Journal of The Korean Association
of Information Education Vol. 22, No. 2,
April 2018, pp. 205-212

http://dx.doi.org/10.14352/jkaie.2018.22.2.205

예비교사들이 프로그래밍 학습 시 발생시키는 오류 데이터 분석

문외식

진주교육대학교 컴퓨터교육과

요 약

예비교사들의 소프트웨어교육 능력을 키우기 위한 방안으로 정규 교과시간에 두 종류의 프로그래밍 도구(파이썬, 스크래치)를 이용하여 프로그래밍 학습을 각각 실시하였다. 프로그래밍 학습에서 지속적으로 흥미와 성취감 및 창의성을 저해하는 요소인 각종 오류들의 종류들을 수집하고 유형별로 분석하였다. 분석된 자료들을 활용하면 향후 예비교사들이 초등학교에서 가르쳐야 할 소프트웨어교육에서 발생 가능한 오류들을 줄일 수 있도록 대처할 수 있는 능력을 키울 수 있어 최적의 학습효과를 올릴 수 있다. 본 연구에서는 평균적으로 텍스트를 입력하는 기존 형태의 언어와 불럭을 조립하는 형태의 언어 모두에서 프로그래밍 시 가장 많은 오류를 발생시키는 유형이 논리오류(37.63%)로 가장 많았다. 또한, 두 언어에서 차이점이 많이 나타나는 세부적인 오류는 문법 등의 사용미숙, 오타 등으로 인한 단순오류가 파이썬이 14.3%, 스크래치가 3.5%로 큰 차이가 있음을 알 수 있었다.

키워드 : 프로그래밍 언어, 텍스트입력 언어, 드래그앤드롭 언어, 소프트웨어 오류, 오류 유형

Analysis of error data generated by prospective teachers in programming learning

Wae-shik Moon

Dept. of Computer Education, Chinju National University of Education

ABSTRACT

As a way to improve the software education ability of the pre - service teachers, we conducted programming learning using two types of programming tools (Python and Scratch) at the regular course time. In programming learning, various types of errors, which are factors that continuously hinder interest, achievement and creativity, were collected and analyzed by type. By using the analyzed data, it is possible to improve the ability of pre-service teachers to cope with the errors that can occur in the software education to be taught in the elementary school, and to improve the learning effect. In this study, logic error (37.63%) was the most frequent type that caused the most errors in programming in both conventional language that input text and language that assembles block. In addition, the detailed errors that show a lot of differences in the two languages are the errors of Python (14.3%) and scratch (3.5%) due to insufficient use of grammar and other errors.

Keywords : Programming language, text input language, drag and drop language, software error, error type

이 논문은 2016년 진주교육대학교 교내연구비 지원으로 이루어 졌음.
논문투고 : 2018-02-14
논문심사 : 2018-02-19
심사완료 : 2018-02-23

1. 연구의 필요성 및 목적

4차산업혁명 시대의 키워드라 볼 수 있는 소프트웨어는 컴퓨터시스템의 신뢰성을 좌우하는 핵심요소이다. 지식정보사회가 급격하게 변화하게 됨에 따라 세계 각국은 소프트웨어의 중요성을 인식하여 과거의 컴퓨터 활용위주 교육을 지양하고 초·중등학교, 대학에서 소프트웨어교육을 어떻게 실시해야 하는지에 대해 매우 활발하게 연구하고 있으며 실제로 많은 국가에서 초등학교부터 프로그래밍교육을 정규교과에 편입하여 실시하고 있다[11][12].

