論文 98-23-9-18

MVPE : 멀티패러다임 시각 프로그래밍 환경

정회원 유 재 우*, 최 종 명*

MVPE : Multiparadigm Visual Programming Environment

Chae-Woo Yoo*, Jong-Myung Choi* *Regular Members*

요 약

시각 프로그래밍은 많은 장점을 가지고 있지만, 범용 프로그래밍 언어로 사용되기 어려운 제약 사항들을 가지고 있다. 이러한 단점을 보완하기 위한 방법으로 멀티패러다임을 도입하려는 시도가 있어왔다. 그러나, 기존의 연구들은 특정 모델이나 구조적인 방법 없이 비효율적인 패러다임들의 결합을 시도하였기 때문에 객체지향을 지원하지 못하고, 시각 프로그래밍의 단점을 크게 극복하지 못했다. 이러한 문제점을 해결하고자 본 논문에서는 시각 프로그래밍에서 가장 성공적이었던 패러다임들 - 데이터 플로우 패러다임, 스프레드쉬트 패러다임, 직접 조작 패러다임, 객체지향 패러다임 - 을 "메소드 = 패러다임"이라는 개념 모델을 바탕으로 결합한 멀티패러다임 시각 프로그래밍 시스템을 소개한다. 논문에서 제시한 멀티패러다임 시각 프로그래밍 시스템은 다양한 형태의 시각 프로그래밍 패러다임을 결합함으로서 멀티패러다임의 장점을 활용할 수 있고, 시각 프로그래밍의 단점을 보완할 수 있다.

ABSTRACT

Although visual programming is used in many fields of computer science and engineering, some dis- advantages can be found when they work together in an integrated programming environment. To overcome these shortcomings, there have been researches in combining multiparadigm with visual programming. However they have failed because they tried to combine the paradigms without any conceptual model and structured method. In this paper, we investigate a new multiparadigm visual programming environment(MVPE), in which dataflow paradigm, form-based paradigm, direct manipulation paradigm, and object-oriented paradigm are integrated together in an object-oriented way, based on the conceptual model of "method = paradigm." This MVPE would overcome the limits of visual programming, and may also lead to the new discipline of visual programming environment.

I. 서 론

시각 프로그래밍이란 "프로그래밍 과정에서 의미 있는 그래픽을 사용하는 것[1]", 혹은 "프로그램을 2차원 이상으로 기술하는 것[2]"을 의미한다. 시각 프로그래밍은 구체성[3], 언어의 장벽이 없음[4], 생산성 향상[5], 오류 감소[6], 배우기 쉬움[7] 등의 장점을 지니고 있기 때문에 여러 방면에서 시각 프로그래밍에 관한 연구를 수행하고 있다. 상업적인 면에서도 "visual"

* 숭실대학교 컴퓨터학부
論文番號 : 98093-0303
接受日字 : 1998年 3月 3日

이라는 단어를 붙인 많은 프로그래밍 환경(Visual Basic, Visual C++, VisualWorks 등)이 발표되고 있는 것도 이러한 이유 때문이다. 그러나, 이러한 프로그래밍 환경을 모두 시각 프로그래밍이라고 부를 수는 없다. 시각 프로그래밍은 Prograph[4], LabView[9] 등과 같이 좀 더 높은 수준의 시각 표현력을 가지고 있어야 한다.

시각 프로그래밍은 많은 장점을 가지고 있지만 아직 상업적으로 성공한 제품이 나오지 않고 있는 이유는 비효율적인 표현[11], 제한된 도메인[11], 제한된 자료 타입과 연산자[11], 알고리즘 표현의 어려움[8] 등의 문제점 때문이다. 이러한 문제점들 때문에 시각 프로그래밍은 제한된 분야에서만 사용되어왔다. 대부분의 시각 프로그래밍은 하나의 프로그래밍 패러다임이나 계산 모델에 바탕을 두고 있으며, 매우 구체적이라는 특성을 가지고 있기 때문에 사용된 패러다임에 적합한 문제는 수월하게 해결할 수 있지만, 그렇지 않은 경우에는 문제를 해결하기 어렵다는 단점을 가지고 있다[11]. 반면에 실세계의 큰 시스템은 모든 프로그래밍 특성들을 가지고 있기 때문에 한 패러다임만을 지원하는 시각 프로그래밍 언어나 도구를 이용해서 문제를 해결하기에는 많은 어려움이 있다.

C++를 사용해 작성한 객체지향 프로그램과 Prolog를 사용한 로직 프로그램, MS사의 Excel을 사용한 프로그램은 프로그램의 형태가 상이하다. 이렇게 서로 상이한 프로그래밍 스타일을 패러다임이라 한다. 보다 일반적으로 말하면 프로그래밍 패러다임이란 "문제를 해결하기 위한 접근 방법 혹은 모델[24]", "설계 과정의 틀을 제공하고 프로그램의 구조를 결정하는 개념적 패턴[20]"을 의미한다. 일반적으로 한 패러다임은 좁은 범위에 한정되어 있기 때문에 어떤 특정 영역의 문제를 기술하는데 매우 효과적이지만, 크고 복잡한 시스템을 기술하는데는 어려운 점이 많다[17]. 예를 들면, 데이터플로우 프로그래밍 패러다임은 자료를 변환하는 함수를 기술하는데는 편리하지만, 복잡한 자료구조를 기술하기에는 적합하지 않다. 한편, programming-by-example 시스템은 자료 구조를 기술하고 변환하는데 편리하지만, 자료의 흐름이나 제어의 흐름을 기술하기는 어렵다. 마찬가지로, 함수형 프로그래밍 패러다임은 정형화된 이론을 증명하는데 적합한데 비해서, 로직 프로그래밍 패러다임은 규칙 기반(rule-based) 시스템을 표현하는데 효과적이다[12,13]. 이렇게 패러다임을 하나만 사용하는 경우에 발생하는 문제를 해결하기 위해 텍스트 프로그래밍 언어에서는 LEDA[10,16], Blueprint[12,13] 등 다양한 패러다임을 결합하는 멀티패러다임 언어에 관한 연구가 많이 이루어졌다.

그러나, 현재까지 개발된 대부분의 시각 프로그래밍 언어는 하나의 계산 모델이나 패러다임만 지원하고 있고, 멀티패러다임 시각 프로그래밍 언어에 관한 연구는 매우 적게 이루어 졌다. 여러 프로그래밍 패러다임을 동시에 지원하는 멀티패러다임 시스템은 제한된 도메인에서만 효율적인 시각 프로그래밍의 한계점을 해결하기 위한 하나의 방법이 될 수 있다. 현재까지 Visual ToolSet[11], Vista[14] 등이 멀티패러다임 시각 프로그래밍 언어로 개발되었지만, 각 언어는 서로 다른 목적을 위해 개발되었다. Visual ToolSet은 시각 프로그래밍의 단점을 보완하기 위한 목적으로 개발되었고, Vista는 시각 모델을 이용한 프로그램 구현 및 설계를 지원하기 위해서 개발되었다.

