효 명령어 횟수는 0)로 S_3보다 오히려 저하되었다. S_3와 비교하여 코드는 간결해졌으나 성능은 열악해졌다.

$$S_3' = G_3 G_3, G_3 = ffff$$

S_4와 S_5는 불필요한 명령어가 배제되어 있기 때문에 S_2, S_3'보다 코드 품질이 우수하다. 이들의 코드길이는 모두 5로 S_1과 비교하여 코드 간결성은 높아졌으나, 성능면에서 S_4와 S_5의 총 호출수 각기 8과 9회로 S_1보다 열등하다. 즉, 코드 품질은 우수하나 성능측면에서는 우위를 점하지 못하고 있다.

S_4와 S_5는 어떤 코드가 우수하다고 평가하기는 어렵다. 동일한 코드 성능에 대하여 함수 코드길이를 늘리는 대신 함수의 반복 호출 횟수를 줄이는 것이 좋을지, 아니면 그 반대일지 여부는 모호하다. S_6처럼 극단적으로 함수반복을 늘이고, 함수 코드길이를 줄이는 것이 더 우수하다고 평가할 수는 없을 것이다.

S_6는 함수 추상화를 극대화하여 반복횟수를 최대로 한 극단적인 모델이다. 오히려 S_6의 코드길이는 7, 총 호출수는 12회로 S_1보다도 매우 열등하게 된다. 함수 사용 및 반복 호출을 통한 추상화를 극대화시키는 작업은 실제 코드 품질이나 코드 성능을 저하시키게 되는 모순을 야기한다.

이와 같은 모순된 상황을 제거하기 위해 본 연구에서는 반복 축약 개념을 도입한다. 반복문을 표

현하기 위해 숫자·명령어 혹은 숫자(명령어 시퀀스)와 같이 표기하면 예제2는 다음과 같이 수정할 수 있다.

$S_0 = 6f$
$S_1 = G_1, \quad G_1 = 6f$
$S_2' = 2G_2, \; G_2 = 5f$
$S_3' = 2G_3, \; G_3 = 4f$
$S_4 = 2G_4, \; G_4 = 3f$
$S_5 = 3G_5, \; G_5 = 2f$
$S_6 = 6G_6, \; G_6 = f$

이때, 원자단위 명령어 혹은 함수를 포함한 반복문을 하나의 함수 호출을 추상화한다고 가정하면 반복문은 하나의 코드길이에 상응하게 되며 이에 따라 7개의 서로 다른 솔루션의 코드 길이는 순서대로 다음과 같다: 2, 3, 4, 4, 4, 4, 3.

따라서 반복문을 축약한 변환 규칙 하에서는 수정된 예제2의 코드를 기준으로 평가한다면 함수를 사용한 솔루션 중에서는 S_1이 가장 우수하다고 평가할 수 있으며 직관적인 분석과도 일치한다. S_1과 동일한 코드길이를 갖는 S_6의 경우 총 호출 횟수가 매우 많기 때문에 S_1보다 매우 열등하다고 평가할 수 있다.

여기에서 다룬 반복 축약 변환 규칙은 퍼즐 솔루션 예제2에서 제기된 모순을 설명하는 용도로 사용했다. 또한 이 반복 축약 변환 규칙을 활용하여 다음에서 제시하는 무한 반복시의 비용을 계산하는 모델에 적용할 수 있다.

3.2.3 무한 반복 비용 계산

무한 반복이란 주어진 함수에서 자기 자신을 재귀적으로 호출하는 함수를 말한다. 이때 주어진 무한 반복 구간 동안 미션이 완료되어 반복을 중지하고 실행이 종료되어야 한다.

무한 반복의 경우 다음과 같은 형식으로 표현될 수 있다. 먼저 지정된 함수에서 자기 자신을 직접 호출하는 무한반복으로 다음과 같은 형식을 따른다.