창업이 많은 이스라엘, 에스토니아에서는 소프트웨어 원리와 문제해결 능력 향상에 초점을 두어 초등학교 저학년에서 스크래치 등의 교육용 언어를 이용하여 프로그래밍 소양교육을 먼저 배운 후 고학년인 5, 6학년에는 직접 주어진 문제를 해결하기 위한 응용 소프트웨어를 체험함으로서 흥미 위주의 소프트웨어 학습을 경계하고 창의성과 문제해결력을 향상시키는데 주력하고 있다[4][5][13]. 우리나라에서도 2018년부터 초등학교에서 소프트웨어교육을 정규교과에 편입하였다. 본 연구에서는 미래 초등학생들을 위해 체계적이고 효율적인 소프트웨어 교육시스템을 만들게 되는 주체인 예비교사들을 상대로 정보교양 교과시간에 텍스트 입력을 기본으로 하는 기존 모형의 언어인 파이썬과 불럭조립형인 스크래치를 각각 소양 학습시킨 후 담당교수가 제시한 초등학교 일부 교과내용을 기초로 제시한 프로젝트를 해결하는 방법으로 프로그래밍하는 과정에서 경험적으로 발생하는 오류들을 조사하고 분석하여 예비교사들이 프로그래밍 시 사전에 발생 가능한 오류들을 예측하는 능력과 문제 해결력을 키워 소프트웨어 학습에 대한 지속적인 흥미 및 창의성을 유발시켜서 최적 소프트웨어 학습과 성취도 향상에 도움을 주고자 한다.

2. 관련연구

2.1 소프트웨어 오류

소프트웨어 오류란 소프트웨어를 개발하기 위한 전체 단계에서 작성자의 실수에 의해 만들어진 결함과 만들어진 프로그램이 실행 시 사용자가 요구하는 요구명세서대로 작동하지 않는 모든 현상을 말한다. 이는, 소프트웨어 결함을 발생시키는 부주의한 행위를 말한다. 소프트웨어 오류는 개발과정에서 입력된 설계오류, 프로그래밍 오류가 중심이 되며 테스트 등의 일련의 작업 과정에서 발견된 오류를 수정함으로서 실행시간이 경과하면 점차적으로 오류비율이 감소하는 패턴으로 나타난다[1][7].

2.2 소프트웨어 오류 유형

2.2.1 오류

오류(Errors)란 소프트웨어 결함을 발생시키는 우발적, 부주의한 행위를 말하며 이는 연산, 실제 값과 불일치, 사양에서 누락으로 인한 부적절한 처리, 부정확한 결과를 유도하는 논리 등을 말한다.

2.2.2 결함

결함(Default)이란 개발자 실수의 행위가 소프트웨어 내에 입력되는 것으로 프로그램 실행결과가 본래의 요구명세서 내용과 다를 때를 결함이라 하며 이를 버그(bug)라고도 한다. 따라서, 결함을 없애는 작업을 디버깅(Debugging)이라 한다.

2.2.3 고장

고장(Failure)이란 소프트웨어 결함이 있으면 생기는 현상을 말하며 개발된 소프트웨어가 실행하면 사용자가 원하지 않거나 적절하지 않은 상황을 발생시킨다.

2.2.4 결점

결점(Defect)이란 소프트웨어 사용에 부적당하거나 명세서, 설계명세 및 실행조건과 일치하지 않는 경우를 말하며 이를 소프트웨어 하자라고도 한다.

2.2.5 변칙

변칙(Anomaly)이란 설계된 소프트웨어 요구명세서, 설계명세서 등과 다르게 변칙적으로 소프트웨어가 실행하는 것을 말한다.

2.3 소프트웨어 오류 분류

소프트웨어 오류는 프로그램 작성자의 숙련도 등과 같은 환경적 원인과 복잡한 소프트웨어 구성과 요구사양의 다양성으로 등의 원인으로 소프트웨어 개발의 모든 단계에서 발생하며 다음과 같이 크게 분류할 수 있다[2][3][10][14].

2.3.1 논리 오류

논리적 오류는 논리 표현 시 적절하지 않는 연산자와 피연산자를 사용한 경우, 순서에 맞지 않는 논리식 사용한 경우, 부적절한 변수 사용, 논리식 또는 조건 테스트 실수, 루프(무한 루프)의 정확하지 않는 반복 횟수 지정, 그리고 중복된 논리지정 등 여러 가지 상황을 들 수 있다.

2.3.2 데이터 오류

적절하지 않는 데이터를 이용하여 프로그래밍 시 사용(참조 또는 저장오류, 부정확/부적절한 변수지정, 데이터변환오류) 한 경우 나타나는 데이터 취급오류들이 있으며, 적절하지 않는 데이터를 초기화한 경우, 부정확한 데이터 단위 사용 및 변수선언 등으로 인한 데이터 정의 오류가 있다.

