스프레드시트에서 데이터 플로우 프로그래밍 적용 기법 연구

최종명*, 김기원**, 양진영***

요약

스프레드시트와 데이터 플로우는 시각 프로그래밍 분야에서 널리 사용되는 프로그래밍 패러다임이지만, 두 개의 패러다임을 결합해서 두 패러다임의 장점을 얻으려는 연구는 상대적으로 적었다. 본 논문에서는 스프레드시트에 객체지향과 데이터 플로우 패러다임을 결합함으로써 스프레드시트를 좀 더 쉽고, 효율적으로 개발할 수 있는 방법을 소개한다. 또한 객체지향 스프레드시트에서 데이터 플로우를 적용하기 위한 방법과 데이터 플로우에서 액션 형태를 분류해서 프로그래밍에 적용하는 것을 보여준다.

Study on Applying Data Flow Programming to Spreadsheet

Jong-Myung Choi*, Gi-Weon Kim**, Jin-Young Yang***

ABSTRACT

There have been two popular programming paradigms in visual programming field: spreadsheet and data flow programming. However, there have been little effort to combine these two programming paradigms to support easy programming. In this paper, we introduce the way that enables spreadsheet users to develop their applications easily and efficiently by combining the two programming paradigms into one programming language. Furthermore, we also propose how to apply data flow to spreadsheet and how to classify actions and apply them to programming.

Key Words : Spreadsheet, Dataflow, Visual programming, Object oriented programming, Action

* 목포대학교 컴퓨터공학과
** 초당대학교 컴퓨터과학과
· 제1저자(First Author) : 최종명 · 교신저자(Correspondent Author) : 최종명
· 접수일(2009년 2월 17일), 수정일(1차 : 2010년 3월 19일), 게재확정일(2010년 3월 23일)

Ⅰ. 서 론

스프레드시트는 사용하기 쉽고, 배우기 쉽기 때문에 간단한 계산에서부터 분석, 통계, 모델링 등의 다양한 분야에서 널리 사용되는 프로그래밍 도구이다. 스프레드시트는 시각 프로그래밍 분야에서 가장 성공적인 예로 인식되고 있으며, Forms/3[1]와 같은 많은 시각 프로그래밍 시스템이 연구되고 있다. 그러나 스프레드시트는 셀들간의 복잡한 의존관계 때문에 프로그램을 이해하기 어렵고, 에러를 찾기 어렵다는 문제점이 있다.

시각 프로그래밍 분야에서는 스프레드시트이외에 데이터 플로우 패러다임을 이용한 많은 시스템들이 개발되고 있다[2]. 데이터 플로우 패러다임은 데이터 흐름을 시각적으로 보여주기 때문에 직관적이고 이해하기 쉽다는 장점이 있다[3].

스프레드시트와 데이터 플로우 패러다임은 시각 프로그래밍에서 가장 대표적인 프로그래밍 패러다임이지만, 이 두 패러다임을 결합해서 사용해보려는 연구는 상대적으로 적었다. 본 논문에서는 스프레드시트 패러다임, 데이터 플로우 패러다임, 객체지향 패러다임을 결합하는 방법을 소개할 것이다.

스프레드시트는 데이터들 간에 복잡한 의존 관계를 가지고 있으며, 이러한 의존 관계는 데이터 플로우로 표현할 수 있다. 스프레드시트에서 데이터 플로우는 프로그램을 이해하기 쉽게 하고, 개발 노력을 감소시켜줄 수 있다. 또한 스프레드시트에서 객체지향 프로그래밍은 프로그램의 재사용 및 유지 보수성을 높여줄 것이다.

본 논문은 2장에서 관련 연구들을 소개하고, 3장에서 데이터 플로우 패러다임을 결합한 스프레드시트를 소개한다. 4장에서는 결론 및 향후 연구 과제를 밝힌다.

