Journal of The Korean Association
of Information Education Vol. 20, No. 3,
June 2016, pp. 293-302

http://dx.doi.org/10.14352/jkaie.2016.20.3.293

블록기반 프로그래밍 코드의 수준 및 취약수준 측정방안

손원성

경인교육대학교 컴퓨터교육과

요 약

최근 SW 교육의 관심이 증대되고 프로그래밍 교육이 대학 학부교육의 주요한 부분으로 인식되고 있다. 특히 프로그래밍 입문 도구로서 블록 기반 프로그래밍 도구가 널리 사용되고 있으며 프로그래밍 입문자에게 기존 프로그래밍 언어와 비교하여 매우 다양한 장점들을 제공하고 있다. 한편 블록기반 프로그래밍 도구로 작성된 코드가 스크립트 언어일 경우 스크립트의 품질과 수준을 정교하게 측정하기 위해서는 상당한 노력을 기울어야 한다. 따라서 블록기반 프로그래밍 코드의 품질측정과 관련된 대부분의 연구는 단순히 프로그래밍 개념과 연관된 블록의 사용개수를 정량화하여 스크립트의 수준을 평가하고 있다. 그 결과 기존연구의 기법으로는 취약수준을 측정하거나 스크립트에 명시되지 않는 프로그래밍 개념에 대한 평가가 어렵다. 본 연구는 블록기반 프로그래밍 스크립트의 품질측정 및 취약수준 분석이 가능한 프레임워크를 제안한다. 프레임워크에서는 블록기반 프로그래밍 언어들이 내포한 다양한 프로그래밍 개념을 구조화한 평가 매트릭스를 구축하고 동 지표를 기반으로 스크립트의 품질 측정과 항목별 취약점 개선에 따른 수준향상 모델을 제안함으로써 개인별 수준진단 및 향후 개선 가능한 목표수준을 예측할 수 있도록 하였다.

키워드 : 블록기반 프로그래밍 도구, 프로그래밍 수준평가, 코드 취약수준 측정, 소프트웨어 교육, 스크래치

A Method for Measuring and Evaluating for Block-based Programming Code

Wonsung Sohn

Dept. of Computer Education, Gyeongin National University of Education

ABSTRACT

It is the latest fashion of interesting with software education in public school environment and also consider as high priority issue of curriculum for college freshman with programming 101 courses. The block-based programming tool is used widely for the beginner and provides several positive features compare than text-based programming language tools. To measure quality of programming code elaborately which is based script language, it is need to very tough manual process. As a result the previously research related with evaluation of block-based script code has been focused very simple methods in which normalize the number of blocks used which is related with programming concept. In such cases in this, it is difficult to measure structural vulnerability of script code and implicit programming concept which does not expose. In this research, the framework is proposed which enable to measure and evaluate quality of code script of block-based programming tools and also provides method

논문투고 : 2016-06-13
논문심사 : 2016-06-13
심사완료 : 2016-06-22

to find of vulnerability of script code. In this framework, the quality metrics is constructed to structuralize implicit programming concept and then developed the quality measure and vulnerability model of script to improve level of programming. Consequently, the proposed methods enable to check of level of programming and predict the heuristic target level.

Keywords : Block-based Programming, Scratch, Code Quality, SW Education, Vulnerability of Code

1. 서론

최근 SW 교육의 열풍으로 국내 초중등 현장에서 프로그래밍 교육에 대한 관심이 높아지고 있으며 대학에서도 학부 신입생 또는 비 IT 전공 학생 대상의 기초 프로그래밍 교육이 확산되고 있다[6].

한편 비전공 학부생 대상의 기초 프로그래밍 강좌에서는 블록 기반의 프로그래밍 도구[23]들이 효과적으로 활용되고 있다[9][10][19]. Alice[10]는 미국 학부 학생들의 프로그래밍 입문 강좌에 사용되었으며 최근에는 Scratch[14], Snap![8], Blockly[7]과 같은 블록 기반 프로그래밍 도구들이 대중적으로 사용되고 있다. 또한 블록 기반 프로그래밍 인터페이스는 앱 개발[20]과 같은 보다 복잡도가 높은 분야에 활용되는 등 프로그래밍의 대중화에 크게 기여하고 있다[1][3][5][13][16].

블록기반 프로그래밍 도구는 'tinkering'[14][20] 또는 퍼즐 메타포[23]에 기반 한 시각적 단서(visual cues)를 제공하여 문법을 학습하지 않고서도 명령 스크립트를 쉽게 작성할 수 있도록 한다. 이러한 시각적 단서들은 블록의 형태가 동일한 경우에만 조합이 가능함으로 문법적 에러가 원천적으로 발생하지 않는다. 또한 컬러를 이용한 블록들의 기능적 분류, 블록의 내포 관계를 시각화한 스크립트 범위 및 논리관계의 명시화를 통하여 텍스트 기반 언어보다 월등히 높은 재인(recognition) 과정 및 어포던스(affordance) 효과를 제공함으로써 누구나 쉽게 스크립트를 작성할 수 있는 환경을 제공하고 있다[18].