본 논문의 "멀티패러다임 시각프로그래밍 환경(MVPE)" 시스템은 Visual ToolSet과 같이 시각 프로그래밍의 문제점을 보완할 목적으로 구현되었다. MVPE는 Visual ToolSet에서 제공하지 못했던 객체지향 패러다임, 스프레드쉬트 패러다임, GUI 인터페이스 개발도구를 "클래스의 메소드 = 프로그래밍 패러다임"이라는 개념 모델을 바탕으로 개발된 멀티패러다임 시스템이다. MVPE는 반 독립적인 4개의 프로그래밍 도구를 이용해서 객체지향, 데이터플로우, 스프레드쉬트, 직접조작을 지원한다. MVPE는 C++언어를 바탕으로 구현되었으며, 4개의 패러다임은 각 프로그래밍 도구들에 의해 지원되고, 각 도구들은 C++ 소스 코드를 생성한다.

논문의 제 2장에서는 본 논문과 관련 있는 연구를 소개하였고, 제 3장에서는 MVPE 시스템 모델을 기술하였다. 제 4장에서는 본 논문을 통해 구현한 MVPE 시스템을 소개하였고, 제 5장에서는 시스템의 구현 및 평가, 제 6장에서는 마지막으로 결론을 기술하였다.

II. 관련 연구

1. Visual ToolSet

Visual ToolSet[11]은 범용 프로그래밍 언어로 사용되지 못하는 시각 프로그래밍 언어의 한계를 극복하기 위해 멀티패러다임을 도입한 시각 프로그래밍 언

어이다. Visual ToolSet은 서로 다른 패러다임을 지원하는 간단한 서브 시각 프로그래밍 언어들의 집합으로 구성되어 있다. 각 서브 시각 언어들은 자기 자신의 언어로 프로그램을 작성할 수 있는 도구를 가지고 있다. Visual ToolSet은 5개의 반 독립적인 도구들로 구성되어 있으며 도구들은 서로 통신할 수 있고, 한 도구에서 정의된 함수, 프로시듀어, 데이터는 다른 도구에서 사용될 수 있다. Visual ToolSet은 다음과 같은 도구들을 가지고 있다.

(1) 자료정의 도구(DDT) : 새로운 자료 타입과 변수를 정의한다.

(2) 함수정의 도구(FDT) : 기존의 함수들을 이용해서 데이터플로우 패러다임 방법으로 새로운 함수를 정의한다.

(3) DBMS 도구 : 자료를 테이블 형태로 관리, 저장, 질의어 처리 등의 작업을 수행하는 함수를 정의한다. 이 도구는 Query-by-Example[25] 모델에 기초를 두었고, 계산적인 기능은 부족하지만 주어진 관계에 따라 데이터 튜플의 갱신 및 질의어 처리에 편리하다.

(4) 제어흐름 도구(CFT) : 이 도구에서는 제어의 흐름이 프로시듀어를 수행하는 플로우차트 그림으로 표현된다. 이 도구는 자료나 화면의 출력은 기술하지 못하고, 미리 정의되어 있는 프로시듀어를 이용해 고수준의 제어를 수행한다.

(5) Programming-By-Example 도구 : 함수를 자료에 적용하거나 사이드 이펙트를 시각화하는 작업, 프로시듀어를 정의하고 실행하는 기능, 다른 도구에서 정의한 프로시듀어를 통합하는 기능을 제공한다.

Visual ToolSet의 서브 시각 프로그래밍 언어들은 각각 사용할 수 있는 연산자, 가능한 프로그래밍구조, 계산 기능을 가지고 있다. Visual ToolSet은 시각 프로그래밍의 문제점을 해결하기 위해 멀티패러다임을 도입했다는데 큰 의의를 갖고 있다.

2. Andrew

Andrew[15] 시스템에서는 시각 프로그래밍 분야에서 그래픽과 텍스트를 동시에 사용하는 방법에 관한 연구가 많이 이루어지지 않은 것에 착안해서 시각 문장과 텍스트를 동시에 지원할 수 있는 프로그래밍 모델을 제시했다. 시각 프로그래밍에서 텍스트는 프로그램의 저장, 인쇄, 프로그램에 주석 첨가 등의 용도로 사용될 수 있다. Andrew 시스템에서 시각 문장은 그래픽이나 텍스트 형태로 서로 호환적으로 표현될 수 있는 기능을 지원한다. Andrew 시스템은 여러 서브 시각 프로그래밍이 결합되는 경우에 각 서브 프로그래밍의 원소들이 그래픽이나 텍스트로 표현될 수 있는 멀티패러다임 모델을 제시하였다. 결론적으로 Andrew ToolKit은 멀티패러다임 시스템이 아니라 멀티패러다임 시스템을 위한 환경으로 사용될 수 있는 것이다.

3. Vista

Vista[14]는 Smalltalk 언어를 바탕으로 소프트웨어의 설계 및 프로그래밍을 지원하기 위한 시각 프로그래밍 환경이다. Vista는 Smalltalk의 객체지향 설계와 프로그래밍 방법을 시각 모델을 이용해서 프로그램하기 위해서 객체지향과 데이터플로우 패러다임을 결합하였다. Vista는 데이터와 시그날을 처리할 수 있는 시각 언어와 컴포넌트를 시각적으로 설계할 수 있는 기능을 제공한다. Vista는 Smalltalk의 라이브러리를 접근해서 사용할 수 있으며 필요한 곳에서는 텍스트를 사용할 수 있다. Vista는 시각 프로그래밍에서 소프트웨어의 설계를 지원한다는 점에서 의의가 있다.

III. MVPE 모델

도메인이 제한된 단일 패러다임의 한계를 극복하기 위한 멀티패러다임 언어는 다음과 같은 요구사항들을 충족시켜야 한다.

- 2개 이상의 패러다임 지원
- 다양한 표기법 및 문법적 사항 수용
- 불필요한 상호작용 배제
- 모듈단위의 패러다임 결합

MVPE 시스템이 멀티패러다임 시스템이 되기 위해서는 최소한 2개 이상의 프로그래밍 패러다임을 지원해야 한다[12,13].

멀티패러다임에 사용되는 각 시각 프로그래밍 패러다임은 그 패러다임에 맞는 데이터/연산자 표기법 및 문법적 사항을 이용할 때 가장 잘 프로그래밍에 이용될 수 있다[12,13]. 따라서 멀티패러다임 시스템은 각 패러다임에 맞는 데이터/연산자 표기법들을 사용할 수 있도록 지원해야 한다.

멀티패러다임 시스템에서는 다양한 표기법 및 문

법적 사항은 수용하지만, 각 패러다임간에 불필요한 상호 작용은 배제해야한다[12,13].