Ⅱ. 관련연구

2.1 Spreadsheet 2000

Spreadsheet 2000[4]은 객체 기반(object-based)의 데이터 플로우를 지원하는 시각 프로그래밍 언어이다. Spreadsheet 2000은 Prograph[5]를 이용해서 작성되었으며, Prograph의 영향을 받아 스프레드시트에 데이터 플로우 패러다임을 적용하고 있다.

Spreadsheet 2000은 Grid, Operator, Chart, Note 라는 4가지 종류의 객체 클래스들을 가지고 있다.

Grid는 셀들의 집합으로 보통 숫자 값을 가지고 있으며, 문자열의 라벨을 추가할 수 있다. Grid는 공식을 가지지 않는다.

Operator는 Grid에 포함된 데이터를 처리하는 객체이다. Operator는 데이터가 연결되는 방향에 따라 행 혹은 열 방향으로 연산을 수행한다. Chart는 데이터를 차트로 표현하기 위한 객체이다. Note는 주석을 기술할 수 있는 객체이다.

그림 1은 Spreadsheet 2000을 이용해서 계산을 수행하는 프로그램 예이다.

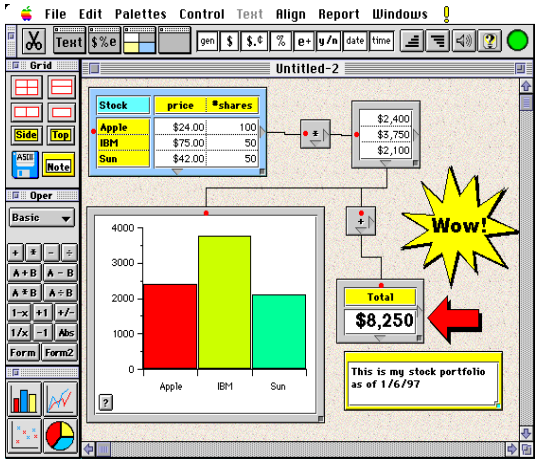

그림 1. 스프래드시트 2000
Fig. 1. Spreadsheet 2000

2.2 객체지향 데이터 플로우

객체지향 데이터 플로우[6]에서 각 노드는 객체 혹은 객체들의 집합을 의미하고, 아크는 메시지 패싱을 의미한다.

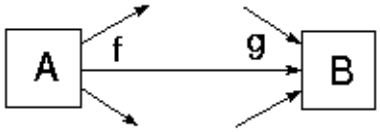

그림 2. 입력라벨과 출력라벨
Fig. 2. Input Label and Output Label

객체는 주위의 다른 객체들로부터 메서드의 매개변수가 모두 채워지면 자신의 메서드를 실행시킨다. 실행 결과는 다시 다른 객체의 메서드의 매개 변수로 이동한다. 따라서, 객체지향 데이터 플로우에서 메서드의 활성화는 다른 객체들과의 상호작용을 의미한다.

각 노드는 객체 이름의 라벨을 가지고 있고, 각 아크는 두 개의 라벨을 가지고 있다. 아크의 양끝에는 시작하는 쪽에는 출력 라벨이 있고, 끝 부분에는 입력 라벨을 가지고 있다.

2.3 액션-객체 플로우

UML[8]의 액션-객체 플로우는 액션을 둥근 사각형으로 표현하고, 객체를 사각형으로 표현한다. 데이터 흐름을 위해서 점선 화살표를 사용하고, 제어 흐름을 표현하기 위해서 실선 화살표를 사용한다.

액션-객체 플로우는 프로그래밍을 표현하는 것이 아니고, 객체지향 분석/설계에 사용된다.