그러나 블록기반의 프로그래밍 도구들의 장점들은 오히려 기존 프로그래밍 언어 환경에서 일반적으로 통용되었던 소스코드 품질측정 기법[4]을 적용하기 어려운 다음과 같은 문제점들을 내포하고 있다.

첫째 블록기반 프로그래밍 도구는 캐릭터와 스토리 구현과 관련된 저작개념과 프로그래밍 개념을 동시에 내포하기에 정교한 스크립트의 레벨 또는 품질을 평가하는 작업은 상당한 수준의 인지적 부하를 수반한다[2]. 둘째 'tinkering' 인터페이스는 누구나 쉽게 저작활동을 가능케 하지만 문제 해결 전략 차원에서는 상향식(bottom-up) 기법[10]이다. 이 경우 블록 인스턴스를 모듈화하지 못한다면 최하위 단위의 블록 인스턴스만이 존재하는 환경이 초래될 수 있다.

그 결과 블록기반 프로그래밍의 소스코드(스크립트)의 평가와 관련된 기존 연구들은 주로 블록들이 내포한 프로그래밍적 요소 및 특성들을 선정하고 해당 요소의 블록이 사용된 회수를 취합하여 이것을 정규화하는 방법을 제안하고 있다. 전통적인 프로그래밍 소스 코드 품질 측정의 목표[4][21][22]는 코드의 복잡도를 최소화하여 보수성을 높이는 것에 있으나 블록 기반의 프로그래밍 도구는 앞서 언급한 특성으로 인하여 기존의 방식을 적용하기가 매우 어렵다. 블록기반의 스크립트 품질을 평가하는 가장 편리한 방법은 블록의 사용횟수를 스크립트 품질측정의 기준으로 이용하는 것이다.

따라서 본 연구는 구현된 블록 개수와 같은 단순한 정량적 지표 보다는 블록의 중요도와 구현 난이도 그리고 코드 단순화 등과 같은 개념을 반영하여 기존 연구의 문제점을 개선한 소스코드(스크립트) 품질 측정방안을 제안하고자 한다.

본 연구는 블록기반 프로그래밍 스크립트의 품질측정 및 취약수준 분석이 가능한 프레임워크를 제안한다. 프레임워크에서는 블록기반 프로그래밍 언어들이 내포한 다양한 프로그래밍 개념을 구조화한 평가지표(quality metric)[21][22]를 구축하고 동 지표를 기반으로 스크립트의 품질 측정과 항목별 취약점 개선에 따른 수준향상 모델을 제안함으로써 개인별 수준진단 및 향후 개선 가능한 목표수준을 예측할 수 있도록 하였다. 또한 제안모델의 평가지표 항목은 AHP기법[24]을 적용

하여 평가항목의 중요도에 따른 가중치를 산출하고, 프로그래밍 수준 및 취약수준의 산출에 반영함으로써 평가의 정확성을 높이고자 하였다. 마지막으로 본 연구의 제안기법을 지난 7년 동안 교육대학교 스크래치 프로그래밍 수업에서 도출된 스크립트에 적용해봄으로써 기존 평가 모델 및 연구와의 차별성을 확인하고 제안 모델의 효용성과 우수성을 검증하였다.

2. 관련연구

전통적인 프로그래밍 코드의 품질 측정 및 수준 평가는 매트릭스 측정기법을 통하여 수행되며 소스코드의 간결화 및 재사용성의 극대화 그리고 복잡도 감소를 목표로 한다. 이와 관련하여 ISO 9126표준[11]에서는 소스코드 품질평가를 위한 매트릭스 기준을 포함하고 있다.

일반적으로 블록기반 프로그래밍 도구는 에러 없는 코드실행 환경을 제공하기 위하여 컴파일러 보다는 스크립트 언어 기반으로 제작된다. 따라서 서론에서 언급한 바와 같이 스크립트를 실행하면서 제작된 블록의 종류와 개수 등을 기준으로 품질 및 수준을 평가하는 방법을 주로 사용하며 이와 관련된 다양한 연구가 수행된 바 있다. 블록기반 프로그래밍 소스코드 평가와 관련 연구는 크게 매뉴얼과 자동화 기법으로 구분된다.

스크래치 프로그래밍 가이드[1]에서는 스크립트 평가 루브릭을 제안하고 있으며 디자인, 스크립트, 인터페이스와 관련된 항목을 포함하고 있다. 제안 스크래치 루브릭이 효과적이려면 수행 수준에 따라 보다 세분화된 척도가 요구된다.