각 패러다임으로 작성된 프로그램은 모듈 단위이어야 한다. 패러다임간 결합은 프로세스 단위[18], 모듈단위[17], 문장/연산 단위[16]로 이루어 질 수 있다. 모듈 단위의 패러다임간 결합은 모듈 단위의 편집, 컴파일, 링킹을 가능하게 한다[12,13]. 문장과 연산 수준에서 패러다임간의 결합은 강력한 프로그래밍 도구를 제공할지 모르지만 패러다임간에 사용되는 표기법 및 문법이 다름으로 해서 많은 혼란을 야기할 수 있다. 따라서, 멀티패러다임 시각 프로그래밍에서는 모듈 단위의 결합이 가장 적절하다. 하나의 모듈을 하나의 패러다임으로 표현함으로서 패러다임간의 독립성과 협력성이 조화롭게 이루어질 수 있다.

그밖에 객체지향 패러다임은 재사용성과 확장성 등에서 많은 장점[19]을 가지고 있기 때문에 멀티패러다임 시각 프로그래밍 시스템이 객체지향 패러다임을 지원하면 바람직하겠다.

패러다임을 구조적으로 결합하기 위한 연구에서 패러다임을 객체 클래스로 인식하고 상속을 통해서 패러다임을 결합하는 연구가 Diomidis[12,13]에 의해 이루어졌다. 그러나 이러한 관점보다 패러다임을 클래스의 메소드로 보는 관점이 더 타당하다. 객체지향 프로그래밍의 가장 중심이 되는 클래스는 어트리뷰트와 메소드의 집합으로 되어 있다. 이 클래스의 메소드들이 각각 하나의 패러다임으로 표현된다면 전체적으로는 객체지향을 지원하고, 클래스내부에서는 멀티패러다임을 지원하게 된다. 그림 1은 MVPE 모델에서 클래스의 메소드는 다양한 패러다임으로 표현될 수 있음을 보여준다.

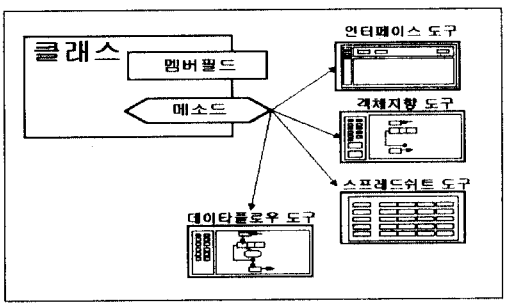

그림 1. MVPE 모델

MVPE는 4개의 프로그래밍 패러다임을 지원하며, 각 패러다임마다 프로그래밍 도구를 두어서 패러다임 고유의 제어 구조나 연산자를 사용할 수 있다. 따라서 MVPE는 멀티패러다임 언어의 요구 사항을 충족시키면서, 새로운 패러다임 도구를 쉽게 추가해서 확장할 수 있게 구성되었다.

이렇게 멀티패러다임 시스템을 구축하는 경우에 다른 결합 방법과 마찬가지로 여러 가지 문제점들이 발생할 수 있다. 이러한 문제점들 중에서 특히 고려할 사항들로는 다음과 같은 것들이 있다.

- 어떠한 패러다임을 결합할 것인가
- 패러다임간의 자료 교환
- 패러다임간의 제어 이동
- 자원 관리
- 실행 모델의 차이

첫 번째로 고려할 사항은 어떤 패러다임들을 결합할 것인가 하는 문제이다. 현재 수많은 프로그래밍 패러다임들이 존재하고, 이론적으로나 실제적으로 각 패러다임들을 서로 결합하는 것이 가능하기 때문에 패러다임 결합 가능성은 수도 없이 많다고 할 수 있다. MVPE시스템은 시각 프로그래밍 분야에서 이제까지 성공적이었다고 평가되거나 앞으로 유망하다고 여겨지는 프로그래밍 패러다임들을 결합하고자 하였다. 시각 객체지향 패러다임은 앞으로 유망한 분야[26]이고, 데이터플로우 패러다임은 시각 프로그램에서 알고리즘을 표현하기 적합하기 때문에 가장 많이 사용되는 패러다임이다[1]. 또한 스프레드쉬트 패러다임은 현재까지 가장 일반적으로 사용되는 시각 프로그래밍 패러다임이다[1]. 직접조작 방식의 인터페이스 개발 도구들은 Visual Basic, Delphi 등의 상용 시스템에서 많이 이용되고 있다. 이렇게 MVPE는 현재까지 가장 성공적이었다고 평가되는 패러다임들을 결합하여 시각 프로그래밍의 한계를 극복하고자 설계되었다.

다음으로 고려할 사항은 패러다임간의 자료 교환이다. 자료 교환은 패러다임간 자료 교환이 이루어 지는 방법과 서로 다른 패러다임에서 지원하는 자료형들이 어떻게 서로 교환될 것인지를 고려해야 한다. 패러다임간의 자료 교환은 Pamela[17]의 연구에 의하면 함수 호출, 스트림, 이벤트에 의해서 이루어질 수 있다. 자료형은 대부분의 패러다임에서 지원하는 정수형, 문자형 등의 자료형과 특정 패러다임에서만 지원하는 배열(임퍼래티브 패러다임), 길이에 제한이 없

는 리스트(함수형 패러다임) 등의 자료형이 있다. 멀티패러다임에서는 자료 교환 방법과 다른 패러다임의 자료형을 지원하는 방법을 고려해야 한다. MVPE에서는 패러다임간의 자료 교환 방법으로 함수 호출 결과와 클래스의 어트리뷰트의 전역변수처럼 사용하는 방법을 사용한다. 교환되는 자료형으로는 MVPE의 기반 언어인 C++에서 지원하는 자료형으로 제한함으로서 시스템 구현을 쉽게 하였다.

세 번째로 고려할 사항은 패러다임간의 제어 이동이다. 패러다임간의 자료 교환과 함께 제어 이동은 패러다임들 간의 협력을 위해 반드시 필요한 부분이다. MVPE에서 패러다임간의 제어 이동은 메소드 호출 방법을 이용하도록 하였다.

패러다임간에 자원을 관리하는 방법이 다르기 때문에 자원의 사용과 관리는 중요한 고려사항 중의 하나이다. 가장 대표적인 자원으로는 변수, 변수의 네임스페이스, 메모리 등을 들 수 있다. 예를 들어, 어떤 프로그래밍 패러다임은 변수를 사용하지만, 다른 패러다임에서는 변수를 사용하지 않을 수 있다. 일반적인 데이터플로우 패러다임과 스프레드쉬트 패러다임은 변수를 사용하지 않지만, MVPE의 데이터플로우 패러다임과 스프레드쉬트 패러다임에서는 변수를 사용하도록 했다.

