Proceedings of the 11th
KACC, October 1996

객체지향형 제어 시스템 디자인 패키지의 개발
Development of Object Oriented Computer Aided Control Systems Design Package

°양광웅*, 박재현*

*인하대학교 자동화공학과(Tel: (032)860-7713; Fax:(032)873-4386; E-mail: {ygkgwg,jhyun}@rcsl.inha.ac.kr)

Abstract Object-oriented programming goes on increasing in many areas, as its advantages of flexibility and ease of maintenance have been recognized. As in usual programmings, the productivity and flexibility of CACSD package can be improved by adopting object-oriented programming. This paper describes our efforts to implement an OO-CACSD(object Oriented CACSD) package for control system design and simulation. Since the proposed OO-CACSD is based on the modularity, portability, resualbility, and matrix-oriented data structure, a control system can be not only modeled and simulated but also maintained easily.

Keywords Computer-Aided Control System Design, Object-oriented Programming, Control System, Computer Simulation, Modeling

1. 서론

현대의 산업 기기 들은 점점 복잡하고 다양하게 발전한다. 그리고 기존의 시스템을 새로 개발하거나 신기술을 적용하는 주기가 점점 짧아지므로 시스템의 개발시간을 단축하고 안정성과 성능을 미리 분석하고 검증해 주는 제어 시스템 디자인 (CACSD: Computer-Aided Control System Design) 패키지가 널리 사용된다. 하지만 복잡하고 다양한 시스템을 디자인하는데 있어서 한가지 종류의 CACSD 패키지로는 사용자들의 다양한 욕구를 만족시키지 못한다. 그래서 패키지들은 적용분야에 따라 다양한 형태로 개발되어 있다. 이로 인해 각각의 패키지간에 일관된 프로그래밍 방식이 존재하지 않고 패키지별로 다양한 개발 환경이 사용되고 있다[8]. 현재 시스템 디자인에 주로 사용되고 있는 CACSD 패키지들을 살펴보면 CTRL-C(Systems Control Technology), EASY5(Boeing Computer Services Company), MATLAB(Math Works Inc.), MATRIXx(Integrated Systems Inc.), Simmon(SSPA Systems) 등이 있다.

CACSD 패키지들 중에서 MATLAB은 일반인들에게도 많이 알려져 있는 패키지이다. MATLAB은 행렬(Matrix)을 기본적인 자료의 단위로 처리하고, M-파일(macro file)이라는 구조를 사용하여 하나의 함수를 하나의 M-파일로 구현한다[6]. MATLAB을 이용해서 시스템을 시뮬레이션 하거나 분석하는 프로그램을 작성한다면 위와 같은 특성으로 인해 함수간의 자료 공유에는 전역 변수를 통해서 이루어지고 프로그램 코드에는 전역 변수를 직접 접근하는 부분이 포함되어진다. 시스템을 구성하는 각각의 모듈들이 함수별로 나누어져 잘 모델링 되었다고 하더라도 함수들은 전역 변수로 긴밀하게 결합되어 있기 때문에, MATLAB으로 작성된 모듈(함수)을 다른 시스템을 모델링 하는 모듈에서 이용한다는 것은 쉽지가 않다.

대부분의 시뮬레이션 패키지들은 실세계의 시스템을 시뮬레이션 하거나 분석할 때 MATLAB과 같이 함수를 기반으로 어떻게 작업이 진행되는지에 따라 모듈화 되고, 각각의 모듈은 순차적 프로그래밍 (SPP:Structured Procedual Programming) 방식으로 작성된다. 함수를 기반으로 모듈화 된 프로그램은 논리적인 요소들과 이들의 흐름을 위주로 하여 순차적으로 작성되므로, 이렇게 작성된 프로그램은 실제의 시스템을 표현하기가 어려우며 실제의 시스템과 전혀 다른 개념을 도입해야 할 때도 있다. 그리고 일단 프로그램이 개발되었다 하더라도 모델링 한 시스템에 새로운 요구가 추가되거나 처리할 대상이 변경된다거나 하는 등으로 인해 차후에도 수정할 부분이 많이 생기게 된다. 이런경우 한번 작성한 코드에 새로운 기능을 추가하거나 기존의 기능을 수정하여 보완할 경우 해당 부분만 고쳐도 프로그램이 이전처럼 잘 돌아가는 것이 가장 이상적이다. 하지만 SPP 방법의 언어서는 모듈간에 처리할 대상의 분리가 제대로 이루어지지 않고 전역 변수(Global Variable)를 많이 이용하기 때문에 모듈간의 결합도가 높아지게 된다. 특히 다른 모듈의 자료를 직접 접근하는 경우도 발생한다. 이때는 한 모듈이 새로운 기능의 보강을 위해 수정되면 다른 모듈에도 영향이 파급되어 많은 부분을 수정하고 테스트해야 되기때문에 유지 보수하는데 상당한 경비가 소요된다.

