UML과 LVOOP를 이용한 프로그래밍 교육의 적용 방안

정덕길*· 정민포** · 조혁규**

*동의대학교, **영산대학교

The Application to the Programming Education Using UML and LabVIEW OOP

Deok-Gil Jung* · Min-Po Jung** · Hyuk-Gyu Cho**

*Dongeui University, **YoungSan University

E-mail : minpo@ysu.ac.kr

요 약

컴퓨터 언어를 배우는 학습자는 다양한 분야에 적합한 프로그래밍 언어를 배우고 텍스트 기반으로 된 프로그래밍을 하는 것이 매우 어렵다고 생각한다. 이러한 문제를 쉽게 풀기 위한 한 방법이 프로그램을 비주얼로 표현하는 것이다. 기존의 시각 프로그래밍인 Visual C++, Visual Basic, Delphi와 같은 비주얼 언어는 외부 인터페이스는 비주얼 컴포넌트로 표현되고 컴포넌트의 작동에 대해서는 텍스트 기반으로 표현한다. 이러한 프로그램을 배우는 학습자들은 컴포넌트 작동에 대한 텍스트 프로그래밍에 대해 어려워하고 있으며 프로그래밍을 싫어하는 한 요소가 되었다.

논문에서는 이러한 문제를 해결하기 위해 논리적 사고를 표현하면서 객체지향을 지원하기 위해 UML을 도입하고 텍스트 프로그래밍 요소를 비주얼 프로그래밍 요소로 대체하기 위해 객체지향을 지원하는 LabVIEW OOP를 사용하여 학습자들에게 프로그래밍 교육을 하는 방법을 제시하였다. 또한, 제시된 프로그래밍 교육 방법에 대해 설문조사를 실시하여 교육적인 효과를 분석하였다.

ABSTRACT

To learn a programming language as a text-based programming and a computer language suitable for a wide range, learner thinks it is very difficult. To represent a visual program is one way to solve this problem easily. The visual language such as Visual C++, Visual Basic and Delphi is represented an interface as the visual component and represented a component action as a text-based. The programmer is very difficult about the component action with text-based and dislikes programming.

In this paper, so solve these problems, we use the UML for representing a logical thinking and supporting and object-oriented programming. We suggest for programming education method to replace text-based programming to LabVIEW OOP. In addition, we conduct a survey on how programming education and analyze the training effect.

키워드

UML, LabVIEW OOP, Education, Visual Programming

Ⅰ. 서 론

컴퓨터 언어를 배우는 학습자는 다양한 분야에 적합한 프로그래밍 언어를 배우고 텍스트 기반으로 된 프로그래밍을 하는 것이 매우 어렵다고 생각한다. 이러한 문제를 쉽게 풀기 위한 한 방법이 프로그램을 비주얼로 표현하는 것이다. 기존의 시각 프로그래밍인 Visual C++, Visual Basic, Delphi와 같은 비주얼 언어는 외부 인터페이스는 비주얼 컴포넌트로 표현되고 컴포넌트의 작동에 대해서는 텍스트 기반으로 표현한다. 이러한 프로그램을 배우는 학습자들은 컴포넌트 작동에 대한 텍스트 프로그래밍에 대해 어려워하고 있으며 프로그래밍을 싫어하는 한 요소가 되었다[1]. 대학의 프로그래밍 언어 교육 현장을 살펴보면, 텍스

트 방식의 주입식 교육을 하는 것이 일반적이다. 이는 대부분의 수강생들이 프로그래밍 문제를 해결하기 위한 본질을 생각하기 전에 프로그래밍 문법의 어려움으로 인해 중도 포기를 하는 계기가 되었다. 이러한 문제를 해결하기 위해 논리적 사고를 표현하면서 객체지향을 지원하기 위해 UML을 도입하고 텍스트 프로그래밍 요소를 비주얼 프로그래밍 요소로 대체하기 위해 객체지향을 지원하는 LabVIEW OOP를 사용하여 학습자들에게 프로그래밍 교육을 하는 방법을 제시하였다. 또한, 제시된 프로그래밍 교육 방법에 대해 설문조사를 실시하여 교육적인 효과를 분석하였다.