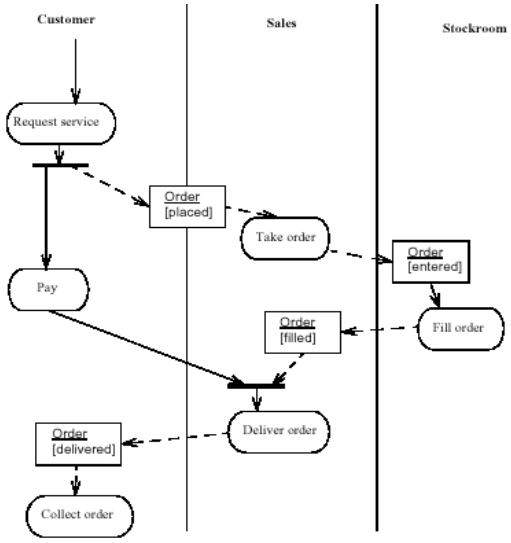

그림 3. 액션-객체 플로우
Fig. 3. Action-object Flow

III. 스프레드시트 객체지향 데이터 플로우

3.1 스프레드시트에 데이터 플로우 도입

스프레드시트는 상수와 공식으로 구성되어 있으며, 공식은 다른 셀의 데이터를 참조해서 계산을 수행한다. 따라서, 스프레드시트 프로그램은 셀들 간의 복잡한 데이터 의존 관계를 가지고 있다. 이러한 복잡한 데이터 의존 관계는 프로그램을 이해하기 어렵게 하고, 오류를 찾기 어렵게 하는 원인이 되고 있다. 이러한 문제를 해결하기 위하여 데이터 셀들 간의 의존 관계를 시각적으로 표현하기 위한 연구들이 진행되어 왔으며, 일부 스프레드시트 시스템에서 지원하고 있다[8].

그림 4는 MS 엑셀에서 셀들 간의 데이터 의존 관계를 화살표로 표현한 것이다.

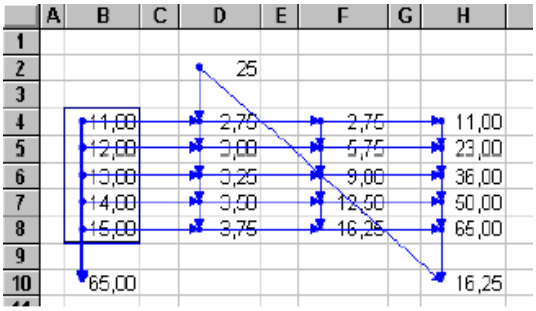

그림 4. 엑셀에서 셀들의 의존관계
Fig. 4. Relationship of Cells in Excel

이러한 시각화는 프로그램의 이해를 도와주기 위한 목적으로 개발되었기 때문에 프로그램 개발에는 도움을 주지 못하고, 셀단위로 시각화가 이루어지기 때문에 매우 복잡하다는 단점이 있다.

스프레드시트에서 표1과 같은 학생들의 성적 데이터들이 있을 때, 스프레드시트를 이용해서 평균을 내림차순으로 정렬하는 작업을 수행한다고 가정해보자. 기존의 스프레드시트를 이용해서 성적을 정렬하려면, 정렬을 원하는 데이터 블록을 선택하고, 정렬 기준이 되는 셀을 결정하고, 정렬 방법을 설정하는 절차로 이루어진다.

표 1. 성적 데이터
Table 1. Score Data

name	math	eng	avg
홍길동	85	90	87.5
김철수	80	70	75

만약 스프레드시트과 데이터 플로우를 결합해서 사용한다면 성적을 정렬하는 프로그램은 그림 5와 같이 표현할 수 있다. 그림 5는 직관적으로 학생의 성적 데이터를 Math 클래스의 sort() 메서드를 이용해서 내림차순으로 정렬한다는 것을 파악할 수 있다.

name	math	eng	avg
홍길동	85	90	87.5
김철수	80	70	75
=Math.sort(self[i].avg, '-')			

그림 5. 학생들의 성적 정렬
Fig. 5. Sorting of Student's Score

전통적인 데이터 플로우 시스템에서 노드는 오퍼레이션을 표현하고, 아크는 데이터 흐름을 표현한다. 즉, 그림 6은 a = f(x, y)를 표현하는 데이터 플로우이다. 따라서 프로그램적으로 데이터 흐름은 함수 호출을 통해서 발생하게 된다.