2.3.3 인터페이스 오류

소프트웨어의 화면을 디자인하는 과정에서 발생하는 구성오류, 잘못된 서브루틴이나 없는 서브루틴을 호출하는 경우, 데이터베이스의 부적절한 배치 또는 사용, 인터럽트의 부적절한 취급 등이 있다.

2.3.4 설계 오류

소프트웨어를 처음 설계 시 잘못 설계하여 발생하는 전반적인 오류(알고리즘 설계가 적절하지 않는 오류, 일관성이 없는 데이터베이스 및 인터페이스, 예외적인 조건을 무시한 오류, 작업순서에 따른 오류 등)들이다.

2.3.5 연산 오류

소프트웨어 개발을 위한 프로그래밍 시 적당하지 인터페이스 않는 연산자와 피연산자 사용한 경우, 잘못된 부호 및 등식 사용, 복잡한 연산식 사용으로 인한 결과의 정밀도 저하, 연산식 누락, 자리 반올림 또는 자리 절상(하) 등으로 인해 원하는 계산 결과가 도출되지 않는 경우의 오류를 말한다.

2.3.6 기타 오류

앞의 오류 외에 문법이해 부족, 소프트웨어 사용 미숙으로 인해 나타나는 오류, 하드웨어 오류, 컴퓨터 조작 오류, 단순한 오타 등이 있다.

2.4 프로젝트 수행과정에서 단계별로 발생할 수 있는 오류발생 원인

원하는 프로젝트를 수행하는 전체 프로그래밍 과정의 각 단계마다 오류를 발생시키는 요인들이 있다. 일반적으로 오류발생의 약 60%가 자신이 원하는 결과를 도출할 수 있도록 프로그래밍하기 전에 분석하고 설계하는 단계에서 나타나며 약 40%의 오류는 코딩단계에서 발생한다. 이러한 프로젝트 수행을 위한 프로그래밍 각 단계별 오류 발생 원인들을 분석하면 다음과 같다[6][9][11][14].

2.4.1 요구정의 단계

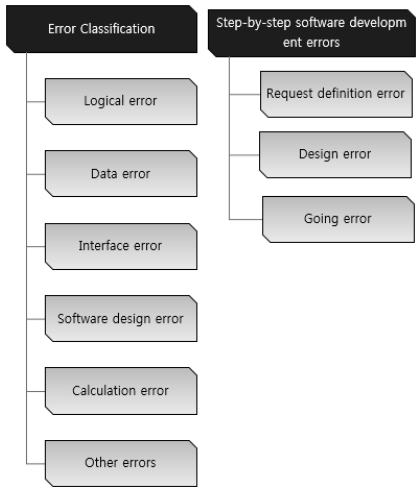
프로그램 작성자의 소프트웨어 개발능력이나 하드웨어 환경을 무시한 상태에서 자신들의 무리한 소프트웨어 능력 요구로 인한 요구정의로 인해 원천적이고 구조적인 프로그래밍 오류가 발생하는 경우로 이때 발생하는 오류는 수정하기가 어렵다.

2.4.2 설계 단계

하드웨어 및 사람 능력을 무시한 프로그래밍(시스템 설계), 불충분하고 구조화되지 않은 설계, 불필요하고 중복되거나 누락된 설계, 표준화되지 않은 언어사용으로 이해하기 어려운 설계 내용 및 예외적인 조건을 누락한 경우에 나타난다.