마지막으로 고려할 대상은 실행 모델이다. 어떤 프로그래밍 패러다임은 컴파일을 통해 실행되는데 반해 다른 패러다임은 인터프리트 방식으로 실행된다. 따라서 멀티패러다임 시스템은 어떤 실행 모델을 지원할 것인가를 고려해야 한다. 스프레드쉬트 패러다임은 원래 인터프리트 방식으로 실행되는데 MVPE 시스템에서 사용되는 스프레드쉬트 패러다임은 인터프리트 되지 않고 컴파일 방식을 지원한다.

IV. MVPE 시스템 구성

MVPE는 그림 2와 같은 구조로 되어있으며 4개의 프로그래밍 도구로 구성되어 있다.

(1) 객체지향 프로그래밍 도구
(2) 데이터플로우 프로그래밍 도구
(3) 스프레드쉬트 프로그래밍 도구
(4) 인터페이스 개발 도구

각 프로그래밍 도구는 하나의 패러다임을 지원하며 사용자의 입력에 따라 시각 문장을 생성한다. 각

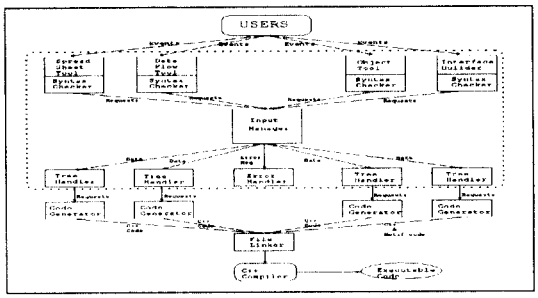

그림 2. MVPE 시스템 구성

도구의 트리 핸들러는 사용자가 기술한 시각 문장을 트리 형태로 관리한다. 각 도구에서 생성된 트리를 이용해 코드 생성기는 사용자의 요구에 따라 C++ 코드를 생성할 수 있다. 생성되는 코드는 파일 단위로 저장되며, 저장된 파일은 C++ 컴파일러와 링커에 의해 컴파일 되고 실행 가능한 코드로 변환된다.

MVPE의 객체지향 도구와 데이터플로우 도구에서 사용되는 시각 문장은 C++의 제어 구조를 변형하여 일반 텍스트 프로그래머가 직감적으로 알아볼 수 있도록 하였다. 예로서, 그림 3은 MVPE에서 사용된 for 문의 시각 문장이다. C++의 문법 구조를 변경하였기 때문에 사용자는 쉽게 알아볼 수 있다.

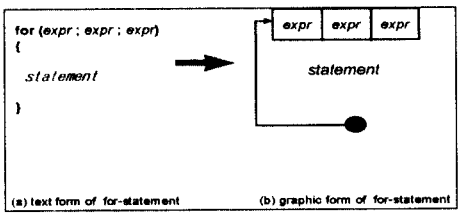

그림 3. for 문의 유사성

MVPE는 많고 다양한 시각 문장을 제공하는 대신에 시각 문장과 텍스트 문장을 동시에 사용하는 것을 허용한다. 사용자는 시각 문장으로 표현하기 어려운 내용을 기술할 필요가 있을 경우에는 시각 문장 중간에 텍스트 문장을 이용할 수 있다.

1. 객체 지향 프로그래밍 도구

객체지향 도구는 객체지향 패러다임을 지원하는

MVPE에서 가장 중심이 되는 도구이다. 객체지향 도구는 클래스 선언 도구와 메소드 정의 도구로 구성된다. 클래스 선언 도구를 이용하여 사용자는 클래스를 정의하고, 정의된 클래스를 원하는 헤더 파일에 저장한다. 그림 4는 클래스 선언 도구를 이용해서 Employee 클래스를 선언하는 예제이다.

클래스 선언 도구는 이미 정의되어 있는 클래스 목록을 사용자에게 보여주고 사용자가 원하는 클래스를 슈퍼 클래스로 정의할 수 있다. 클래스 선언 도구는 클래스의 맴버 필드와 메소드를 private, protected, public 별로 선언할 수 있는 기능을 제공한다. 사용자는 원하는 억세스 컨트롤에 따라 해당되는 텍스트 영역에 입력하면, 클래스 멤버 필드와 메소드가 정의되고 리스트 위짙에 등록된다. 정의된 내용을 삭제하거나 변경하고자 할 경우에는 원하는 내용을 Shift키와 마우스 선택 버튼을 동시에 누르면 리스트 위짙에서는 삭제되고 텍스트 영역에 변경할 내용이 놓여지게 된다. 표 1은 클래스 선언 도구의 오퍼레이션과 사용자의 마우스/키보드 인터랙션과의 매핑관계를 보여준다. 코드 1은 그림 4의 클래스 선언 도구를 이용해 생성된 코드의 일부분이다.

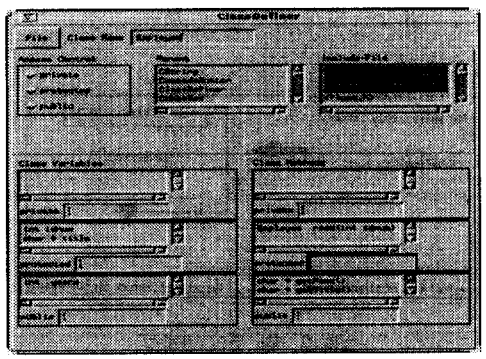

그림 4. 클래스 선언 도구

표 1. 클래스 선언 도구의 인터랙션

오퍼레이션	마우스/키보드 오퍼레이션
변수/메소드 선언	키보드 입력
헤더파일 인클루드	마우스 클릭
부모 클래스 선언	마우스 클릭
변수/메소드 삭제 및 변경	Shift 키 + 마우스 클릭
메소드 정의 도구 액티베이션	메소드명 더블 클릭

```
#ifndef  _EMPLOYEE_H
#define  _EMPLOYEE_H

#include <assert.h>
#include <ctype.h>
#include <fcntl.h>

class  Employee
 {
// Class Member fields

  protected:
        char * name ;
        int  idnum ;
        char * title ;

  public:
        int  years ;

// Class Methods

  protected:
        Employee  read(int idnum) ;

  public:
        int  getSalary(int year) ;
.....
```

코드1. 클래스 선언도구에서 생성한 코드 예

메소드 정의 도구는 시각 문장을 이용하여 알고리즘을 표현한다. 클래스 선언 도구에서 등록된 메소드 이름을 더블 클릭하면, 해당되는 메소드를 정의할 수 있는 메소드 정의 도구가 화면에 나타난다. 메소드 정의 도구는 전역 변수와 메소드 내에서만 사용되는 지역 변수를 정의할 수 있다. 이 곳에서 정의된 변수와 클래스 선언 도구에서 정의된 클래스 멤버 필드 변수들은 드래그-앤-드랍해서 작업 영역에 기록될 수 있다. 이 도구에서 제공하는 시각 문장도 마찬가지로 드래그-앤-드랍해서 사용된다. 작업 영역에 드랍된 시각 문장은 이동, 크기 변경, 삭제, 텍스트 입력 등의 연산이 적용될 수 있다.