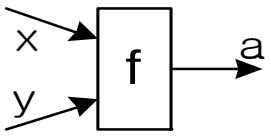

그림 6. 데이터 플로우
Fig. 6. Data Flow

스프레드시트의 데이터들은 서로 연관성을 갖기 때문에 클래스와 객체로 표현될 수 있다. 예를 들어 그림 5의 스프레드시트 데이터는 name, math, eng, avg는 모두 Student라는 클래스의 멤버 필드로 표현될 수 있다. 이렇게 스프레드시트에서 Student 클래스를 정의하는 경우에 스프레드시트의 데이터는 Student 클래스의 인스턴스로 표현될 수 있다. 예를 들어, (홍길동, 85, 90, 87.5) 데이터는 name 멤버 필드의 값이 "홍길동"인 Student 클래스의 인스턴스로 볼 수 있다. 평균값을 구하는 메서드를 정의한다면, avg 속성의 값은 메서드 호출에 의해 자동적으로 계산될 수 있다.

객체지향 프로그래밍에서 데이터의 흐름은 메시지 패싱을 통해서 이루어지고, 데이터 처리 및 변환은 메서드 내에서 이루어진다. 메서드의 매개 변수와 리턴

값은 데이터 흐름의 기본이 된다. 메서드에서 처리된 결과는 다시 다른 메서드의 데이터로 이동된다. 따라서 노드는 메서드를 이용해서 표현된다. 그러나 객체지향 프로그래밍에서 객체는 데이터와 오퍼레이션을 모두 포함하고 있기 때문에 객체도 노드로 표현할 수 있다.

객체지향 데이터 플로우에서 노드는 객체, 객체의 집합, 객체의 메서드를 의미한다. 아크는 노드와 노드 사이의 데이터의 흐름을 표시한다.

노드는 사각형으로 표현하고, 아크는 화살표로 나타낸다. 아크에는 데이터 흐름을 제어할 수 있는 조건을 기술할 수 있다. 조건을 기술하는 경우에는 조건이 참인 경우에만 데이터 흐름이 발생한다. 또한 노드의 내용이 다른 노드의 메서드로 이동할 때 매개 변수의 순서를 위해서 아크에 #num를 기술할 수 있다.

그림 7. 아크
Fig. 7. Arc

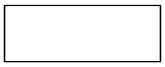

그림 8. 노드
Fig. 8. Node

다음과 같이 연속적으로 아크가 사용되는 경우에 A 객체가 B 객체의 b() 메서드의 데이터로 이동된다. b() 메서드가 수행된 결과는 다시 C 객체의 c() 메서드로 데이터가 흘러간다.

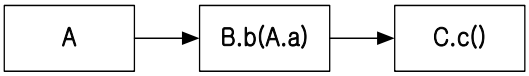

그림 9. 데이터 플로우
Fig. 9. Data Flow

그림 9는 객체지향 프로그래밍 언어로 표현하면 다음과 같이 표현할 수 있다.

C.c(B.b(A.a))

객체 혹은 메서드가 노드로 표현되기 때문에 아크의 시작점과 끝점의 노드 타입에 따라 네 가지 경우로 분류할 수 있다.

3.2 아크 시작점과 끝점이 모두 객체인 경우

아크의 시작점과 끝점이 모두 객체인 경우에 시작점의 객체 데이터는 끝점의 객체로 이동한다. 이러한 예는 아크의 시작점이 파일 객체이고, 끝점이 GUI의 텍스트 영역인 경우를 들 수 있다. 이것은 파일의 내용을 텍스트 영역으로 읽어오는 것을 표현한다. 이러한 경우에 데이터 소스에서 데이터를 추출하는 과정과 데이터 목적지에 데이터를 넘겨주기 위한 내용을 기술할 필요가 있다. 이러한 목적을 위해서 데이터 소스에 export와 데이터 목적지에 import를 두어서 데이터 흐름에 필요한 사항들을 기술할 수 있도록 한다.