II. 본 론

프로그래밍 언어 교육 시 수강생들에게 개발자, 분석가, 디자이너, 사용자들 간의 의사소통, 문서화, 시스템의 기능 분석 등의 능력을 갖추기를 원한다. 컴퓨터 프로그래밍 언어 분야에서 프로그래밍 언어란 논리적 사고의 표현 도구라고 표현하였다[2]. 논리적인 시스템은 그것이 하드웨어이건 소프트웨어이건 또는 이 둘의 결합된 형태이건 사람의 머릿속에서 형성된 논리적 체계가 현실 세계에서 실현된 것이다. 사람의 논리적 사고 과정이나 그 결과물을 표현하는 수단이 바로 프로그래밍 언어이다. 실제로 하드웨어 설계 분야에서는 회로도를 직접 그리기 보다는 VHDL과 같은 언어를 사용하여 설계하는 것이 현재의 추세이다. 따라서 프로그래밍 언어는 단순히 프로그래머의 프로그램 작성 도구로서만 교육되어서는 안 되며, 논리적인 사고를 계발하고 표현하는데 중점을 두고 교육이 되어야 한다.

이러한 문제에 접근하기 위해, 이 논문에서는 기존의 프로그래밍 교육 방식과 다른 개선된 교육 방식이 필요하다. 특히 논리적 사고를 표현하기 위해 체계화되고 검증된 소프트웨어 설계도구인 UML을 통해 프로그램에 대한 논리적인 사고를 효율적으로 표현하고, LabVIEW OOP와 같은 비주얼 개발 도구를 이용하여 구현하는 교육 방식을 제안한다[3][4].

II-1절에서 UML과 그래픽언어인 LabVIEW OOP와 텍스트 언어인 JAVA를 활용한 교육 적용 방안을 비교 설명한다. II-2절과 3절에는 교육적용 방안에 대해 설문조사를 하여 교육적인 효과를 분석하였다.

II-1. UML과 LabVIEW OOP/JAVA를 활용한 교육 적용 방안

프로그래밍 언어 교육현장에서 그래픽 언어인 LabVIEW OOP와 텍스트 언어인 JAVA를 이용한 프로그래밍 언어 교육에 대한 적용 단계는 그림 1과 같다.

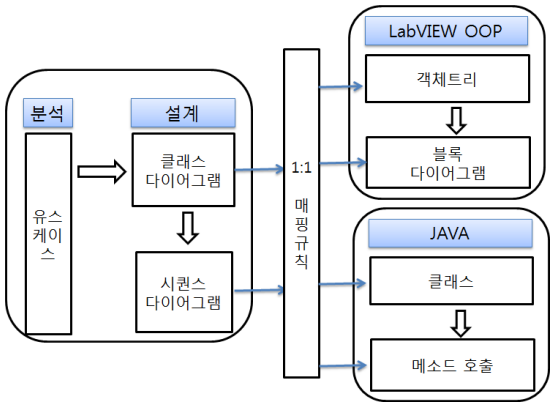

그림 1. UML 다이어그램의 LabVIEW OOP 클래스 및 JAVA 클래스의 1:1 매핑

UML로 설계된 유스케이스를 제시한다.

UML에 대한 교육 과정을 사전에 설계하면 매우 바람직하지만, 그렇지 못한 경우에는 UML의 개념과 표기법, 사례를 들어 충분히 설명을 한다. 프로그래밍을 한다는 것은 현실 세계에 대한 논리적인 사고를 정리하는 과정이다. 이 과정을 UML을 사용하여 충분히 사전에 설명하고 실습한다. 이 단계에서의 결과물은 요구 조건에 적합한 유스케이스를 제시한다.