그림 5는 메소드 정의 도구를 이용해서 Employee클래스의 print(int id) 메소드를 표현한 예제이다. print (int id) 메소드는 id를 아규먼트로 받아서 "Employee" 파일에서 해당되는 사원의 이름과 근무년수를 읽어오는 프로그램이다. print(int id) 메소드는 읽어온 근무년수를 이용해서 getSalary(int year) 메소드를 호출해서 사원의 급여를 출력한다. 메소드 정의 도구의 왼쪽에는 사용될 수 있는 시각 문장들과 지역/전역 변수들 목록이 나와있다. 표 2는 메소드 정의 도구의 오퍼레이션과 마우스/키보드 오퍼레이션의 매핑관계를 보여준다.

일반적으로 시각 프로그래밍 시스템에서는 주로 문장의 위치 정보와 문장과 문장 사이 관계 정보를 이용하여 파싱한다. MVPE의 메소드 정의 도구에서는 다른 일반 시각 프로그래밍 시스템에서 하는 문법적인 오류 체크나 의미 분석 과정은 생략한다. 이러한 과정은 전적으로 C++컴파일러에 의존적이다. 메소드 정의 도구는 C++코드를 생성하는데 필요한 최소한의 내용만 체크한다. 메소드 도구의 작업 영역 좌측 상단을 X축과 Y축의 원점(0,0)으로 보고, 좌에서 우를 X축, 위에서 아래를 Y축으로 정한다. 메소드 정의 도구는 시각 문장이 시작하는 X축과 Y축 좌표를 보고 좌표 값이 적은 문장이 시간적으로 우선한다고 판단한다. 같은 Y축 상에서는 X값이 작은 문장이 우선하고, 같은 X축 상에서는 Y값이 작은 문장이 우선한다. X축과 Y축에서는 Y축에 우선권이 있다. 따라서 시각 문장 VS_1의 시작 위치 (x_1, y_1)이고, VS_2의 시작 위치 (X_1, Y_1)일 때 $x_1 < X_1$이고 $Y_1 < y_1$이면 시각 문장 VS_2가 VS_1에 대해 우선권을 갖는다. 어떤 시각 문장은 다른 시각 문장을 포함하는 포함 관계를 가질 수도 있다. 그러나, 어떠한 두 문장도 Y축 상에서 서로 교차하지는 않는다. 시각 문장이 서로 교차하는 경우에 메소드 정의 도구는 자동적으로 시각 문장을 이동시켜 교차가 일어나지 않도록 한다.

메소드 정의 도구와 클래스 선언 도구에서 정의된 변수들은 드래그-앤-드랍해서 작업 영역에서 사용될 수 있다. 일반적으로 선언된 변수가 프리미티브 타입인 경우, 변수가 사용되는 형태가 변경되는 경우는 드물다. 그러나 복잡한 자료구조 타입의 변수라든가 클래스의 인스턴스 변수는 단순히 변수의 이름이 사용되지 않고 ".", "->", "[ ]"연산자를 이용하여 필드나 메소드 등을 지칭하기 위해서 사용되는 경우가 많다. 이런 다양한 표현 방법을 MVPE에서는 그래픽으로 표현하지 못하고 있다. 대신에 사용자가 쉽게 이러한 표현을 텍스트 형태로 이용할 수 있는 기능을 제공한다. 사용자가 정의된 변수 이름에 해당되는 부분을 더블 클릭하면, 해당되는 변수의 타입에서 가능한 표기법이 팝업 윈도우로 제공이 된다. 팝업 윈도우에 레이블 위질으로 표현된 표기법은 드래그-앤-드랍으로 작업 영역에서 사용될 수 있다.

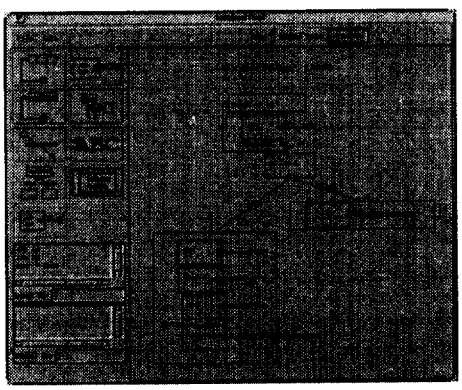

그림 5. 메소드 정의 도구

표 2. 메소드 정의 도구 인터랙션

오퍼레이션	마우스/키보드 오퍼레이션
지역/전역 변수 선언	키보드 입력
시각 문장 사용	드래그-앤-드랍
시각 문장 이동	마우스 선택버튼 드래그
시각 문장 크기 변경	드랍버튼 드래그
시각 문장에서 텍스트 입력	메뉴버튼 클릭
시각 문장 삭제	Shift 키 + 마우스 버튼 클릭

2. 데이터플로우 프로그래밍 도구

데이터플로우 프로그래밍 도구는 함수형 패러다임의 일종인 데이터플로우 프로그래밍 패러다임을 지원한다. 데이터플로우 패러다임은 자료 값이 변경되는 내용을 흐름도를 통해 보여주기 때문에 자료의 변환에 관한 알고리즘을 그래픽으로 자연스럽게 표현할 수 있다. 이런 장점 때문에 현재 대부분의 시각 프로그래밍 시스템들이 데이터플로우 프로그래밍 방식을 지원하고 있다. 데이터플로우 프로그래밍 패러다임은 자료의 흐름을 보여주기 때문에 병렬성을 파악하기 쉽다는 장점도 가지고 있다.

데이터플로우 프로그래밍 도구는 자료의 흐름을 표기하는 화살표만으로 표현할 수도 있겠지만, MVPE에서는 사용자가 자료의 흐름을 이해하고, 자료 흐름을 손쉽게 제어할 수 있도록 몇 가지 시각 문장을 제공한다. 이 도구에서는 자료의 흐름을 화살표로 표현하고, 자료의 변환은 원으로 표현한다. 그림 6은 데이터 플로우 프로그래밍 도구를 이용해서 2개의 숫자를 입력받아 곱셈을 한 뒤에 출력하는 프로그램 예제이다. 표 3은 데이터 플로우 프로그래밍 도구의 오퍼레이션과 마우스/키보드 오퍼레이션의 매핑관계이다.

데이터 플로우 프로그래밍 도구에서는 화살표에 의해서 데이터의 의존성이 결정되기 때문에 화살표의 순서에 따라 시간적인 우선 순위가 결정된다. 화살표의 순서와 함께 데이터, 연산자, 시각 문장의 위치도 순위에 영향을 준다. 위치 정보를 이용하는 경우에는 객체지향 도구에서 사용되는 규칙이 그대로 적용된다. 그러나, 위치 정보와 화살표 정보가 서로 상반될 경우에는 화살표 정보가 우선한다.