2.4.3 코딩 단계

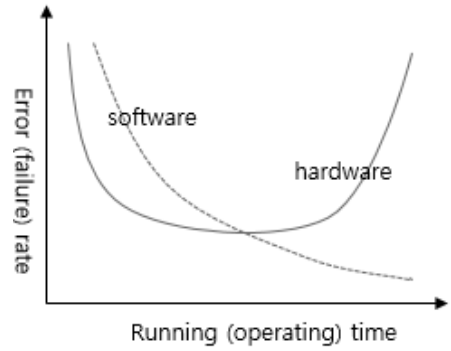
요구정의 및 설계단계에서 없거나 분석되지 않는 부분들을 추가, 프로그램 논리에 맞게 기능을 실행할 수 없도록 정확하지 않고 모순된 논리 추가, 프로그램 작성자의 능력 부족으로 모듈기능 및 데이터 정의 이해부족, 불필요한 논리추가, 타이핑 오류, 설계된 논리대로 코딩하지 않고 프로그램 작성자의 순간 판단에 의한 코딩, 프로그램 언어에 대한 충분한 지식 부족 등으로 인해 발생한다.



(Fig. 1) Types of errors that can occur during software development

2.5 하드웨어 오류

소프트웨어 오류는 일반적으로 소프트웨어 개발과정에서 입력된 결함이나 논리적 착오 등으로 오류가 발생하지만 하드웨어는 보통 구성 부품의 잘못된 설계 등의 제조과정에서 발생하는 오류와 제품 사용이 어느 정도 진행하여 부품의 마모나 피로도에 따라 오류율은 급격히 증가하는 현상이 발생한다. 따라서, 오류 발생을 줄이기 위해서는 마모 고장기간 내에 부품교환 및 수리 등을 실시하여 오류를 예방할 수 있다. 따라서, 소프트웨어 오류 유형과는 다르다.



(Fig. 2) Hardware and software error patterns

2.6 선행연구

소프트웨어 교육을 위한 프로그래밍 작성 과정에서 발생되는 오류를 수집하고 분석하여 향후 프로그래밍 학습 시 피드백하기 위한 선행 연구들은 거의 없으며 대부분 최적의 상업용 소프트웨어 개발을 위해 개발 전 과정 특히 테스트 과정에서 오류를 수집하고 수정하는 선행연구들은 일부 볼 수 있다. 소프트웨어 품질을 평가하는 연구로서 Myron Lipow는 많은 전문적인 상업용 소프트웨어 개발과정에서 프로그래머들로 부터 수집하고 분석한 다량의 오류데이터의 전체 평균 발생빈도에서 논리오류가 30%로 가장 많고 데이터 취급, 정의, 입출력 오류, 인터페이스 오류가 각각 17%, 7%, 11%, 17%이며 연산오류가 8%, 데이터베이스 오류가 5%이었으며 코딩 등의 단순한 오류가 5%로 분석되었다[10]. 10인 이상 50개의 소프트웨어 개발업체에서 프로그래밍 경력이 1년 전·후의 사원들을 대상으로 프로그래밍 과정에서 직접 경험한 오류들을 설문형식으로 조사한 오류 데이터 발생빈도에서 논리오류가 31%, 데이터정의 오류가 15%, 데이터 취급 및 연산오류가 각각 13%, 인터페이스 오류가 11%, 입출력오류가 10%, 데이터베이스 오류가 7% 그리고 코딩 등의 오류인 기타오류가 7%로 수집 및 분석되었다[8]. 선행연구에서의 결과도 본 연구와 마찬가지로 가장 많은 오류 종류는 논리오류로 프로그래밍 시 주어진 결과를 도출하는 논리적 과정이 가장 어려움이 있는 것으로 판단된다.

3. 오류 데이터 수집과 분석

3.1 데이터 수집 대상

본 연구는 소프트웨어 개발기업에서 행하는 테스트를 중심으로 수집하는 기존 오류데이터 수집방법과는 다르게 오류데이터를 수집하였다. 연구에 필요한 오류데이터 수집은 진주교육대학교 3학년 교과인 “소프트웨어와 프로그래밍 언어교육”을 수강하는 학생들 대상으로 A반(25명)은 스크래치를 선행 학습하고 다른 한반인 B반(25명)은 파이썬을 선행 학습한 후 매월(4, 5, 6월) 마지막 주에 초등교과 내용을 기초로 담당교수가 프로젝트를 제시하고 이들 문제들을 해결하기 위해 학습자 스스로 알고리즘을 구현하고 프로그래밍 한 결과를 도출하는 과정(1시간 이내)에서 발생한 모든 오류들을 수집하고 분석하였다. 프로그래밍 과제는 <Table 1> 처럼 초등학교 5학년 수학, 과학, 6학년 수학, 과학교과에서 특정 영역의 단원을 응용한 과제로 스크래치 및 파이썬으로 각각 프로그래밍 하였다.