설계된 유스케이스를 이용하여 UML의 클래스 다이어그램을 설계한다.

설계된 유스케이스를 충분히 학생들에게 이해를 시키고, 클래스 다이어그램을 설계하여 유스케이스를 만족하는 클래스를 만들어낸다. 이 과정은 클래스가 완성될 때까지 충분히 반복하여 유스케이스와 클래스 다이어그램을 여러 번 반복하여 완성한다. 이 단계에서의 결과물은 요구 조건에 적합한 클래스 다이어그램을 제시한다.

설계된 클래스 다이어그램을 이용하여 요구조건에 적합한 시퀀스 다이어그램을 설계한다.

설계된 시퀀스 다이어그램을 충분히 학생들에게 이해를 시키고, 클래스 다이어그램에서 만든 클래스를 사용하여 시나리오 적합한 시퀀스 다이어그램을 완성한다. 이 과정 역시, 시나리오->UML->클래스 다이어그램->시퀀스 다이어그램의 순으로 반복적으로 수정을 한다.

설계된 UML의 클래스 다이어그램을 사용하여 LabVIEW OOP의 프로젝트 탐색기에 1:1 매핑 규칙으로 클래스를 추가하여 객체 트리를 구성한다. JAVA에서도 1:1 매핑 규칙으로 클래스를 텍

스트로 추가한다.

이 단계를 진행하기 위해서, LabVIEW OOP 사용법과 JAVA 사용법에 대해 미리 사전 교육이 되어 있어야 한다. 만들어진 UML의 클래스 다이어그램을 이용하여 각각의 클래스를 LabVIEW OOP 탐색기에 추가한다. JAVA는 텍스트 언어로 추가한다. 추가시에는 1:1 매핑 규칙에 따른다.

설계된 UML의 클래스 다이어그램을 사용하여 LabVIEW OOP에서 생성된 클래스에 변수와 메소드를 추가한다. JAVA에서 생성된 클래스에 변수와 메소드를 추가한다.

UML의 클래스 다이어그램을 이용하여 각 클래스 내의 멤버 변수와 메소드를 LabVIEW OOP의 클래스에 추가한다. 이 단계에서는 프라이빗 데이터 컨트롤에 대한 개념과 메소드 vi에 대한 개념을 알고 있어야 한다. JAVA의 클래스에 멤버 변수와 메소드를 추가하기 위해 메소드 선언 방법과 변수 타입 등을 미리 알고 있어야 한다.

설계된 UML의 시퀀스 다이어그램을 사용하여 LabVIEW OOP에서 생성된 메소드 VI를 사용하여 블록 다이어그램에서 Wiring을 진행한다. JAVA에서 생성된 클래스 내의 메소드를 사용하여 시퀀스 다이어그램에 맞게 객체간의 메소드를 호출한다.

UML의 시퀀스 다이어그램을 이용하여 LabVIEW OOP내에 설계된 메소드 vi들을 서로 연결(wiring)한다. 이 단계에서는 LabVIEW의 블록 다이어그램에서 wiring에 대한 개념과 문법을 파악하고 있어야 한다. JAVA는 텍스트 에디터를 이용하여 메소드를 호출한다. JAVA의 메소드 호출 단계에서는 객체 간의 인스턴스를 만들어야 하고, 만들어진 인스턴스를 통해 호출한다.

요구사항에 따라 화면을 설계하기 위해, LabVIEW OOP에서 프런트 패널을 설계한다. JAVA에서는 화면을 설계하기 위해서는, AWT와 같은 별도의 라이브러리를 사용해야 한다.

LabVIEW OOP내의 프런트 패널을 사용하기 위해, 프런트 패널에 대한 컴포넌트를 어느 정도는 파악해야 한다. 화면 컴포넌트와 멤버 vi들 사이와 연결하여 최종 프로그램을 완성한다. JAVA에서 화면을 그래픽으로 디자인하기 위해서는 별도의 문법으로 되어있는 AWT와 같은 라이브러리를 호출하여 설계한다.