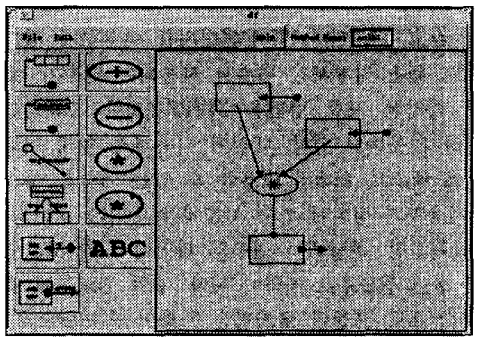

그림 6. 데이터플로우 프로그래밍 도구

표 3. 데이터플로우 프로그래밍 도구 인터랙션

오퍼레이션	마우스/키보드 오퍼레이션
시각 문장 사용	마우스 드래그-앤-드랍
시각 문장 이동	마우스 선택버튼 드래그
시각 문장 크기 변경	마우스 드랩버튼 드래그
시각 문장에서 텍스트 입력	마우스 메뉴버튼 클릭
시각 문장 삭제	Shift 키 + 마우스 버튼 클릭

3. 스프레드쉬트 프로그래밍 도구

스프레드쉬트 프로그래밍은 일반 사용자가 가장 많이 사용하는 시각 프로그래밍의 일종이다[1]. 스프레드쉬트 프로그래밍 도구는 폼베이스드(form-based) 프로그래밍 패러다임을 지원한다. 이 도구는 사용자가 문서에 직접 기입하는 형태를 지원하기 때문에 사용자가 친숙성을 느끼고, 사용하기 편리하다는 장점이 있다. 스프레드쉬트 프로그래밍 도구는 배열 연산, 초기화, 또는 $X = f(Y_1, Y_2, ..., Y_n)$형태의 방정식에 가장 잘 적용이 되기 때문에 이런 형태의 문제를 해결하는데 많은 도움을 줄 수 있다[20].

스프레드쉬트 프로그래밍 도구의 첫 번째 칼럼은 변수를 선언·정의할 수 있는 영역이다. 사용자는 다른 도구에서 정의된 전역 변수나 클래스 멤버 필드를 첫 번째 칼럼에 드래그-앤-드랍해서 스프레드쉬트 프로그래밍 도구에서 사용할 수 있다. 그림 7은 스프레드쉬트 프로그래밍 도구를 이용해서 Employee 클래스의 getSalary(int year) 메소드를 표현하는 예제이다. getSalary(int year) 메소드는 그림 5의 메소드 정의 도구에서 호출되어 사용된다. 그림 7은 다음과 같은 식을 스프레드쉬트 방식으로 프로그래밍한 예이다.

급여 = (기본급 + 근무연수 * 10) − 세금

표 4는 스프레드쉬트 프로그래밍 도구의 오퍼레이션과 마우스/키보드 오퍼레이션의 매핑관계이다.

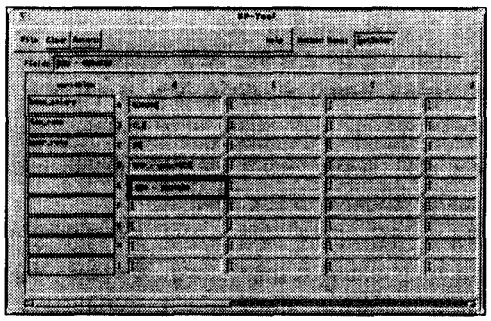

그림 7. 스프레드쉬트 프로그래밍 도구

표 4. 스프레드쉬트 도구 인터랙션

오퍼레이션	마우스/키보드 오퍼레이션
변수 선언	마우스 드래그-앤-드랍
값 입력	키보드 입력

스프레드쉬트 프로그래밍 도구에서 C++ 코드를 생성하기 위한 시간 우선 순위는 칼럼 와이즈(column-wise)형태로 위에서 아래, 좌에서 우로 이동하면서 우선 순위를 갖는다. 현재 스프레드쉬트 프로그래밍 도구에서는 단순한 컴파일 방식만 지원한다. 이 방식은 간단한 수식 계산을 즉시 실행해서 사용자에게 결과를 보여줄 수 있는 인터프리트 방식보다 때론 불편할

수 있지만, MVPE의 다른 도구에서 정의되어 있거나 아직 정의되어 있지 않은 함수를 호출할 수 있다는 장점을 가질 수 있다.

4. 인터페이스 개발 도구

인터페이스 개발 도구는 응용 프로그램에 사용될 그래픽 유저 인터페이스를 만들어 주는 도구이다. 인터페이스 개발 도구는 사용자가 원하는 형태의 인터페이스를 직접 작성할 수 있는 직접 조작(direct manipulation) 방식을 지원한다. 사용자는 원하는 위짙을 선택하고, 작업 영역의 알맞은 위치에 원하는 크기로 인터페이스를 그린다. 그려진 인터페이스는 MVPE에서 라이브러리로 제공하는 Motif 위짙 클래스로 된 코드를 생성한다. Motif 위짙 클래스 라이브러리는 Motif 위짙과 위짙에 필요한 함수들을 캡슐화해서 C++ 클래스로 정의한 라이브러리이다[21,22].

인터페이스 개발 도구에서는 각 위짙들이 놓이는 위치에 따라 위짙들 간의 관계가 결정된다. 작업 영역에서 위짙을 나타내는 사각형은 포함 관계를 보고, 포함하는 사각형을 부모로, 포함되는 사각형을 자식으로 본다. 부모-자식 관계를 보고 개발 도구는 C++ 코드를 생성하게 된다. 시간 우선 순위는 항상 부모가 자식 보다 앞서게 된다. 또한 어느 두 사각형도 서로 교차할 수 없으며, 프리미티브 위짙 클래스는 매니저 위짙 클래스를 포함할 수 없다. 이러한 규칙이 지켜지지 않을 경우에는 자동적으로 오류 메시지 다이얼로그 윈도우가 화면에 나타난다. 인터페이스 개발 도구 사용자는 작업 영역에서 원하는 위짙을 마우스 메뉴 버튼으로 클릭하면, 해당 위짙에서 사용 가능한 리소스 이름과 값이 제공된다. 사용자는 원하는 리소스와 리소스의 값을 선택하기만 하면 된다.

그림 8, 9, 10은 인터페이스 개발도구에서 사용되는 윈도우들이다. 그림 8에서 원하는 인터페이스 위짙을 선택하고, 그림 10에서 Employee 클래스를 이용해서 원하는 사원을 검색하거나 전체 목록을 볼 수 있는 GUI 인터페이스를 인터페이스 개발 도구를 이용해서 기술한 예제이다. 그림 9에서는 그림 10에 그려진 각 인터페이스 컴포넌트의 리소스 값을 설정하기 위한 도구이다.