<Table 1> The curriculum used for programming and error collection

5, Grade 6 Curriculum	
Curriculum	Regions and sections
Mathematics (5th grade)	Shape, joint, symmetry
Science (5th grade)	Materials and energy, acids and bases
Mathematics (6th grade)	proportional and proportional allocation
Science (6th grade)	Matter and energy, combustion and digestion

3.2 오류 데이터 수집

학습자들의 프로젝트 수행과정에서 발생시키는 오류들은 전문 소프트웨어 개발자들을 위해 Lipow가 제안한 8개(16개의 세부항목)의 오류항목들을 기초로 하였다[10]. 프로그래밍이 미숙한 학습자들의 실정에 맞게 5개 항목(논리오류, 연산오류, 데이터오류, 인터페이스오류, 기타오류)으로 하였다. 각 항목의 세부적인 오류 종류들을 모두 9종류로 분류하였다. 프로젝트 수행과정에서 발생한 총 오류건수는 각반 25명의 학습자들이 4과제의 프로젝트를 수행하는 과정에서 발생한 총 누적 건수로 파이썬이 622건, 스크래치가 388건이다. 파이썬 프로그램 작성자 1명당 평균 오류 수는 6.22개이다.

<Table 2> Collected error data and classification

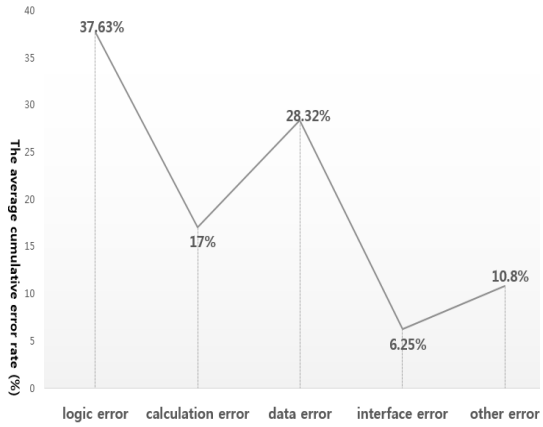
Error item	Division(Type of error)	Number of errors	
		Python	Scratch
Logical error	Invalid logical expression	65	48
	Use improper operator or sign	48	18
	Using an infinite loop	40	22
	Using an inappropriate control structure	45	36
	Using logical result is wrong	46	26
	Sub total	244	150
Calculation error	Calculation not specified or invalid specification	43	43
	Specifying an expression that does not fit the results	59	27
	Sub total	102	70
Data error	Specifying inappropriate data	43	20
	Incorrect input and output formatting	44	40
	Input and output non-specified	34	24
	Incorrect Data Property	25	27
	Sub total	146	111
Interface error	Incorrect Wallpaper design	34	28
	Sub total	34	28
Other error	Windows (Open and Close) Misuse	7	15
	Lack of understanding of grammar, simply typing wrong	89	14
	Sub total	96	29
Total errors		622	388

<Table 2> 결과에서 파이썬과 스크래치 작성자의 한 과제당 발생시키는 평균 오류 수는 6.22개와 3.88개로 파이썬으로 프로그래밍 하는 학습자가 더 많은 오류를 발생시켰다. 이는 파이썬 문법이 비교적 스크래치에 비해 어렵고 직접 영어로 텍스트를 입력하다보니 알고리즘 표현과 코딩 중 오류가 더 많았던 것으로 판단된다.