블록 스크립트가 내포한 프로그래밍 개념을 도출[15]하고 이와 관련한 블록의 작성 개수를 수준평가의 척도로 간주하는 다양한 연구가 선행되었다. 스크래치 및 기타 블록 프로그래밍 도구를 이용한 아동용 온라인 저작도구 프로그램[5], 8-18세 대상의 프로그래밍 교육[13], 학부 여름학기의 기초 프로그래밍 과목[3], 학부 프로그래밍 기초과목[20] 등의 사례에서는 프로그래밍 개념과 관련된 블록의 작성 개수를 정량화하여 프로그래밍 결과물의 품질을 평가하는 기법을 널리 사용하고 있다. 그러나 세분화된 개인별 프로그래밍 수준을 평가하기 위해서는 블록의 개수와 더불어 각 블록의 중요도를 고려하여야 한다.

Michal Armoni와 Moti Ben-Ari[16][17]는 스크래치가 내포한 컴퓨터과학 개념을 8개 항목으로 분석하고 각 영역별 개념 이해도를 평가하기 위한 테스트 문항과 정성평가를 위한 인터뷰 항목을 개발한 바 있다. 제안 연구의 평가도구는 지필 평가 및 프로젝트 모델을 포함하기 때문에 소규모 강의 환경에 적절하다.

한편 블록 스크립트 분석을 자동화하여 결과의 수준을 평가하기 위한 연구[2][12]에서는 추상화, 제어문, 데이터, 상호작용 등의 프로그래밍 개념을 자동으로 정량화하는 기법을 개발하였다. 제안 기법은 python 플러그인을 이용하여 스크래치 내부의 squeak 코드에서 사용된 블록의 개수를 정량화하여 총 점수를 산정하게 된다. 그러나 자동화 기법에서 판별 가능한 프로그래밍 개념의 수준은 매우 낮은 편이며, 블록에 명시된 개념 이외의 알고리즘 및 문제 해결 등과 같은 비명시적 프로그래밍 개념의 자동화는 불가능하다.

3. 블록기반 프로그래밍 코드의 수준 측정 및 취약점 개선을 위한 프레임워크

3.1 수준평가 매트릭스 정의

<Table 1> Quality metrics of Programming Concept

항목	평가기준	메트릭스 개수
Command	-스크래치의 기본 명령문 -스택기반 블록조합 원리	7
Parameter	-Value를 리턴하는 argument의 원리	6
Variable	-Value, reference, scope의 원리	6
Trigger	-Event 생성 및 제어 원리	7
Conditional Statement	-조건문을 통한 분기제어 원리	7
Iteration	-반복문을 통한 구문의 반복제어 원리	7
Data Type	-Boolean, number, string의 제어	5
Input & Output	-키보드, 마우스 등을 통한 데이터 입출력 및 상호작용 제어원리	6
Data Structure	-List를 이용한 데이터 제어 원리	4
Concurrency	-Event handler를 이용한 multi-threading 생성 및 제어	5
Procedures	-Make a Block 기능을 이용한 subroutine의 생성, 함수의 재사용, Parameter 제어	5
		65

본 연구에서는 블록기반 프로그래밍 코드수준을 측정하기 위한 평가지표를 3가지 그룹으로 구분하였으며 소프트웨어 품질평가를 위한 ISO 9126 표준[11]의 절차를 참고 및 응용하였다. 먼저 스크래치의 블록을 통하여 명시적으로 인식할 수 있는 프로그래밍 관련의 11가지 항목들을 <Table 1>과 같이 도출하였다.

기본 프로그래밍 개념과 더불어 블록에 명시되지 않은 고급 프로그래밍 개념을 평가하기 위한 복합개념 평가항목을 정의하였으며 그 내용은 <Table 2>와 같다.

<Table 2> Quality metrics of Advanced Concept

항목	평가기준	메트릭스 개수
Algorithms	-문제를 개선하기 위한 전략의 설계 -복수개의 결과를 생성 -연산명령의 종결을 보장	6
Divide & Conquer	-문제를 작은 인스턴스로 나누어 top-down 또는 buttom-up 방식으로 해결해가는 능력	6
Interaction Design	-Dynamic data input, sprite 속성 정보를 이용한 다양한 상호작용의 설계 및 구현능력	7
		19

마지막으로 프로그래밍과 저작개념을 모두 포함한 스크래치와 같은 도구들을 고려한 콘텐츠 요소를 평가하기 위한 항목을 도출하였으며 그 내용은 다음과 <Table 3>과 같다.