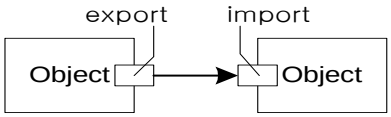

그림 10. 노드가 모두 객체인 경우
Fig. 10. Both Nodes are Objects

3.3 아크 시작점이 객체이고 끝점이 액션인 경우

아크 시작점이 객체이고, 끝점이 액션인 경우에는 객체의 데이터가 액션의 매개 변수로 사용되거나, 액션의 처리 과정에서 사용된다. 이때에는 액션 부분에 사용될 객체의 데이터와 액션의 메서드 이름이 기술되기 때문에 별도의 export와 import가 불필요하다.

이러한 예는 학생들 성적 데이터가 정렬 함수로 이동하는 경우를 들 수 있다.

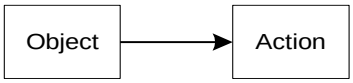

그림 11. 시작점은 객체이고 끝점이 액션인 경우
Fig. 11. Start point is Object and end point is Action

3.4 아크 시작점이 액션이고 끝점이 객체인 경우

아크의 시작점이 액션이고, 끝점이 객체인 경우에는 액션의 처리 결과가 객체의 메서드로 이동하는 것이다. 이 경우에 데이터 목적지는 데이터를 어떻게 받아들일 것인지는 기술하기 위해서 import를 기술할 필요가 있다. 이러한 예는 데이터베이스 검색 결과를 텍스트 영역으로 이동하는 경우를 들 수 있다.

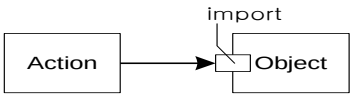

그림 12. 시작점은 액션이고 끝점이 객체인 경우
Fig. 12. Start point is Action and end point is Object

3.5 아크 시작점과 끝점이 모두 액션인 경우

아크의 시작점과 끝점이 모두 액션인 경우에는 데이터 목적지에서 객체, 메서드, 데이터 이름을 모두 기술할 수 있기 때문에 export와 import를 기술할 필요가 없다. 이러한 예는 데이터베이스 검색 결과를 정렬하는 경우를 들 수 있다.

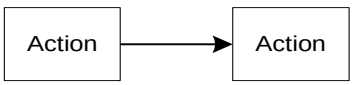

그림 13. 시작점과 끝점이 액션인 경우
Fig. 13. Start point and end point are Actions

Ⅳ. 데이터 플로우에서 액션의 형태

데이터 플로우에서 액션은 데이터를 생성하거나, 처리 및 소비하는 작업을 수행한다. 액션은 데이터를 처리하는 방식에 따라 세 가지 형태로 분류할 수 있다.

4.1 데이터 생성 액션

데이터 생성 액션은 외부로부터 데이터를 받아들이지 않고, 데이터를 생성하는 오퍼레이션이다. 대표적인 예로는 난수 발생하는 함수나 데이터베이스에서 검색하는 함수를 예로 들 수 있다. 그림 14의 DB.sql() 메서드는 데이터베이스 검색을 통해서 데이터를 생성하는 오퍼레이션의 예이다.

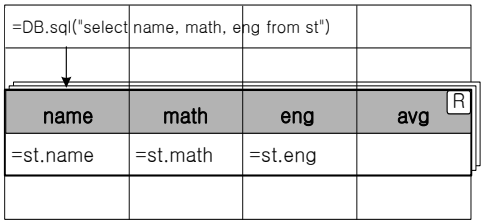

그림 14. 데이터베이스에서 검색
Fig. 14. Retrieval in the Database

그림 14에서 DB.sql() 메서드의 결과는 Student 클래스로 이동한다. 이 경우에 DB.sql() 메서드의 결과 데이터는 Student 클래스의 생성자를 호출해서 Student의 인스턴스들을 생성한다.

4.2 데이터 처리 액션

데이터 처리 액션은 데이터를 입력받아서, 데이터를 처리하고, 결과 데이터를 전달하는 오퍼레이션이다. 데이터 처리 액션은 입력과 출력 데이터 수에 따라 다음과 같이 분류될 수 있다.