II-2. 교육효과 적용에 대한 설문

이 논문에서는 RFID의 FA(Factory Automation) 프로그램 개발에 대하여 교육하고, LabVIEW OOP 방법과 JAVA로 개발하는 것을 실험하였다. 5일의 교육을 통해 학생들에게 예제를 사용하여 교육을 실시하였다. 교육 실시 후, 표 1에 따르는 설문조사를 실시하였다. 그림 2에서 과정을 도식화 하였다.

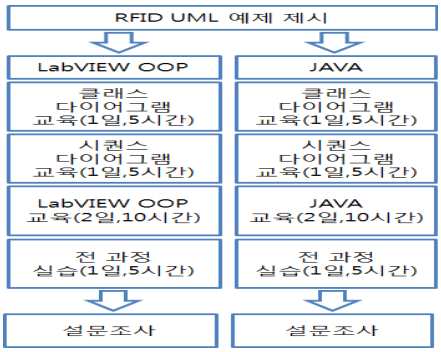

그림 2. 교육 적용 과정

표 1. 설문 조사 양식

평가 항목 \ 도구(언어) 표기법	UML	LabVIEW OOP	JAVA
요구사항 분석			
클래스 생성을 위한 요구분석 이해도(5점)			
분석된 클래스의 도식화(5점)			
분석된 클래스의 호출관계(5점)			
프로그램 설계			
클래스와 객체의 상관성 및 상호작용에 대한 이해도(5점)			
객체의 속성 값에 대한 이해 정도(5점)			
메소드의 호출 및 기능에 대한 이해정도(5점)			
프로그램 구현(코딩)			
클래스 시각 표현(5점)			
객체의 실체화의 용이성(5점)			
도구/환경 사용의 용이성(5점)			
클래스 구현과 인터페이스에 대한 이해도(5점)			
유지보수 가능성			
클래스 데이터 추가 및 삭제의 용이성(5점)			
메소드 추가 및 삭제(5점)			
상속의 용이성(5점)			
확장성			
클래스의 기능 추가 용이성			
클래스의 상속 기능 추가 용이성			
재사용성			
재사용을 위한 모듈 기능(5점)			
재사용을 위한 클래스 내부 변경(5점)			

II-3. 교육효과 적용에 대한 설문 결과

설문 조사 대상은 일반 IT 전공학과 3학년, 4학

년(시스템/보안 전공, C언어, JAVA 가능함) 40명을 대상으로 조사를 하였다. UML과 JAVA를 잘 못하는 학생들을 대상으로 미리 사전 교육을 5일 정도 실시하여 클래스 다이어그램, 시퀀스 다이어그램, 유스케이스 다이어그램을 이해하고 기본적인 것을 작성하는 수준이다. 대상의 30% 정도는 JAVA나 C를 개념적인 이해가 부족한 학생으로 판단된다.

이 논문에서 설문한 3가지 교육방법(UML, LabVIEW OOP, JAVA)에서 설문 문항별로 만족도의 집단 평균치에 차이를 자료 분석하여 표 2에서 정리하였다.