$G = c_1 \cdots c_n G$

이때 명령어 시퀀스 $c_1 \cdots c_n$ 사이에는 무한 반복 구간이 없어야 한다. 만일 $c_1 \cdots c_n$ 사이에 무한 반복 구간이 있다면, G의 반복 종료 시점을 지정하는 G가 호출되지 않는다. 따라서 무한 반복 구간은 내부에 있는 무한 반복 구간을 의미한다.

$S = a G_2 S$
$G_2 = b G_2$

예를 들어, 위와 같은 솔루션에서 무한 반복 구간은 S가 아니라 G_2가 된다. 따라서 여기에서 무한 반복이라 함은 중첩된 여러 개의 무한 반복 구간 중 맨 처음에 나타나는 반복 구간을 의미한다.

무한 반복 구간이 존재하는 경우, 반복문 형태로 표현하면 다음과 같다.

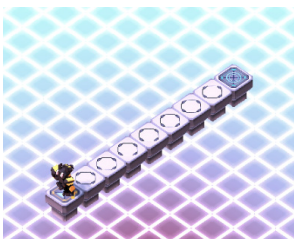
$\infty (c_1 \cdots c_n)$

반복문은 무한 표식 대신 숫자를 기입하는 것으로 단순화되며 무한 반복문도 반복문의 일종으로 취급하여 1만큼 반복문 사용비용과 무한 반복 내 반복 구간을 표현하는 명령어 시퀀스 $c_1 \cdots c_n$의 코드 길이만큼 코드길이를 결정할 수 있다. 따라서 무한 반복 G의 코드길이는 다음과 같이 정의할 수 있다.

$|G| = 1 + |(c_1 \cdots c_n)|$

4. 성취도 평가 비용 모델 평가

앞에서 제시한 비용 모델을 검증하기 위해 본 장에서는 두 개의 자명한 사례 연구를 통해 위에서 제시한 비용 모델을 검증하고자 한다.



[Fig. 3] Case Study 1: Move Forwards

[Fig. 3]에서 제시한 사례는 초급레벨 퍼즐의 전형적인 예제이다. 기본 명령어인 한 칸 전진 명령어를 8회 사용하고 마지막 원대 복귀 명령어인 a(ction)를 호출함으로써 미션을 달성한다. 따라서 미션을 달성하는데 필요한 최소 총 호출횟수는 9회이다. 이를 퍼즐 솔루션에 적용하면 다음과 같다.

$$S_0 = fffffffffa$$

코드길이 측면에서 본 예제의 이상적인 솔루션은 다음과 같다.

$$S_{ideal} = 8fa$$

최소 코드의 길이는 3, 총 호출횟수 9회, 무효명령어수는 0이 되어 코드구조 측면에서나 성능측면에서 가장 우수하다.

하지만 반복되는 기능을 사용하면 S_0의 코드길이 9보다 적은 퍼즐 솔루션으로 다음과 같은 솔루션들이 제시될 수 있다.

$$S_1 = G_1 G_1 a, \quad G_1 = ffff$$
$$S_2 = G_2 G_2 G_2 G_2 a, \; G_2 = ff$$

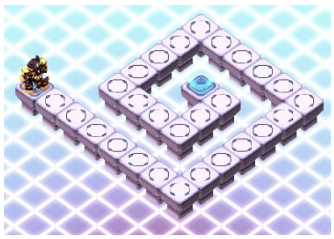
이들의 코드길이는 7이므로 코드길이 측면에서는 S_0보다는 우수하나, 여분의 함수 호출 횟수가 필요하게 된다.

만일 무한반복을 사용하게 되면 아래 S_3와 같이 코드길이를 최적의 솔루션과 동일한 3번으로 제한할 수 있다. S_3에서는 도착지까지 도착해서 원대 복귀하기 전까지 한 칸 전진 후 원대 복귀 명령을 무한히 반복한다. 실제 원대 복귀 명령은 도착 타일에서만 적용되기 때문에 일반 타일위에서는 무효화 명령이 된다.

$$S_3 = faS_3 = \infty(fa)$$

총 호출수는 16회이며 이중 7번은 NOP 동작을 수행하였다. 이는 전체 호출수 대비 44%의 무효명령어 수행 비율로서 성능측면에서 매우 비효율적이라 할 수 있다. 그럼에도 불구하고 코드길이를 획기적으로 줄일 수 있기 때문에 코드구조화 측면에서 보이지 않는 반복패턴을 잘 활용한 예제로 볼 수 있다.