3.3 오류 항목 분석

3.3.1 오류 데이터 비율

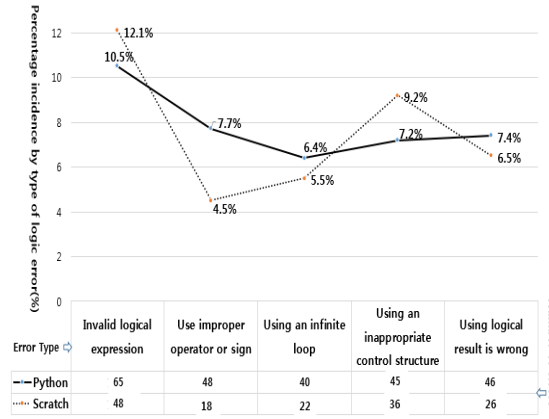
5개 항목별 오류데이터 중에서 가장 많이 발생시키는 오류는 논리오류로 파이썬이 244건으로 전체 오류의 39.2%이고, 스크래치가 전체 오류의 38.7%로 나타나 프로그램 작성 시 발생시키는 전체 오류 중 $\frac{1}{3}$이상으로 평균 37.63%를 차지하고 있다. 두 번째로 많이 발생된 오류데이터는 부적절한 데이터를 지정하거나 입출력 미지정 및 잘못된 입·출력 형식 지정 등으로 발생하는 데이터오류 항목이 28.32%로 나타났다. (Fig. 3)은 프로그램 작성 과정에서 수집한 전체 오류의 항목별 평균 비율을 나타내고 있다.



(Fig. 3) The error rate per item

3.3.2 논리 오류 수집 및 분석

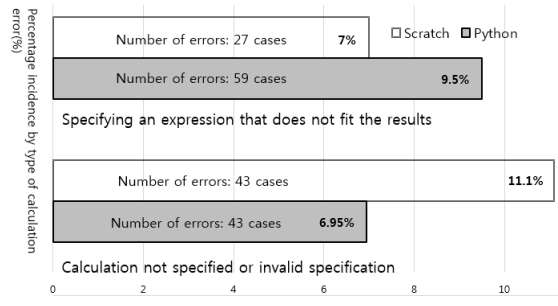
논리오류에서의 항목별 세부오류 발생 수집은 프로그래밍 과정에서 논리식을 잘못 사용한 경우가 파이썬이 65건, 스크래치가 48건으로 나타나 전체 오류의 10.5% 그리고 12.1%로 가장 많이 나타났으며, 부적절한 제어구조를 사용하거나 부적절한 연산자 사용 그리고 결과가 잘못된 논리사용, 무한루프 사용 순으로 오류를 발생시켰다. 학습자들이 주어진 프로젝트를 결과로 도출하는 논리적인 알고리즘 표현이 가장 어려운 부분으로 분석되었으며 향후 프로그램 복잡도가 높아지면 더욱 많은 오류가 지속적으로 발생할 것으로 판단된다. 이는, 소프트웨어 개발업체에 입사한지 1년 전·후의 프로그래머가 가장 많이 발생시키는 오류가 논리오류로 비슷한 현상을 보였다[8]. 논리오류는 불럭조립형 언어(스크래치)나 텍스트입력형 언어(파이썬) 둘다 오류 비율에 있어 큰 차이가 나지 않았다.



(Fig. 4) Logical error details

3.3.3 연산 오류 수집 및 분석

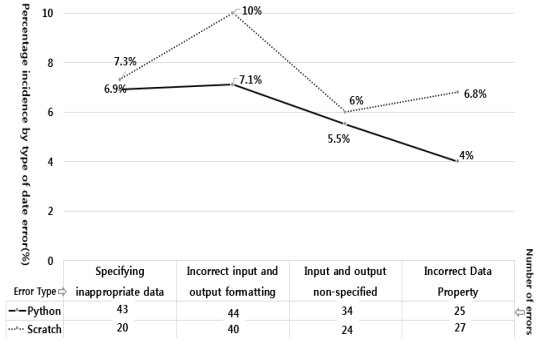
연산오류의 항목별 세부오류 발생은 프로그래밍 과정에서 결과에 맞지 않는 식을 지정하여 오류를 범한 경우가 파이썬이 9.5%(59건), 스크래치가 7%(27건)이었으며 연산이 지정되지 않거나 잘못된 수식을 지정하여 발생한 오류가 파이썬이 6.95%(43건)이고 스크래치가 11.1%(43건)로 전체 오류 유형 중 3번째로 많이 발생시켰다. 이는 연산을 위한 과정과 논리적 개념이해 부족에서 나타난 현상이라 판단된다.