그러나 시스템을 모델링 하거나 시뮬레이션 하는 프로그램을 작성할 때 객체지향 프로그래밍(OOP:Object-Oriented Programming) 방법에 따라 시스템을 각각의 기능별로 모듈화하고 이를 바탕으로 프로그램을 작성한 후, 각 모듈별로 만들어진 프로그램을 조립하는 식으로 프로그램을 개발하면 모듈간의 결합도를 낮출 수 있다. 그리고 시스템에 새로운 기능을 추가할 때마다 이미 개발된 프로그램의 전반적인 수정 없이 관련 모듈만의 수정으로도 프로그램을 재 사용할 수 있다[2][5]. 현재의 프로그래밍 표준으로 자리를 잡아가고 있는 OOP 방식의 이식성과 호환성은 여러 가지 면에서 장점을 가지고 있기 때문에, OOP환경을 지원하는 새로운 CACSD 패키지들이 필요하다[4].

본 논문의 목적은 OOP 방식을 CACSD 패키지에 도입한 객체 지향형 제어 시스템 디자인(OO-CACSD: Object Oriented-CACSD) 패키지인 CEMTool(Control Engineering Mathmatics Tool)을 개발하여 사용함으로 코드의 재활용도를 높이고 시스템의 개발 기간을 단축하여 실질적인 생산성을 높이는데 있다. 논문의 전반부에서는 OO-CACSD 패키지 개발의 골격을 이루는 개념들에 대해서 기술하고, 중반부에서는 CEMTool의 구현 방법에 대해서 기술하고, 후반부에서는 CEMTool로 작성된 예제 프로그램을 MATLAB에서 작성한 프로그램과 비교한다.

2. OO-CACSD의 기본 요소

2.1 모듈화 *(Modularity)*

OOP 방법은 설계나 구현의 기본 단위인 객체가 전역 변수를 사용하지 않고 사용할 자료를 모두 객체의 속성으로 가지고 있다. 그리고 내부적으로 사용되는 속성은 해당 모듈 내에서만 접근을 허용하여 다른 모듈이 직접 접근하여 자료를 가공하는 것을 막을 수 있다. 그리고 모듈의 속성들에 대한 접근과 변경은 해당 객체간의 메시지 전달로 이루어 지기 때문에 객체 내서 자료를 외부로부터 숨기는것이 가능하다. 그림 1은 SPP와 OOP의 각 모듈간 상호 작용 방식을

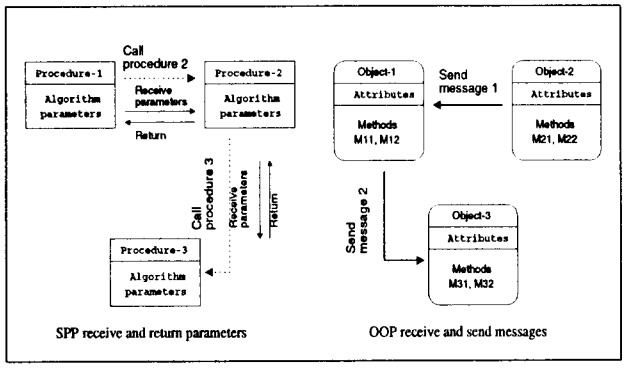

그림 1: SPP와 OOP의 상호 작용 비교.
Fig. 1 Comparison of SPP and OOP approach.