<Table 3> Quality metrics of Contents Evaluation

항목	평가기준	메트릭스 개수
창의성	-구현 가능한 콘텐츠 스토리의 완성도 -풍부한 스토리 전개 및 복수의 시나리오 제공 등	5
심미성	-멀티미디어 및 스크래치 오브젝트의 커스터마이징을 통한 심미적 효과의 존재여부 등	5
기능성	-콘텐츠의 기능적 요인 존재여부 등	5
완성도	-스토리 전개의 다양한 패스 존재여부 -사용자 선택이 가능한 옵션의 존재여부 -스토리 구현의 완성도 등	5
		20

한편 블록기반 프로그래밍 코드(스크립트)의 평가 지표에 대한 적정성을 확보하고자 컴퓨터 IT 그룹(21명)과 정보/컴퓨터교육 그룹(20명)을 대상으로 세부 평가메트릭(104개)의 적정성에 대한 설문을 실시하였다.

제안 연구의 스크래치 프로그래밍 수준 평가지표로 구성된 항목들에 대한 신뢰도 분석은 내적일관성 방법을 적용하였으며, SPSS 19를 이용하여 평가지표의 적정성에 대한 신뢰도 분석을 수행한 결과 크론바흐 알파값은 0.892의 신뢰도를 보였다.

3.2 프로그래밍 코드의 수준 측정모델 설계

3.2.1 프로그래밍 코드 수준모델 정의

본 연구에서는 평가된 전체 평가항목의 평가 값에 가중치를 반영하여 산출한 점수를 평가지표의 만점에 대한 백분율로 환산하고 그 결과에 따라 프로그래밍 코드(스크립트)의 수준을 판단하는 방법을 구현하였다. 본 연구에서 적용하는 평가기준은 코드수준 5단계를 기준(5점 척도)으로 평가점수를 부여(1단계: 1점~5단계: 5점)한다. 이것을 만점(5점)에 대한 백분율로 환산하면 1점은 20%, 2점은 40%, 3점은 60%, 4점은 80%, 5점은 100%가 되며, 이를 기준으로 코드수준의 구간을 <Table 4>와 같이 정의하였다.

<Table 4> Five levels of Code Quality

단계	점수	수준 설명	수준 정의
1수준	20%미만	취약	블록을 조합하고 간단한 상호작용을 저작하는 수준
2수준	20%이상~40%미만	미흡	데이터의 가공 및 처리, 상호작용 및 이벤트구현이 가능한 수준
3수준	40%이상~60%미만	보통	자료구조 사용 및 프로시저 설계가 가능한 정도
4수준	60%이상~80%미만	우수	알고리즘과 문제해결전략을 이해하고 고급 상호작용 설계가 가능한 정도
5수준	80%이상~100%미만	매우 우수	기본 개념 및 복합개념에 대한 이해 및 구현수준이 매우 우수한 정도

3.2.2 평가지표 가중치 산출

본 연구에서는 평가항목에 대한 가중치 산정방식으로 AHP 기법을 적용하며, 국내 정보교육 및 정보/컴퓨터/IT 분야의 전문가를 대상으로 중요도에 대한 설문을 실시하고, 설문 결과를 기초로 상대적 가중치를 산정하였다. 가중치 산정을 위한 설문은 평가지표의 적정성 설문과 함께 진행되었다. 블록기반 프로그래밍 수준 판별을 위한 3가지 그룹에 대한 상대적 가중치는 <Table 5>와 같다.

각 계층별로 산출된 평가영역 가중치와 평가분야 가중치 값을 곱하여 평가분야의 가중치의 합이 1이 되도록 정규화한 값이 측정 모델에 적용될 평가분야의 가중치 값이 된다.

<Table 5> Quality metrics with Weighted Value

평가영역		평가분야		평가영역이
영역	가중치 (A)	세부항목	가중치 (B)	가중된 가중치 (A x B)
기본 개념	0.523	Command	0.025	0.013
		Parameter	0.014	0.007
		Variable	0.097	0.051
		Trigger	0.079	0.041
		Conditional Statement	0.172	0.090
		Iteration	0.168	0.088
		Data Type	0.054	0.028
		Input & Output	0.072	0.038
		Data Structure	0.054	0.053
		Concurrency	0.105	0.055
		Procedures	0.113	0.059
복합 개념	0.538	Algorithms	0.392	0.130
		Divide & Conquer	0.386	0.128
		Interaction Design	0.222	0.073
콘텐츠 영역	0.146	창의성	0.303	0.044
		심미성	0.199	0.029
		기능성	0.197	0.029
		완성도	0.301	0.044
영역 합계	1.000			1.000

3.3 블록기반 프로그래밍 코드수준 측정방안

블록 스크립트의 평가자는 블록에 명시된 프로그래밍 기본 개념과 알고리즘 및 문제해결 전략에 관한 복합개념 그리고 콘텐츠 요소와 관련된 수준평가항목(총 104개 항목)을 다음 <Table 6>의 평가기준에 따라 평가를 진행하며, 전체 항목에 대한 평가를 완료한 후 다음 절차에 따라 스크립트의 수준을 산출한다.