(Fig. 5) Calculation error details

3.3.4 데이터 오류 수집 및 분석

데이터 오류의 항목별 세부오류 발생은 부적절한 데이터 지정이 파이썬은 6.9%, 스크래치가 7.3%이며 잘못된 입·출력 형식 지정이 7.1%와 10%로 각각 나타났으며 입출력 미지정이 5.5%와 6%로 나타났다. 또한, 데이터 속성을 잘못 지정한 오류도 파이썬과 스크래치 각각 4%와 6.8%로 발생하였다.



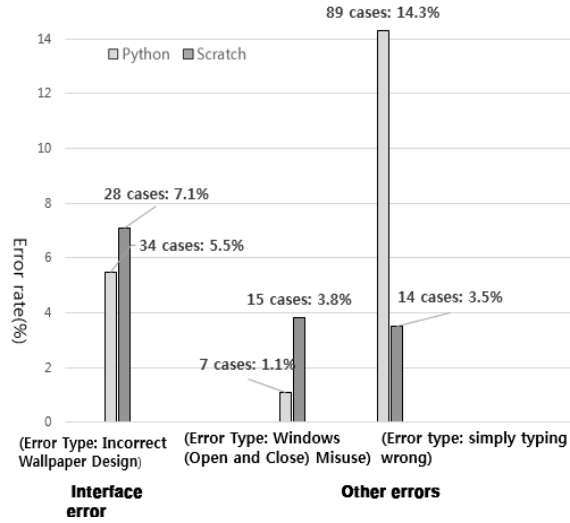
(Fig. 6) Data Error detailsy

3.3.5 인터페이스 및 기타 오류 수집 및 분석

속성 잘못 지정, 화면 디자인 등의 설계 잘못으로 인해 발생되는 인터페이스 오류에서 파이썬과 스크래치에서 각각 5.5%와 7.1%로 나타났다.

이는 제시한 프로젝트가 복잡한 속성 지정, 상세한 화면설계를 필요로 하지 않는 비교적 단순한 과제를 수행한 결과로 판단된다. 기타오류에서는 열기, 저장 등의 시스템사용 미숙과 일부 노후화된 컴퓨터시스템의 하드웨어 장애로 인한 단순한 오류가 파이썬이 1.1%(7건), 스크래치가 3.8%(15건)로 오류가 가장 적게 발생하였다. 그러나 텍스트를 입력하는 파이썬에서의 오타 발생으로 14.3%(89건)로 수집되어 스크래치에서 발생한 3.5%에 비해서 4배 정도 차이를 보였다. 이는 파이썬 프로그래밍이 불럭조립형 언어인 스크래치에 비해 오타발생이 월등히 많이 발생시키고 있음을 알 수 있으며 코딩 경험이 많을수록 복잡한 문법 이해와 사용미숙 해소 등으로 인해 코딩오류를 많이 줄일 수 있을 것으로 판단된다.

그러나, 단순한 오타를 발생시켜 이를 찾고, 수정하는 지루한 작업으로 인해 초등학생 및 프로그래밍을 배우는 초보자들인 경우 프로그래밍 학습을 기피할 수도 있어 학습자의 능력에 따라 프로그래밍도구 선택에 신중할 필요가 있다.



(Fig. 7) Details of interfaces and other errors

4. 결론

소프트웨어교육은 상상력과 창의성을 키울 수 있는 최적의 학습방법이다. 특히, 초등학교 전체 교과에서 상상력과 문제해결력을 키워 학업에 대한 성취도와 성적 향상에 큰 도움을 줄 수 있다.