1) N x M

N개의 데이터를 입력받아 M개의 결과를 생성하는 데이터 처리 액션이다. 대표적인 예로는 두 개의 행렬을 입력받아 행렬의 곱을 계산하는 경우이다.

2) N x N

N개의 데이터를 입력받아 N개의 결과를 생성하는 데이터 처리 액션이다. 대표적인 예로는 다음 그림과 같이 학생들의 성적을 입력받아서, 정렬하는 경우이다. 이러한 예는 그림 15와 같이 학생 수 N명에 해당하는 데이터를 받아서 정렬하고, 정렬된 N 명의 성적을 전달하는 경우를 들 수 있다. 그림 15에서 Math.sort() 메서드는 N x N 형태의 데이터 처리 액션이다.

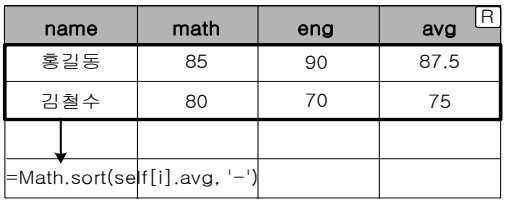

그림 15. 학생 성적 정렬
Fig. 15. Sort of Students' Score

3) N x 1

N개의 데이터를 입력받아 1개의 결과를 생성하는 데이터 처리 액션이다. 대표적인 예로는 그림 16과 같이 학생들의 성적에서 전체 평균을 구하는 경우를 들 수 있다. 그림 16에서 Math.average() 메서드는 N x 1 형태의 데이터 처리 액션이다.

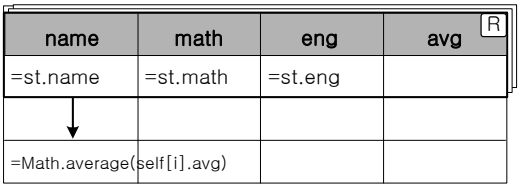

그림 16. 평균 구하기
Fig. 16. Average Calculation

4) 1 x N

1개의 데이터를 입력받아 N개의 결과를 생성하는 데이터 처리 액션이다. 대표적인 예로는 하나의 데이터를 여러 곳에 복사하는 경우를 들 수 있다.

4.3 데이터 소비 액션

데이터 소비 액션은 데이터 입력을 받아서, 처리하고, 결과 데이터를 전달하지 않는 오퍼레이션이다. 예를 들면, 학생들의 성적을 바 차트로 표현하는 것을 들 수 있다.

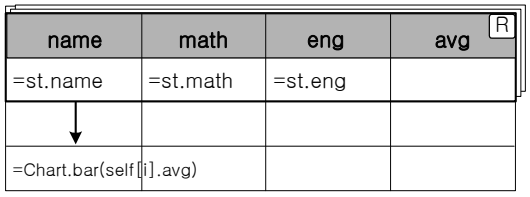

그림 17. 성적을 차트로 변환
Fig. 17.Conversion to Chart from Data

이 경우에 데이터를 입력 받지만 바 차트로 출력하고, 출력된 바 차트를 다른 오퍼레이션에서 사용할 수 없기 때문에 데이터 소비 액션으로 볼 수 있다.

V. 결 론

스프레드시트는 배우기 쉽고, 사용하기 쉬운 프로그래밍 도구이기 때문에 시각 프로그래밍 분야에서 널리 사용되고 연구되고 있다. 그러나 공식을 사용함으로서 발생하는 복잡한 데이터 의존 관계는 프로그램을 이해하기 어렵게 하고, 유지 보수하기 어려운 문제점을 가지고 있다. 이러한 데이터 의존관계를 시각적으로 표현하기 위한 연구가 활발하게 진행되어 왔다.

그러나 이러한 의존관계를 프로그램 이해가 아닌 프로그램 개발에 적용하려는 시도는 상대적으로 적었다.