<Table 6> Evaluation Criteria of Code Quality

평가 점수	1점	2점	3점	4점	5점
점수 기준	기본개념의 이해와 구현수준이 낮다	기본개념을 이해하고 있다	기본개념 및 구현수준이 적절하다	이해도 및 구현수준이 우수하다	구현수준이 매우 우수하다
	미흡	보통	양호	우수	매우우수

가. 개별 평가항목에 대한 평가

평가자는 스크래치 결과물을 이용하여 <Table 6>의 평가기준에 따라 독립적으로 평가를 실시하며, 각 항목에 평가 값을 부여한다.

E_{ij} : 평가분야i의 평가항목j의 평가 값 $(ij : 1 \cdots\cdots n)$

나. 평가점수 및 스크립트 수준 산출

수준평가분야에 소속된 항목의 평가 합을 구하고 <Table 5>를 참조하여 각 분야에 해당하는 가중치를 평가항목 평가 합에 적용하여 평가분야의 가중 평가 값을 산출하며, 식으로 표현하면 식 3-1과 같다.

$$E_i = \sum_{j=1}^{n} E_{ij} \times w_i \quad (j = 1,2,\cdots,n) \quad \cdots\cdots\cdots \quad (3-1)$$

$\sum_{j=1}^{n} E_{ij}$: i번째 평가분야 E_i에 속한 평가항목들의 합

E_i : i번째 평가분야의 가중치가 반영된 '스크립트 수준 평가점수'

w_i : 평가분야 E_i의 가중치

식 3-1에 의해 산출된 프로그래밍 수준 평가점수를 해당 평가분야의 가중치가 반영된 만점 값(이하, 가중만점 값)으로 나누어 백분율로 환산하고 각 평가분야의 스크립트 수준을 산출한다. 산출된 수준에 따라 <Table 4>를 참고하여 해당하는 프로그래밍 코드의 수준단계를 부여한다.

$$PE_i = \frac{E_i}{E_i\text{의 가중 만점 값}} \times 100 \quad \cdots\cdots\cdots\cdots \quad (3-2)$$

E_i : 평가분야 E_i의 수준 점수, $i : 1,2,\cdots n$,

PE_i : 평가분야 E_i의 수준 (백분율)

다. 측정집단의 프로그래밍 수준 산출

측정집단의 프로그래밍 수준을 산출하기 위하여 전체 프로그래밍 수준 점수의 합을 구한 다음, 이를 전체 평가분야의 만점 합으로 나누어 측정집단의 프로그래밍 수준 전체의 백분율로 환산하여 집단의 프로그래밍 수준을 산출한다. 그 결과를 <Table 4>를 참고하여 전체 집단의 프로그래밍 수준단계를 도출한다.

$$TE = \sum_{i=1}^{n} E_i \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (3-3)$$

$$ML = \frac{TE\text{의 프로그래밍 수준 점수}}{TE\text{의 가중만점 합}} \times 100$$

TE : 전체 평가분야 E_i의 수준 평가점수,

i : $1,2,\cdots n$,

ML : 집단의 프로그래밍 수준 (백분율)

3.4 취약점 개선을 통한 스크래치 프로그래밍 수준 향상 방안

본 연구에서는 블록 스크립트의 수준측정을 통하여 프로그래밍 취약점을 도출하고 이것의 개선할 수 있는 방안을 제시한다.

먼저 블록 스크립트에 수준 측정 및 평가를 통하여 개인 및 집단의 프로그래밍 및 개념 이해수준을 파악하고, 이어서 개인이 획득한 프로그래밍 수준이 집단의 프로그래밍 목표 수준보다 낮은 평가를 받은 항목을 취약 항목으로 분류한다.

다음으로는, 식별된 취약 항목에 대해서는 집단의 목표수준과 항목의 평가 값의 차이를 계산하여 프로그래밍의 취약수준을 제시한다. 도출된 취약수준은 취약 항목을 목표 수준으로 개선하였을 경우 예측되는 수준향상의 정도를 의미한다. 따라서 프로그래밍 교육을 수행한 집단은 프로그래밍 수준판별과 동시에 취약점 개선에 따른 수준향상 정도를 예측하고, 이를 이용한 취약순위 산출 및 취약점 개선과 같은 다양한 개선방안 수립이 가능하다.

가. 프로그래밍 목표 수준 결정

본 연구에서는 집단의 프로그램 수준측정 결과가 4단계에 미달하는 경우 것으로 평가된 집단은 도출된 수준 보다 한 단계 높은 단계를 기준등급으로 정의하거나 3단계를 목표 수준으로 결정한다. 예를 들면, 스크래치 프로그래밍 수준이 2단계로 평가된 집단은 목표 수준을 3단계나 4단계로 결정하는 것이 바람직하다. 또한 4단계 이상으로 측정된 집단은 목표 수준을 최고 단계인 5단계로 정의한다.