[Fig. 4]와 같이 나선형으로 타일이 배치된 형태의 퍼즐에 대한 코드 품질을 평가해본다. 총 명령어 호출 수는 35회로 다음과 같은 명령어 시퀀스를 가질 수 있다.



[Fig. 4] Case Study 2: Looping in a Spiral

$$S_0 = fffffffflfffffflfffffflfffffflfffflfflfa$$
$$S_1 = 7fl6fl5fl4fl3fl2flfa$$

S_1의 경우, 반복문을 사용하면 코드길이가 35에서 20으로 향상됨을 알 수 있다. 하지만 추상화 관점에서 본다면 막다른 골목이 나올 때까지 전진 후 좌회전을 6번 반복 시행하는 방식으로 추상화하여 반복 사용하게 된다면 패턴 인식 측면에서 매우 유용한 접근법이라 할 수 있다. 이때 다음과 같이 의도적으로 무효명령어를 적극적으로 사용함으로써 추상화 레벨을 높이는 방식을 고려할 수 있다.

$$S_2 = \infty(fffffffla)$$

S_2에서는 무한반복을 사용함으로써 코드 길이는 10으로 줄어 S_1 코드길이의 절반으로 줄일 수 있다. 반복문을 사용하는 S_3에서는 코드길이를 5까지 줄일 수 있게 된다.

$$S_3 = \infty(7fla)$$

반면 S_2와 S_3의 호출수는 63회로 성능이 절반정도 저하되며 이중 28회는 NOP 명령을 실행하게 된다.

위 사례 연구를 통해 본 코드길이와 코드 성능 간 상충하는 예를 많이 볼 수 있다. 특히 본 연구에서 제시한 코드길이 비용 모델은 무한반복과 같이 함수의 추상화 수준을 높이며, 유사한 반복 패턴을 찾는데 중점을 둔 컴퓨팅 사고력 함양에 매우 적합한 요소라 평가할 수 있다. 하지만 무효 명

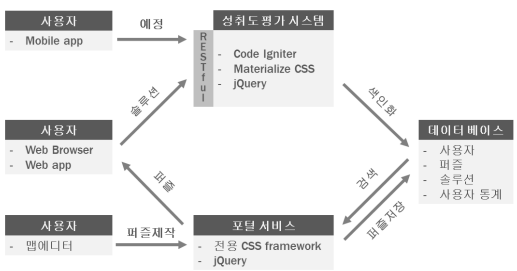
령어를 무분별한 남용을 막을 수 있도록 평가 기준을 보정하는 지표를 포함하여 균형감 있는 코드를 작성할 수 있도록 성취도 평가를 하는 것이 바람직할 것이다.

5. 성취도 시스템 설계 및 구현

본 장에서는 위에서 제시한 비용 모델을 기반으로 성취도 시스템의 설계 및 구현을 소개한다.

5.1 코딩퍼즐 시스템 개요

본 장에서는 현재 운영중인 코딩퍼즐 플랫폼의 내부 시스템 구조를 소개한다. 시제품 형태로 구현된 코딩퍼즐 플랫폼의 로컬 서버 시스템은 [Fig. 5]와 같이 구성되어 있다.



[Fig. 5] System Architecture of Coding Puzzle Coding Assessment Prototype System

사용자는 웹앱 형태로 포탈을 통해 접근하는 웹기반 클라이언트 시스템, 네이티브 모바일 앱으로 구현된 모바일 클라이언트 시스템을 통해 접근한다. 웹서버에 구축된 서버 환경은 사용자 솔루션 정보, 퍼즐 정보 등을 저장하고 검색할 수 있는 DBMS 시스템과 웹을 통해 프론트-엔드에 접근하여 퍼즐 관련 일반 서비스를 담당하는 포탈 서비스, 사용자의 퍼즐 솔루션 정보를 받아 사용자의 성취도를 평가하는 성취도 평가 시스템 등 3개의 독립적인 모듈로 구성되어 있다.