비교한 것이다. 왼쪽 그림은 SPP에서 프로시저가 다른 프로시저를 호출할 때 매개 변수(Parameter)를 통해 자료를 주고받는 구조이다. 오른쪽 그림은 OOP에서 객체간에 메시지를 주고받음으로 자료를 전달하는 구조이다. 오른쪽 그림에서 각각의 객체는 미리 정의된 메시지에 응답하게 된다. 각각의 항목은 객체의 속성이거나 객체이며 이것은 메시지에 의해서 동작한다. 그러므로 객체 사이에서 전달되는 메시지는 동작을 유발하는 능동적인 수행단위가 된다. 이러한 메시지 처리 방식의 OOP 구조는 시스템을 모듈화 하여 분석할 수 있고 전체 소프트트웨어 모듈중 한 부분의 변경에 대해서도 해당 객체 내에서만 영향이 있고 다른 모듈에는 그 영향이 전파되지 않는 장점이 있다.

2.2 이식성 *(Portability)*

대부분의 시뮬레이션 패키지들은 인터프리터 방식으로 소스 코드를 수행한다. 인터프리터 방식은 프로그램을 한 줄씩 해석하여 수행하는 방법으로 수행하는데 상당한 시간을 소비하며 루프를 만나게되면 같은 코드를 여러 번 해석해서 수행해야 하기 때문에 수행속도가 현저하게 떨어진다. 반대로 C++ 같은 컴파일러들이 프로그램을 컴파일 하여 만들어낸 실행코드는 상당히 빠르게 수행되지만 사용자가 C++언어를 배우는데 많은 시간을 보내야 하며 시스템을 디자인하거나 시뮬레이션 하기 위한 라이브러리를 선택하기 어려운 것이 걸림돌이 된다. 그리고 컴파일해서 만들어진 실행코드는 한 종류의 컴퓨터에만 종속되어 수행된다.

하지만 컴파일러가 소스 코드를 컴파일 하여 실행코드 대신에 실행의 최소 단위가 되는 토큰을 만든다면, 만들어진 토큰은 토큰 해석기를 가지고 차례로 수행하면 될 것이다. 컴파일러가 토큰을 만들 때 문법 체크와 토큰의 수행 구조를 최적화 한다면 토큰이 실행될 때는 단지 토큰이 가지고 있는 명령어대로 수행하면 된다. 이로 인해 토큰을 수행하는 토큰 해석기의 부담은 상당히 줄어들게 된다. 그리고 토큰이 실행되어야할 컴퓨터에서 동작하는 토큰 해석기만 있으면 이기종의 컴퓨터에서도 토큰을 수행시킬 수 있다. 소스코드를 다른 종류의 컴퓨터로 옮겨와 새로 컴파일 할 필요가 없을뿐더러 OO-CACSD 패키지를 여러 기종에 맞추어 개발할 필요도 없어진다. 단지 토큰 해석기만 각각의 컴퓨터 기종에 이식 한다면 기계나 기계어의 종류에 무관하게 토큰이 실행 되므로 토큰 수준에서의 이식성과 호환성을 제공해 준어야 한다.

2.3 행렬 중심 *(Matrix-based)*의 자료 구조

대부분의 CACSD 패키지의 자료 표현 방법으로는 MATLAB에서 사용되는 행렬 표기형식이 일반화되어 있다. 그러므로 기존의 CACSD 패키지에 익숙한 사용자들에게 행렬을 객체로 가지는 OO-CACSD 패키지는 기존의 CACSD 패키지 사용자에게 무리한 변화를 요구하지 않고 쉽게 OOP 방식으로 사고를 전환할 수 있도록 한다[8].

2.4 *GUI(Graphic User Interface)* 환경

시스템이 더욱 방대해지고 복잡해질수록 이를 모델링 하거나 시뮬레이션 할 프로그램을 개발한다는 것은 목표로 하는 시스템을 클래스 구조로 얼마나 잘 옮기느냐에 달렸다. OOP 방식을 사용하는 CACSD 패키지로 시스템을 시뮬레이션 하거나 모델링 한다면 당연히 프로그래밍 환경도 객체지향 개념을 지원하고 사용자가 그 개념으로부터 사고를 전개 할 수 있도록 도와주는 사용자 환경이 필요하다. 이러한 OO-CACSD 패키지의 개발을 위해 아래와 같은 방법은 OO-CACSD 패키지의 사용자 환경으로서 가져야할 일반적인 특성이 될 수 있다[3].
클래스의 계층도를 직관적으로 윈도우(Window)에 표현해 주고 부모 클래스(Parent Class)와 자식 클래스(Child Class) 사이의 상속관계를 표시해 줌으로 프로그래머가 클래스 사이의 상속관계를 이해하기 쉽게 하며, 또 그 상태에서 편집 윈도우를 띄워 바로 코드를 편집할 수 있어야 한다. 그리고 지금까지 작성된 클래스를 드래&드롭(Drag&Drop)을 이용하여 쉽게 다른 클래스와 상속 관계를 맺어주는 것이 가능해야 한다.