5. 클래스 정보 뷰어

클래스 정보 뷰어는 사용자가 프로그램 개발 시에

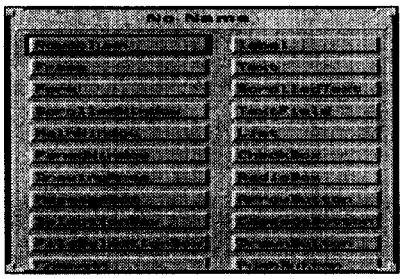

그림 8. 위짙 선택

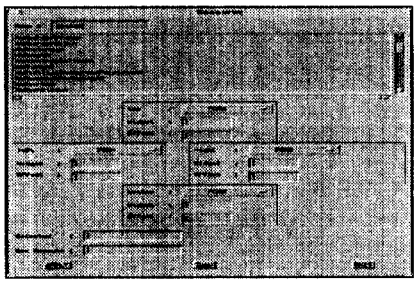

그림 9. 위짙 리소스 설정

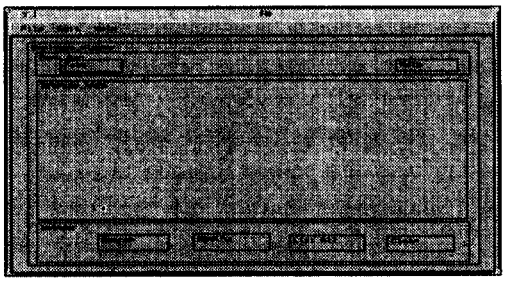

그림 10. 인터페이스 개발 도구

클래스에 관한 정보를 제공하여 개발을 용이하게 해준다. 수많은 클래스가 존재하고, 각 클래스는 또한 많은 멤버 필드와 메소드를 포함하고 있기 때문에 프로그래머는 원하는 멤버 필드의 이름이나 메소드 사용법을 모두 기억하기는 불가능하다. 또한, 객체지향 프로그래밍에서 프로그램을 기술하고, 이해하기 위해서는 클래스 상속 계층을 이해하는 것이 필수적이다[23]. 이런 필요성 때문에 클래스 정보 뷰어는 클래스

상속 계층도, GUI 포함 관계 트리, 클래스 멤버 필드와 메소드 정보를 팝업 윈도우를 이용해 보여준다. 다음 그림 11은 그림 10에서 작성된 GUI 컴포넌트의 포함 관계를 트리형식으로 보여주는 예제이다.

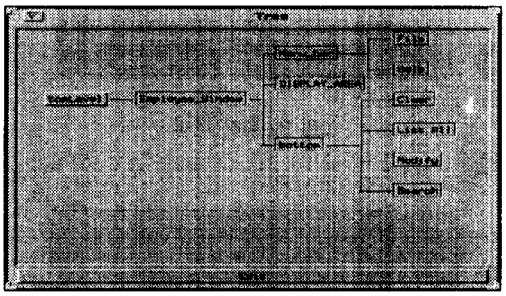

그림 11. Employee_Window의 인터페이스 트리

V. 구현 및 평가

MVPE는 Sun SPARCserver1000, Solaris 2.5 운영체제에서 g++ 2.7.2 컴파일러, Motif 1.2, X11R5 환경에서 객체지향 프로그래밍으로 구현되었다. MVPE에서 사용되는 대부분의 클래스는 Motif의 위젯을 C++ 클래스로 변형한 Motif 위젯 클래스들이다. 많이 사용되는 그래픽 유저 인터페이스 요소들을 하나의 클래스로 정의하면 재사용 면에서 많은 이점을 제공한다.

그림 12는 MVPE를 구현하는데 사용된 Motif 위젯 클래스의 상속 계층 구조를 보여준다. AppEnv는 이 클래스 라이브러리를 이용하는 응용 프로그램을 위한 클래스이다. WiD는 Motif의 위젯에 해당하는 추상 클래스이다. WiD로부터 상속되는 클래스들은 Motif의 각 위젯에 해당되고, 그 위젯에 필요한 기능이 클래스의 메소드로 제공된다. 예를 들면, Motif의 드로잉 위젯에 해당되는 DrawA 클래스는 드로잉 위젯을 만들 수 있는 생성자, 선, 점, 사각형 등을 그리는 메소드, 익스포즈 콜백이 있을 때 다시 그려주는 redraw 메소드 등 많은 함수들을 클래스 메소드로 포함하고 있다.

MVPE 시스템은 제한된 도메인에서만 사용되는 시각 프로그래밍의 단점을 멀티패러다임을 이용해서 시각 프로그래밍의 도메인을 확대하려는 의도에서 설계, 구현되었다. MVPE 시스템은 사용되는 패러다임들을

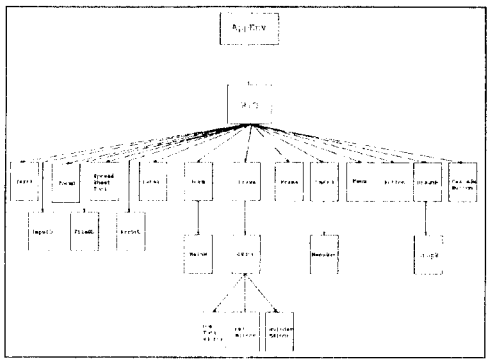

그림 12. 클래스 상속 계층

시각 프로그래밍 분야에서 현재까지 일반적으로 가장 성공적이었다고 평가되는 데이터플로우 패러다임, 객체지향 패러다임, 직접 조작 패러다임, 스프레드쉬트 패러다임을 "클래스의 메소드 = 패러다임"이라는 개념을 바탕으로 결합하였다. "클래스의 메소드 = 패러다임" 모델은 기존에 구조적인 방법이나 모델 없이 패러다임을 결합하던 방식이나 "객체 클래스 = 패러다임"으로 보는 Diomidis[12,13]의 모델보다 프로그래머가 이해하기 쉽고, 자연스럽다.

MVPE에서 멀티패러다임 프로그래밍 방법은 시각 프로그래밍의 문제를 해결하기 위해 "메소드 = 패러다임"이라는 모델을 기반으로 현재까지 성공적인 시각 프로그래밍 패러다임을 결합했다는데 의의를 갖는다. 성공적인 패러다임의 결합으로 MVPE는 시각 프로그래밍의 도메인을 확대하여 더 일반적으로 사용될 수 있다. 그러나 대부분의 시각 프로그래밍 시스템이 토이 시스템(toy system)인 실정에 비추어 MVPE도 아직 실제 프로그래밍 환경에서 사용하기에는 많은 문제점들이 있다. 현재까지 각 패러다임간의 협력이 완벽하게 이루어지고 있지 않고, 패러다임의 결합으로 인해 각 패러다임 고유의 장점을 살리지 못한 경우도 있다. 일례로 스프레드쉬트 패러다임은 인터프리트 되기 때문에 즉석에서 계산 결과를 알 수 있는데 MVPE에서는 패러다임의 결합을 위해 컴파일 방식으로 변환하였기 때문에 스프레드쉬트의 장점을 최대한 살리지 못했다. 앞으로 이러한 문제들은 해결해 나가면서 더욱 세분화된 패러다임을 결합하는 연구를 수행해야 하겠다.