본 연구에서는 2018년부터 실시되는 초등학교 소프트웨어교육에 대비해 예비교사들의 프로그래밍교육 능력 향상과 프로그래밍 시 어떤 문제점과 어려운 점들이 발생할 수 있는지를 미리 파악하여 교사가 코딩교육에 적절하고도 최적의 방법을 찾을 수 있는 능력을 키우기 위해, 예비교사들에게 파이썬과 스크래치를 각각 선행학습 시킨 후 제시한 프로젝트를 구현하기 위해 프로그래밍하고 그 과정에서 오류 데이터들을 유형별로 수집하고 분석하였다.

분석된 오류데이터를 예비교사들에게 피드백 시켜 소프트웨어교육에 활용하면 초등학생들에게 교육전에 프로

그래밍 학습 난이도, 성취도를 예측하여 학습자들에 맞는 최적 소프트웨어교육을 시킴으로서 흥미도 및 성취도를 점진적으로 높여 성공적인 학습 효과를 얻을 수 있다.

가장 많이 발생한 논리오류를 줄이는 방안으로 알고리즘 구현 학습을 위한 충분한 과정이 필요하며 이 과정은 상상력과 창의력을 향상시킬 수 있는 가장 중요한 소프트웨어교육의 핵심 목표라 할 수 있다.

또한, 텍스트를 입력하는 프로그래밍 도구에서는 어려운 문법 이해 및 사용, 생소한 영문 코딩 등의 원인으로 많은 량의 단순한 타이핑 오류가 발생함으로서 성공적인 초등학교 소프트웨어교육 목표 달성에 저해되는 중요한 요인이 될 수 있다. 따라서, 소프트웨어교육 실시 전 초등학생들에게 적합한 프로그래밍언어 선택에 신중을 기할 필요가 있다.

참고문헌

[1] Amrit L. Goal Software Error Dection Model with Application, *The journal of system and software Vol.1 27*, No. 4, pp. 17-25, 2013.

[2] Andy Oram, Greg, Wilson(2010). Making Software : O'reilly Media.

[3] Georage rawvski, Identification of factors which cause software failure, proceedings Annual reliability and maintainability IEEE Symp, 2013.

[4] Hyungshin Choi(2014). Computational Thinking Gramework-based Analysis of After-school Scratch Team project Experience Journal of the Korean Association of Information Education, 18-4, 549-553.

[5] Jun yongju(2017), "A Study on the Cognitive, Psychomotor and Affective Aspects of Software Education by Exploring Computational Thinking", Korea Computer Education Association Summer conference (2017).

[6] Kang jungsuk(2016), The status of software education in preparation for the fourth industrial revolution era, Jeju Special Self-Governing Province Office of Education.

[7] Littlewood, B and Miller D. Software Reliability and safety, Elsevier applied science, London, 1993.

[8] Moon, W. S(2006). Analyzing the types of errors that elementary students encounter in programming learning. *Korea Computer Information Association Vol 11*, No. 2, 2006, pp. 324-326

[9] Muneo Takahssi and Yuki Kamyachi, An Empirical Study of A Model for Program Error Prediction, Proceedings of 8th IEEE Conference on Software Engineering", pp. 331

[10] Myron Lipow, Prediction of software failure, *Journ of system and software, Vol 15*, No 4, pp. 31 - 35, 2012.

[11] Rost, Johann, Glass(2011). Dark Side of Software Engineering : Wley-IEE Computer Socity PR

[12] Shin Seung-yong(2003). Development of information gifted discrimination tool based on creativity and information science characteristics, Korean Computer Education Association Summer Lecture(2).

[13] Youngsik Jeong, Kapsu Kim, Inkee Jeog, Hyunbae Kim, Chul Kim, Jeomgsu Yu, Chongwoo Kim, Myunghui Hong(2015). A Development of the Software Education Curriculum Model for Elementary Students. *Journal of The Korean Association of Information Education. 19*(4). 467-475.

[14] Yongra Kwon(2010). Software Testing : Seng leng Publishing Co.

저자소개

문 외 식

현재 진주교육대학교 컴퓨터교육과 교수 (전공: 소프트웨어공학)

관심분야: 소프트웨어 신뢰도 및 품질평가, ICT 활용교육, 로봇 활용교육, 프로그래밍 및 소프트웨어 교육

e-mail: wsmoon@cue.ac.kr