데이터들의 의존관계는 데이터 플로우로 표현될 수 있으며, 데이터 플로우로 표현하는 경우에 스프레드시트에 데이터 플로우 패러다임을 적용할 수 있다. 데이터 플로우 패러다임은 데이터 흐름을 보여주기 때문에 프로그램을 이해하기 쉽도록 하는 장점이 있고,

이러한 장점 때문에 시각 프로그래밍 분야에서 널리 사용된다.

본 논문에서는 스프레드시트에서 데이터 플로우 패러다임을 적용함으로서 프로그램을 좀더 쉽게 개발할 수 있는 방법을 소개하였다. 또한 스프레드시트와 데이터 플로우를 결합하기 위해서 객체지향에서 데이터 플로우를 적용하기 위한 방법과 데이터 플로우에서 사용할 수 있는 액션 타입들을 분류하였다. 스프레드시트에서 데이터 플로우의 결합은 프로그램을 쉽게 개발할 수 있도록 할뿐만 아니라, 쉽게 이해할 수 있도록 하는 장점이 있다. 향후 연구로는 스프레드시트를 GUI 개발 도구 및 범용 프로그래밍 도구로 확장하는 연구를 수행할 것이다.

참고문헌

[1] M. Burnett and A. Ambler, "*Interactive Visual Data Abstraction in a Declarative Visual Programming Language*", in Journal of Visual Languages and Computing, pp.29-60, Mar. 1994.

[2] Hils, D. D., "*Visual Languages and Computing Survey: Data Flow Visual Programming Languages*," in Journal of Visual Languages and Computing, pp. 69-101, Mar. 1992.

[3] T. Kimura, "*Object-Oriented Dataflow*", in P*roc. 1*1th IEEE Symposium on Visual Languages(VL'95), pp. 180-186, 1995.

[4] Casady & Greene, Inc. Spreadsheet 2000, Online HTML document, 1997. available http://www.emer.com/s2k/

[5] P. T. Cox, F. R. Giles, T. Pietrzykowski, "*Prograph*", in Visual Object-Oriented Programming, Manning Pub. pp. 45-66, 1995.

[6] Lee Braine, Chris Clark, "*Object-Flow*", in Proc. 13th IEEE Symposium on Visual Languages (VL'97*)*, pp. 422-423, 1997.

[7] *OMG Unified Modeling Language Specification v1.3*, http://www.omg.org/uml/

[8] Takeo Igarashi, Jock D. Mackinlay, Bay-Wei Chang, Polle T. Zellweger, "*Fluid Visualization of Spreadsheet Structures*", in Proc. 14th IEEE Symposium on Visual Languages, pp.118-125, 1998.

[9] 이종찬, "패턴 분류를 위한 결정적 아다부스트 알고리즘", *한국지식정보기술학회 논문지*, 제4권 제1호, pp.43-50, 2009.

[9] 이종찬, "패턴 분류를 위한 결정적 아다부스트 알고리즘", *한국지식정보기술학회 논문지*, 제4권 제1호, pp.43-50, 2009.

최종명 (Jong-myung Choi)

1996년 숭실대학교 전자계산학과 (공학석사)
2003년 숭실대학교 전자계산학과 (공학석사)

2004년~현재 목포대학교 컴퓨터공학전공 교수
※ 관심분야 : 프로그래밍 언어, 유비쿼터스 컴퓨팅, 상황-인지 시스템

김기원(Gi-Weon Kim)

1987년 한남대학교 전자계산학과(이학사)
1989년 숭실대학교 컴퓨터공학과(공학석사)
2001년 한남대학교 컴퓨터공학과(공학박사)

1996년~현재 초당대학교 컴퓨터과학과 교수
※ 관심분야 : 멀티미디어, 실시간 영상처리, 음성인식

양진영(Jin-Young Yang)

1983년 조선대학교 경영학과(경영학사)
1988년 조선대학교 전자계산학과(공학석사)
2002년 목포대학교 컴퓨터공학과(공학박사)

1997년~현재 초당대학교 컴퓨터과학과 교수
※ 관심분야 : TCP/IP, Traffic Control, MMI