3. CEMTool 패키지의 구현

CEMTool은 C++ 언어로 작성되었으며 총 프로그램 길이는 약 45,000 라인 정도이고 Windows NT와 Windows 95에서 동작하도록 컴파일 되어 있다. CEMTool은 크게 네 부분으로 나누어져 있다. 기능 별로 자료 입출력 창, 내장 함수 라이브러리, 컴파일러와 토큰 해석기, 내부 자료 구조 선언부로으로 나누어져 있다.

3.1 내부 자료 구조

CEMTool의 내부 자료 구조 중에서 기본 구조는 실수(Real)와 문자열(String) 이다. 실수는 C 언어의 double(8byte) 형과 크기가 같고, 문자열에서 한 문자는 C 언어의 char(1byte) 와 같은 크기를 가지며, 문자들을 일렬로 나열한 구조가 문자열이다. 문자열은 문자의 개수에 따라 크기가 변하므로 크기에 따라 동적으로 메모리가 할당된다. 복소수(Complex)는 실수부(Real Number)와 허수부(Image Number)로 나누어지고 각각 한 개씩의 실수를 참조한다. 행렬의 원소(Element)는 실수, 복소수, 문자열, 행렬 중 하나를 참조한다. MATLAB과는 달리, 행렬이 행렬을 원소로 가지는 것과 같은 자기 참조 자료 구조를 가질 수 있으며, 하나의 행렬 안에 자료 구조가 서로 다른 형태의 원소들도 가질 수 있다. 이러한 자료 구조는 C 언어의 구조체 (struct)와 비슷하고 자료 구조를 용도에 맞게 바꾸거나 확장할 수 있다. 이러한 자료 구조가 가지는 이점은 자료를 외부 파일에서 읽어 오거나 쓸 때 입출력 파일의 구조에 맞추어 행렬의 구조를 만들면 상당히 편리하게 읽고 쓸 수 있다.

자료를 저장하는 공간으로 전역 변수, 지역 변수(Local Variable), 클래스 변수 테이블로 나누어진다(그림 2 참조). 전역 변수 테이블은 프로그램이 시작될 때 할당되어 클래스나 함수의 외부에서 만들어지는 자료를 보관하는데 사용되며, 지역 변수 테이블은 지역 변수를 포함한 함수가 실행될 때 만들어졌다가 함수를 벗어날 때까지 유지된다. 함수를 벗어나면 테이블이 소거되므로 일시적인 자료를 저장한다. 클래스 변수 테이블은 클래스의 속성들을 저장하는데 사용된다.

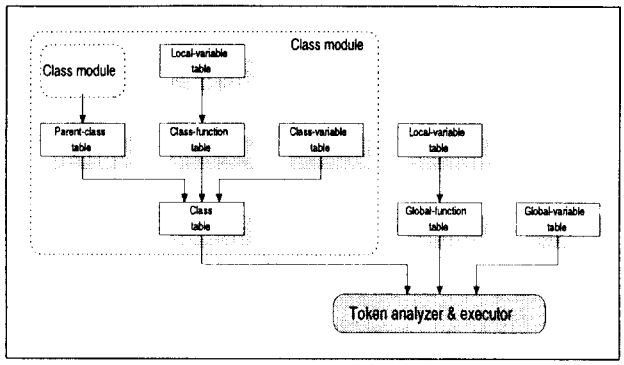

그림 2: CEMTool의 함수, 변수, 클래스 테이블 구조.
Fig. 2 Function, variable, and class table structure of CEMTool.