표 2. 5점 척도 기준의 평균분석

도구(언어) / 표기법 / 평가 항목	UML		LabVIEW OOP		JAVA	
	평균	표준편차	평균	표준편차	평균	표준편차
요구사항 분석						
클래스 생성을 위한 요구 분석 이해도	4.75	0.43	1.00	0.81	0.63	0.86
분석된 클래스의 도식화	4.88	0.33	2.50	0.95	2.00	1.24
분석된 클래스의 호출관계	4.88	0.33	2.50	0.87	2.00	0.84
프로그램 설계						
클래스와 객체의 상관성 및 상호작용에 대한 이해도	4.50	0.67	4.25	0.86	3.00	1.30
객체의 속성 값에 대한 이해 정도	3.75	0.80	3.00	1.47	2.50	1.50
메소드의 호출 및 기능에 대한 이해 정도	4.63	0.58	3.75	1.32	3.50	1.38
프로그램 구현(코딩)						
클래스 시각 표현	4.75	0.43	4.75	0.43	1.00	1.30
객체의 실체화의 용이성	2.50	0.77	4.50	0.77	2.50	1.64
도구/환경 사용의 용이성	2.50	0.95	2.50	1.26	2.00	1.38
클래스 구현과 인터페이스에 대한 이해도	3.00	1.00	4.50	0.81	2.50	1.53
유지보수 가능성						
클래스 데이터 추가 및 삭제의 용이성	4.38	0.83	3.75	1.37	3.00	1.63
메소드 추가 및 삭제	4.63	0.62	4.25	1.16	3.25	1.41
상속의 용이성	4.63	0.62	4.63	0.66	4.38	0.86
확장성						
클래스의 기능 추가 용이성	4.50	0.87	4.25	1.04	3.75	1.61
클래스의 상속 기능 추가 용이성	4.50	0.87	4.00	1.10	3.75	1.34
재사용성						
재사용을 위한 모듈 기능	4.50	0.77	4.25	0.83	3.00	1.60
재사용을 위한 클래스 내부 변경	4.50	0.77	3.75	1.07	2.50	1.64

17개의 설문문항은 6개의 평가 문항으로 분류하였다. 각 평가 문항의 분석에서 설문 문항별로 UML, LabVIEW OOP, JAVA의 평균값을 차트로 표시하였다. 본 연구의 자료 분석은 IBM SPSS Statistics 19를 사용하였으며 5점 척도의 채점표 분석에 평균분석을 이용하였다. 채점표의 5점 척도로 '매우 쉽다' 5점, '쉽다' 4점, '보통' 3점, '조금 어렵다' 2점, '매우 어렵다' 1점으로 하였다. 실증적 분석 방법은 일원분산분석법을 활용하여 분석하고 신뢰도 95%의 범주에 포함되면 유의성 있다고 해석하였다. 어떤 교육방법들이 평균차이를 보이는지에 대한 사후검증을 위해 Scheffe를 사용하였다.

III. 결 론

프로그래밍 언어 교육 시, 논리적인 사고를 텍스트가 아닌 그래픽 방식으로 표현하여 교육적인 효과를 높이는 것은 매우 중요하다. 이 논문에서는 논리적 사고를 표현하기 위해 체계화되고 검증된 소프트웨어 설계도구인 UML을 통해 프로그램에 대한 논리적인 사고를 효율적으로 표현하고, LabVIEW OOP와 같은 비주얼 개발 도구를 이용하여 구현하는 교육을 시행하였다.

프로그래밍을 개발할 때, 요구 분석 단계 및 프로그램 설계 단계에서는 UML과 같은 객체지향 그래픽 설계 도구에서 이해도가 높았으며, 프로그램 구현 단계에서는 LabVIEW OOP로 프로그래밍하는 것이 효율적이라는 결과가 나왔다.

결론적으로, 객체지향 개념을 가지는 도구로 먼저 개념을 도식화하고 LabVIEW OOP와 같은 그래픽 도구로 도식화된 개념을 프로그래밍하는 것이 가장 효과가 높았다.

참고문헌

[1] 주식회사 리얼게인, "그래픽 프로그래밍 언어 개발", 정보통신 산업기술개발사업, p.3, 2003.

[2] 표창우, "프로그래밍 언어 교육 현황 및 개선 방향", 정보과학회지 제16권 제9호, pp.55-57, 1998.

[3] 김상욱, "시각 프로그래밍 기술", 한국통신학회지 제11권 제5호, pp. 33- 45, 1994.

[4] 이성환, 박희선, "시각 프로그래밍과 프로그램 시각화 기술", 정보과학회지 제9권 제5호, pp.41-55, 1991.