나. 개별 평가항목에 대한 취약 수준 산출

스크립트의 취약수준을 판별하기 위하여 목표 수준의 단계를 점수(1단계: 1점, 2단계: 2점, 3단계: 3점, 4단계: 4점, 5단계: 5점)로 변환하고 이 값과 평가항목의 평가값(1점~5점)과의 차이가 0보다 클 경우 취약 항목으로 결정하고, 그 결과 값을 취약수준으로 지정한다. 취약수준이 0보다 크다는 것은 평가항목 값이 목표수준에 미달하기 때문에 이것을 취약 항목으로 결정하며, 취약점수에 해당 분야의 가중치를 곱한 값이 취약수준점수가 된다. 취약수준점수 산출식은 다음 식 3-4와 같다.

$$VE_{ij} = (GL - E_{ij}) \times W_i \quad \cdots\cdots\cdots\cdots \quad (3-4)$$

$GL - E_{ij}$: 평가항목의 취약점수

VE_{ij} : 평가분야i 평가항목j의 취약수준 점수

GL : 측정 집단의 목표 수준

E_{ij} : 평가분야i 의 평가항목j 의 평가 값

W_i : 정규화된 평가분야i의 가중치($1 = \sum_{i=1}^{n} W_i$)

다. 취약 수준 개선을 통한 프로그래밍 수준향상

도출된 취약 수준은 개인 및 집단에게 기대되는 프로그래밍 목표 수준에 부족한 수준으로서 각 항목의 취약성을 목표 수준이 요구하는 사항에 맞게 개선하게 된다면 평가결과에 대하여 기대되는 수준향상의 결과가 된다. 따라서 본 연구에서는 식 3-5와 같이 산출된 평가분야 취약점수를 평가분야 목표수준 점수에 합산하여 취약점 개선 시 기대되는 프로그래밍 수준향상 점수를 산출한다.

$$SVE_i = E_i + VE_i \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (3-5)$$

E_i : 평가분야i의 프로그래밍 수준 점수

VE_i : 평가분야i의 취약수준 점수

SVE_i : 취약점 개선시 평가분야i의 프로그래밍수준 점수

4. 적용결과

본 연구에서는 블록기반 프로그래밍 수준 측정 및 취약수준 개선 방안을 제안하였다. 또한 제한 기법을 교육대학교의 스크래치 프로그래밍 교육과정에서 도출된 데이터에 적용함으로써 제안모델의 효용성을 검증하도록 한다.

4.1 적용 대상

본 연구는 2008년부터 2015년까지 교육대학교 컴퓨터교육과 학부 3학년 대상의 '저작도구' 및 '교육실기' 수업을 수강한 312명의 수강생들이 제작한 스크래치 프로젝트 결과물을 대상으로 프로그래밍 수준측정 및 취약수준을 분석하였다. 커리큘럼의 상세한 내용은 생략하도록 한다.

4.2 스크래치 프로그래밍 수준측정 및 취약수준 산출

제안 연구의 스크래치 프로그래밍 수준측정 모델을 적용하여 분석한 결과는 <Table 7>의 내용과 같다. 312명의 프로젝트 결과물의 최종 점수는 163점이며, 평가영역의 가중치를 적용한 점수 8.462를 가중 만점 값(16.158)에 대한 백분율로 환산하여 스크래치 프로그래밍 수준 54.46%를 산출하였다. <Table 7>의 내용을 살펴보면 기념개념 중 '데이터 타입'을 제외한 모든 항목이 3수준(양호)으로 평가되었으며, 특히 '프로시저' 항목은 최저 2수준(미흡)을 산출하였다. 이는 학생들이 함수 생성원리에 대한 이해도가 낮기 때문이며, 향후 커리큘럼 개발 시 고려할 내용으로 판단된다. 또한 스크래치에서 명시적으로 부가되지 않는 알고리즘(3수준) 및 문제해결 전략(2수준) 등과 관련된 개념은 강의과정에서 보다 집중적으로 다뤄져야 할 필요가 있다.