VI. 결 론

MVPE는 "클래스 메소드 = 패러다임"이라는 개념 모델을 바탕으로 객체지향, 데이터플로우, 스프레드쉬트, 직접조작 패러다임을 결합한 멀티패러다임 시각 프로그래밍 환경이다. 일반적으로 시각 프로그래밍 시스템은 변수나 자료 표현과 알고리즘 표현에서 많은 어려움을 겪고 있다. MVPE는 이런 단점을 극복하기 위하여 멀티패러다임 프로그래밍 방법을 결합하였다. MVPE는 각 패러다임의 프로그래밍 도구를 이용해서 프로그램을 표현할 수 있으며 각 도구는 서로 협력해서 작업할 수 있다.

참 고 문 헌

1. Nan C. Shu, *Visual Programming*, Van Nostrand Reinhold Com., 1988.
2. Brad A. Mayers, "Visual Programming, Programming by Example, and Program Visualization: A Taxonomy", *Conference Proceedings, CHI'86: Human Factors in Computing Systems*, pp.59-66, ACM, Sep., 1986.
3. Georg Raeder, "A Survey of Current Graphical Programming Techniques", *Computer*, IEEE, pp. 11-25, Aug., 1985.
4. P. T. Cox, F. R. Giles and T. PieTrzykowski, "Prograph" in *Visual Object-Oriented Programming*, Manning Pub., pp.45-66, 1995.
5. Ed Baroth, Chris Hartsough, "Visual Programming in the Real World" in *Visual Object-Oriented Programming*, Manning Pub., pp.21-42, 1995.
6. Rajeev K. Pandey, Margaret M. Burnett, "Is it Easier to write Matrix Manipulation Programs Visually or Textually ? An Empirical Study" in *Symposium on Visual Languages*, IEEE, pp.334-351, 1993.
7. Ephraim P. Glinert, "Nontexual Programming Environments" in *Principles of Visual Programming Systems*, Prentice-Hall, pp.144-230, 1990.
8. Wayne Citrin, Michael Doherty, Benjamin Zorn, "The Design of a Completely Visual OOP Language" in *Visual Object-Oriented Programming*, Manning Pub., pp.67-93, 1995.
9. Ed Baroth, Chris Hartsough, "Visual Programming in the Real World" in *Visual Object-Oriented Programming*, Manning Pub., pp.21-42, 1995.
10. 김명호, "다중 패러다임과 Leda 언어", *정보과학회지*, 15권, 제1호, pp.37-44, 한국정보과학회, 1997.
11. Jose' A. Borges, Ralph E. Johnson, "Multiparadigm Visual Programming Language" in *Workshop on Visual Languages*, pp. 233-240, IEEE, Oct., 1990.
12. Diomidis D. Spinellis, *Programming Paradigms as Object Classes:A Structuring Mechanism for Multiparadigm Programming*, Ph.D. Thesis, Dept. of Computing, Imperial College of Science, Technology and Medicine, University of London, Feb. 1994.
13. Diomidis Spinellis, Sophia Drossopoulou, Susan Eisenbach, "Language and Architecture Paradigms as Object Classes: A Unified Approach Towards Multiparadigm Programming", *Programming Languages and System Architectures International Conferences*, LNCS 782, Springer Verlag., pp. 191-207, 1994.
14. Stefan Schiffer, Joachnn Hans Fröhlich, "Concepts and Architecture of Vista - a Multiparadigm Programming Environment", in *Workshop on Visual Languages*, pp.40-47, IEEE, 1994.
15. Wilfred J. Hansen, "Andrew as a Multiparadigm Environment for Visual Languages", in *Workshop on Visual Languages*, pp.256-260, IEEE, 1993.
16. Timothy A. Budd, *Multiparadigm programming in LEDA*, Addison Wesley, 1995.
17. Pamela Zave, "A Compositional Approach to Multiparadigm Programming", *Software*, IEEE, Sep. 1989.
18. Brent Hailpern, "Multiparadigm Languages and Environments", *Software*, IEEE, pp.6-9, Jan. 1986.
19. 김수동, "객체지향 소프트웨어 공학", *정보과학회지*, 11권, 제4호, pp.5-21, 한국정보과학회, 1993.
20. Allen L. Ambler, Margaret M. Burnett, Betsy A. Zimmerman, "Operational Versus Definitional: A Perspective on Programming Paradigms", *Computer*, pp.28-43, IEEE, Sep. 1992.

21. Douglas A. Young, *Object-Oriented Programming with C++ and OSF/MOTIF*, Prentice-Hall, 1992.
22. Daniel J. Bernstein, *Using Motif with C++*, SIGS Books Pub., 1995.
23. 김문회, 한재수, "객체지향 언어 C++를 위한 Information Viewer", *정보과학회지*, 11권, 제2호, pp.84-93, 한국정보과학회, 1993.
24. Shriver B. D., "Software Paradigm", *Software*, IEEE, pp.2, Mar. 1986.
25. Zloof M., "Query-by-Example : A Database Language", *IBM Sys. Journal*, 16(4), pp.324-343, 1977.
26. Adele Goldberg, Margaret Burnett, Ted Lewis, "What is Visual Object-Oriented Programming?", in *Visual Object-Oriented Programming*, Manning Pub., pp.3-20, 1995.

유 재 우(Chae-Woo Yoo) 정회원
1976년 : 숭실대학교 전자계산학과 (공학사)
1978년 : 한국과학기술원 전자계산학과(공학석사)
1985년 : 한국과학기술원 전자계산학과(공학박사)
1986년 ~ 1987년 : Cornell Univ.의 Visiting Scientist
1996년 ~ 1997년 : Univ. of Pittsburgh의 Visiting Scientist
1983년 ~ 현재 : 숭실대학교 전자계산학과 교수로 재직중.
※주관심분야 : 컴파일러, 프로그래밍 환경, 인간과 컴퓨터 상호작용

최 종 명(Jong-Myung Choi) 정회원
1992년 : 숭실대학교 전자계산학과 (공학사)
1996년 : 숭실대학교 전자계산학과 (공학석사)
1997년 ~ 현재 : 숭실대학교 전자계산학과 박사과정
※주관심분야 : 컴파일러, 시각프로그래밍, 멀티패러다임 프로그래밍