웹기반 클라이언트 시스템은 포탈의 서비스 형태로 일반 학습자들이 접근하는 애플리케이션이며 single page 웹 페이지 형식으로 구현되어 있다. 웹 페이지는 unity3D 게임 엔진으로 개발하였다.

네이티브 모바일 애플리케이션은 웹페이지용 웹빌드에서 사용한 unity3D 게임 엔진내 구현된 C# 코드를 기반으로 하고 있으며 안드로이드향 모바일 실행이 가능한 APK 파일로 빌드되어 배포되고 있다. 네이티브 모바일 애플리케이션에서는 학습자가 사용할 수 있는 모든 퍼즐 스테이지 정보를 네이티브 전용 UI/UX를 통해 제공하고 있다.

서버는 WAMP 스택인 Bitnami 패키지 v5.6.24-0 (MySQL 버전 5.6), PHP 프레임워크로는 CodeIgniter 3.1.4 버전을 사용하여 개발하였다.

5.2 성취도 평가 시스템 구현

성취도 평가 시스템은 학습자단에 위치하여 퍼즐 솔루션 정보를 추출하여 서버로 전달하는 클라이언트단과 솔루션 정보를 바탕으로 실제 평가를 진행하고 결과를 반환하는 서버단으로 나뉜다.

클라이언트단에서는 미션 완수 후에 사용된 명령어 시퀀스들을 JSON 포맷으로 변환하는 작업을 수행한다. [Fig. 6]에서는 이러한 샘플 JSON 표현법을 예시하고 있다. 퍼즐 솔루션은 퍼즐을 구분할 수 있는 UUID형태의 puzzle-id 키와 명령어 시퀀스를 담고 있는 솔루션 키가 있으며, 솔루션에는 반드시 하나의 주 명령어 시퀀스인 main 키에 사용된 명령어들을 문자열 형태의 원소로 갖는 배열이 있다. 이때 복합명령어인 경우, main 키와 형제관계를 갖는 키가 생성되어 해당 명령어 키 값에 배열형태의 명령어 시퀀스가 추가된다.

서버쪽 성취도 평가 시스템은 서버내 웹서비스 API 형태로 제공된다. PHP 프레임워크에서 제공하는 세션하에 미션이 완수된 퍼즐 솔루션을 JSON 형태로 클라이언트에서 입력받는다.

JSON 형식으로 입력된 솔루션 파일과 연관 퍼즐을 식별할 수 있는 id를 활용하여 모의 시뮬레이터를 구동하여 미션 완료 여부를 평가하고 완료시에는 비용 계산 모델을 바탕으로 계산 결과를 산

출하고 해당 퍼즐 id를 기본키로 갖는 데이터베이스 테이블에 접근하여 가장 적은 비용을 갖는 퍼즐 솔루션의 비용으로 나누어 웹 API 요청에 응답 결과를 반환한다.

```
{
    "puzzle-id" : "e4e35770-bb3a-4e5d-8ac2-14674ba1d15b",
    "solution" : {
        "main" : [
            "action",
            "move_forward",
            "G1",
            "G1"
        ],
        "G1" : [
            "turn_left",
            "move_forward",
            "move_forward",
            "action",
            "G1"
        ]
    }
}
```

[Fig. 6] Sample Puzzle Solution JSON Representation

6. 결론 및 향후 과제

본 논문에서는 퍼즐 게임 환경에서 코딩에 대한 평가 방법을 연구하였다. 코딩에 대한 평가는 단순한 모델인 무한 반복의 예에서 본 바와 같이 복잡한 요소들이 내재되어 있으며, 이것들에 대한 객관적인 기준을 마련하는 것은 향후 중요한 연구과제이다. 또한 조건문, 재귀문, 이벤트문, 병렬 실행문 등 고도화된 코딩 문법 요소들에 대한 비용 모델링도 향후 중요한 연구과제이다.