<Table 7> Measurement Results of Script Quality of Scratch Programming

평가 영역	평가분야 세부항목	가중치 (A)	평가 값 (B)	평가점수 A x B	스크립트 수준(%)	스크립트 레벨(단계)	영역별 수준 (%)	집단 수준 (%)
기본 개념	Command	0.013	12	0.156	54.67	3	54.77	54.46 (3단계)
	Parameter	0.007	10	0.070	53.45	3		
	Variable	0.051	7	0.357	51.69	3		
	Trigger	0.041	8	0.328	59.02	3		
	Conditional Statement	0.090	11	0.990	55.84	3		
	Iteration	0.088	12	1.056	59.81	3		
	Data Type	0.028	12	0.336	69.16	4		
	Input&Output	0.038	10	0.380	58.84	3		
	Data Structure	0.053	7	0.371	47.21	3		
	Concurrency	0.055	7	0.385	53.39	3		
	Procedures	0.059	5	0.295	39.35	2		
복합 개념	Algorithms	0.130	6	0.780	41.86	3	46.02	
	Divide&Conquer	0.128	6	0.768	39.11	2		
	Interaction Design	0.073	10	0.730	57.10	3		
콘텐츠 영역	창의성	0.044	10	0.440	58.98	3	59.95	
	심미성	0.029	9	0.261	57.22	3		
	기능성	0.029	11	0.319	67.75	4		
	완성도	0.044	10	0.440	55.84	3		
합계		1	163	8.462	54.46	3		

4.3 취약점 개선 시 기대되는 프로그래밍 향상 수준 측정

컴퓨터교육과 학생들의 스크래치 스크립팅 수준은 3단계로 산출되었음으로 목표 수준은 4단계로 정의하였다. 따라서 목표 수준에 미달되는 취약 항목을 도출하여 그 차이 값과 가중치를 이용하여 취약수준을 산출하였다. 전체 평가항목 104개 중 78개 항목이 목표수준 4단계에 미달되는 취약항목으로 판별되었으며, 19개 항목이 2단계이하로 평가되었으며 상세한 내용은 <Table 8>의 내용과 같다.

<Table 8>의 내용을 살펴보면 취약항목 104개에 대한 개선이 진행될 경우 최종적으로 4단계로의 프로그래밍 수준 향상을 기대할 수 있다. 따라서 본 연구자는 <Table 8>에서 산출된 결과를 통하여 2수준으로 판별된 항목 및 특히 스크래치 블록에 명시되지 않는 복합개념을 가장 최우선의 개선 항목으로 지정하여 향후 커리큘럼 및 강의 개선 시 반영할 예정이다.

5. 결론 및 향후연구

본 연구에서 블록기반 프로그래밍 수준을 도출하는 목적은 각 개인 및 집단의 프로그래밍 수준에 대한 평가 및 부족한 취약점을 파악하고 이것을 개선하고자 함이다. 또한 거시적으로는 계산적 사고력 및 문제 해결과 같은 소프트웨어 교육의 근본적 목표를 지향하고 그 효과의 구체적 당위성을 구체화 할 수 있는 방법론을 제시하고자 한다.

이를 위하여 블록기반 프로그래밍 도구에 내포된 다양한 프로그래밍 개념들을 도출하고 이에 대한 가중치를 적용하여 코드에 대한 품질을 측정하고 동시에 취약점을 찾아 개선에 반영하고자 하였다.

제안 기법은 대학 현장에서 수행된 312명의 스크래치 스크립트에 적용되어 그 효용성을 검증하였으며 향후 프로그래밍 교육에 반영할 수 있는 개선점들을 도출하였다.

향후 연구로는 보다 심도 있는 평가항목에 대한 분석이 요구되며 특히 명시적으로 나타나지 않는 잠재적 요인들을 정교하게 분석하여 본 연구에서 간과되었던 부분들을 보완하고자 한다.

<Table 8> The Expected Improvement from Measuring Vulnerability of Script

평가 영역	평가분야 세부항목	평가점수 (A)	취약수준 (B)	기대 향상수준 (A+B)	스크립트 기대 향상수준 (%)	영역별 기대 향상수준(%)	집단의 기대 향상수준(%)
기본 개념	Command	0.156	0.015	0.171	60.00	60.84	60.95 (4단계)
	Parameter	0.070	0.008	0.078	60.02		
	Variable	0.357	0.057	0.414	60.01		
	Trigger	0.328	0.005	0.333	60.00		
	Conditional Statement	0.990	0.073	1.063	60.00		
	Iteration	1.056	0.003	1.059	60.01		
	Data Type	0.336	-	0.336	69.16		
	Input&Output	0.380	0.007	0.387	60.07		
	Data Structure	0.371	0.100	0.471	60.01		
	Concurrency	0.385	0.047	0.432	60.00		
	Procedures	0.295	0.155	0.45	60.00		
복합 개념	Algorithms	0.780	0.337	1.117	60.01	60.03	
	Divide&Conquer	0.768	0.421	1.189	60.06		
	Interaction Design	0.730	0.037	0.767	60.02		
콘텐츠 영역	창의성	0.440	0.007	0.447	60.00	61.94	
	심미성	0.261	0.012	0.273	60.01		
	기능성	0.319	-	0.319	67.75		
	완성도	0.440	0.032	0.472	60.00		
합계	0	8.462	1.316	9.778	60.95	60.95	

참고문헌

[1] An Educator's Guide to Scratch Programming (2016). http://www.scratch-programming.org.