본 연구에서는 현재 운영 중인 코딩퍼즐 웹사이트에서 제공하는 기능을 단순화시켜 표현함으로써 코딩에 대한 정량적인 평가 기준을 제시하였다.

본 연구에서 제안한 성취도 평가 모델은 정량화되어 비교 가능한 평가 모델이기 때문에 Deep Learning에 적용할 수 있을 것이다. 또한 본 모델이 향후 Deep Learning 기반의 자동화된 성취도 평가 모델 연구의 초석이 되기를 기대한다.

ACKNOWLEDGEMENT

This work was supported by 2015 Hongik University Research Fund.

REFERENCES

[1] J. M. Wing, “Computational Thinking”, Communications of the ACM, Vol. 49, No. 3, pp. 33-35, 2006.

[2] Kodable, Retrieved from https://www.kordable.com, 2018.

[3] M. S. Horn, R. J. K. Jacob, “Designing tangible programming languages for classroom use”, Proc. of the 1st International Conf. on Tangible and Embedded Interaction, pp. 159-162, 2007.

[4] D. Yarosklavski, “Lightbot”, Computer Game, Armor Games, 2008.

[5] J. Tessler, B. Beth, C. Lin, “Using Cargo-Bot to provide contextualized learning of recursion”, Proc. of 9th Annual International ACM Conf. on International Computing Education Research, pp. 161-168, San Diego, Aug. 2013.

[6] J. Moreno-Leon, G. Robles, “Dr. Scratch: a Web tool to automatically evaluate scratch projects”, Proc. of the Workshop in Primary and Secondary Computing Education, pp. 132-133, 2015.

[7] S. Kim, S. Song, S. M. Lim, J. Kim, “Development of Entry Automatic Evaluation System for Assessment of Computational Thinking”, Proc. of Korean Association of Computer Education, Vol. 21, No. 1 pp. 53-56, 2017.

[8] J. Hollingsworth, “Automatic graders for programming classes”, Communications of the ACM, Vol. 3, No. 10, pp. 528-529, 1960.

[9] C. Douce, D. Livingstone, J. Orwell, “Automatic test-based assessment of programming: A review”, J. on Educational Resources in Computing, Vol. 5, No. 3,

Article 4, 2005.
[10] J. Mayrand, C. Leblanc, E. M. Merlo, "Experiment on the automatic detection of function clones in a software system using metrics", Proc. of International Conf. on Software Maintenance, Monterey, CA., pp. 244-253, 1996.
[11] F.A. Fontana, P. Braione, M. Zanoni, "Automatic detection of bad smells in code: An experimental assessment", J. of Object Technology, Vol. 11, No. 2, Article 5, 2012.
[11] R. Baggen, J. P. Correia, K. Schil, J. Visser, "Standardized code quality benchmarking for improving software maintainability", Software Qual. J, Vol. 1, No. 20, pp. 287-307, 2012.
[12] K. Park, S. H. Cho, B. Seo, "An Observation-based Movement Control for Educational Coding Robots", J. of Korea Game Society, Vol. 16, No. 6, pp. 131-141, 2016.

서 범 주 (Beomjoo Seo)

2001 LG전자 DTV연구소 주임연구원
2012 싱가폴국립대 Senior Research Fellow
2013- 홍익대학교 게임학부 조교수
관심분야 : 교육용 게임 개발, 가상현실, 분산 멀티미디어 데이터베이스 시스템 설계

조 성 현 (Cho, Sung Hyun)

1978 서울대학교 계산통계학과 이학사
1980 서울대학교 계산통계학과 이학석사
1995 UCLA 컴퓨터과학과 이학박사
1996- 홍익대학교 게임학부 교수
관심분야 : 게임프로그래밍, 게임인공지능