[2] Bryce Boe, Charlotte Hill, Michelle Len, Greg Dreschler, Phillip Conrad, and Diana Franklin (2013). "Hairball: lint-inspired static analysis of scratch projects." In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). ACM, New York, NY, USA, 215-220.

[3] David J. Malan and Henry H. Leitner (2007). "Scratch for budding computer scientists." *SIGCSE Bull., 39*(1), 223-227.

[4] Demuth, Birgit, Sebastian Götz, Harry M. Sneed, and Uwe Schmidt (2013). Evaluation of Students' Modeling and Programming Skills. In EduSymp@ MoDELS.

[5] Fields, Deborah A., et al. (2014). Interactive story-telling for promoting creative expression in media and coding in youth online collaboratives in Scratch. Proceedings of constructionism. 19-23.

[6] For all Software (2016). http ://sw4all.kookmi-n.ac.kr/notice/1.

[7] Fraser, N. (2013). Blockly. Google.

[8] Harvey, B. and Mönig, J. (2010). Bringing "no ceiling" to Scratch. Proc. of Constructionism 2010 (Paris, Fr.), 1-10.

[9] Hundhausen, C., Farley, S., and Brown, J. (2009). Can direct manipulation lower the barriers to computer programming and promote transfer of training?: An experimental study. *ACM Trans. Comput.-Hum. Interact., 16*(3).

[10] Ian Utting, Stephen Cooper, Michael Kölling, John Maloney, and Mitchel Resnick (2010). Alice, Greenfoot, and Scratch - A Discussion. *Trans. Comput. Educ., 10*(4), Article 17.

[11] ISO/IEC 9126-1 (2001). Information technology - Software product evaluation: Quality Characteristics and Guidelines for their use.

[12] Jesús Moreno-León and Gregorio Robles (2015). Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects. In Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE '15). ACM, New York, NY, USA, 132-133.

[13] John H. Maloney, Kylie Peppler, Yasmin Kafai, Mitchel Resnick, and Natalie Rusk (2008). Programming by choice: urban youth learning programming with scratch. In Proceedings of the 39th SIGCSE technical symposium on Computer science education (SIGCSE '08). ACM, New York, NY, USA, 367-371.

[14] JOHN MALONEY, MITCHEL RESNICK, NATALIE RUSK, BRIAN SILVERMAN, and EVELYN EASTMOND (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education, 10*(4), Article 16.

[15] Lee, Y. J. (2011). Scratch: Multimedia Programming Environment for Young Gifted Learners. *Gifted Child Today Magazine, 34*(2), 26-31.

[16] O. Meerbaum-Salant, M. Armoni, M. Ben-Ari. (2013). Learning computer science concepts with Scratch. *Computer Science Education, 23*(3), 2013, 239-264.

[17] Orni Meerbaum-Salant, Michal Armoni, and Mordechai (Moti) Ben-Ari. (2010). Learning computer science concepts with scratch. In Proceedings of the Sixth international workshop on Computing education research (ICER '10). ACM, New York, NY, USA, 69-76.

[18] Raymond Lister (2011). COMPUTING EDUCATION RESEARCH: Programming, syntax and cognitive load. *ACM Inroads, 2*(2) (June 2011), 21-22.

[19] Shitanshu Mishra, Sudeesh Balan, Sridhar Iyer, and Sahana Murthy (2014). Effect of a 2-week scratch intervention in CS1 on learners with varying prior knowledge. ITiCSE '14. ACM, NY, USA,

45-50.
[20] Slany, W. (2014). Tinkering with Pocket Code, a Scratch-like programming app for your smartphone. Proc. of Constructionism 2014 (Vienna, Aus).
[21] Washizaki, H., et al. (2003). A Metrics Suite for Measuring Reusability of Software Components. In: Proc. 9th IEEE International Software Metrics Symposium.
[22] Washizaki, Hironori, et al. (2007). A framework for measuring and evaluating program source code quality. Product-Focused Software Process Improvement. Springer Berlin Heidelberg, 284-299.
[23] Weintrop, David (2015). Minding the Gap Between Blocks-Based and Text-Based Programming. Proceedings of the 46th ACM Technical Symposium on Computer Science Education. ACM.
[24] Young-Rai Park, Yoon-Chul Choy, Won-Sung Sohn (2014). International Journal of Security and Its Applications. Vol.8 No.6. 147-160.

저자소개

손 원 성

1998 동국대학교 컴퓨터공학과 (학사)
2000 동국대학교 컴퓨터공학과 (석사)
2004 연세대학교 컴퓨터과학과 (박사)
2004~2006 Carnegie Mellon University, Post Doc.
2006~현재 경인교육대학교 컴퓨터교육과 부교수
관심분야: 컴퓨터교육, 인간과 컴퓨터 상호작용(HCI), 사용자 경험 설계(UX Design), Design Thinking
e-mail: sohnws@ginue.ac.kr