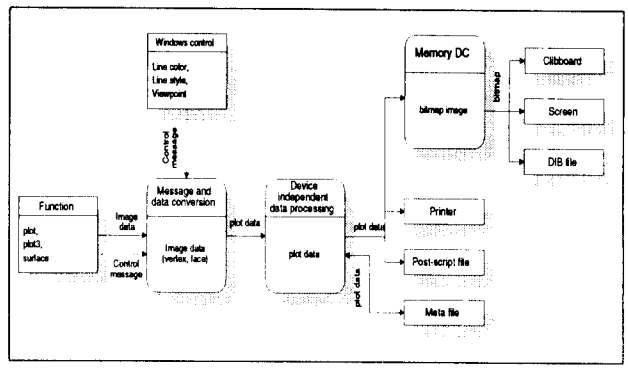

그림 3: 그래픽 장치의 자료 흐름.
Fig. 3 Data flow of Graphic Device.

3.2 내장 함수 라이브러리

사용자가 코드를 작성할 때 기본적으로 필요로 하는 함수들은 내장 함수 형태로 CEMTool의 실행 코드 안에 만들어져 있다. 실수와 복소수의 기본적인 사칙 연산자와 수학 함수를 제공하며, 행렬이나 다항식(Polynomial)에 관한 연산과 조작을 위한 다양한 연산자와 함수들을 제공한다. 또 자료를 문자단위로 입출력하거나 그래프 창에 그래프를 출력하거나 그래프를 조작해주는 함수들이 있다[7]. 내장 함수는 CEMTool이 만들어질 때 컴파일 되어 실행 파일과 같이 링크 되어 있기 때문에 사용자 함수 보다 속도 면에서 빠르다는 장점을 제공한다.

내장 함수는 CEMTool이 시작되면서 내장 함수 포인트를 등록해 두는 테이블에 등록되고, 소스코드가 컴파일 되어 토큰이 만들어질 때 내장 함수를 참조 하는 토큰인 경우 토큰이 내장 함수의 테이블 색인을 가진다. 만들어진 토큰이 토큰 해석기에 의해 수행될 때 내장 함수를 호출하기 위해서 해싱(Hashing)이나 양분법(Bisection method)을 사용하여 테이블을 검색할 필요 없이, 토큰에 있는 색인만으로도 바로 테이블에서 함수 포인트를 얻어오는 것이 가능하기 때문에 토큰과 내장 함수간에 부하가 없는 연계가 가능하다.

3.3 컴파일러와 토큰 해석기

사용자가 작성한 코드가 컴파일러를 거쳐 토큰으로 만들어 지기까지 두 단계의 과정을 거친다[1].

- 구문 분석(Lexical Analysis): 구문 분석기가 코드를 파서가 인식할 수 있도록 문자열이나 기호 단위로 분해하여 토큰으로 만든다.
- 파싱(Parsing): 파서가 토큰의 문법 체크를 하고 토큰을 연산 순위에 따라 재배열하거나 문장 제어 문에 따라 분기되도록 배치시킨다.

CEMTool의 제어 문은 if...elseif...else...end, for...end, while...end, break, continue가 있고, MATLAB의 제어문과 구조가 비슷하다. OOP을 지원하기 위해 추가된 예약어로 class, private, public, protected가 있다.

토큰 해석기(token executor)는 토큰을 기억장치로부터 읽어와 수행시키는 역할을 한다. 이미 컴파일 되어 있는 토큰은 소스코드보다 크기가 작으므로 원격 컴퓨터로 통신을 통해 쉽게 전송 가능하며, 원격 컴퓨터와 지역 컴퓨터가 기계어 수준의 호환이 안될지라도 토큰 해석기가 인스톨된 컴퓨터에서는 토큰을 순차적으로 수행할 수 있다.

지금의 CEMTool은 토큰 해석기가 패키지에 내장되어 있어서 토큰의 이식성과 호환성을 제공하지 못한다. 하지만 추후에 CEMTool을 여러 운영체제와 하드웨어 플랫폼에다 이식시킬 때 토큰 해석기만 CEMTool패키지에서 따로 분리해 낸다면, 하나의 기종에서 만들어진 토큰을 여러 대의 다른 기종으로 구성된 작업 그룹으로 다운로드 하여 수행하는 것이 가능하다.

3.4 입출력 창

CEMTool의 입출력 창은 사용자와 가장 많이 접하게 되는 부분이므로 시스템 디자인 패키지에서 가장 중요한 부분을 차지한다. 무엇보다도 사용자가 배우기 쉽고 사용하기 쉬워야 한다. 입력 창은 기능별로 세 부분으로 나누어진다. 명령어 입력 창, 클래스 편집 창, 코드 작성 창이 그것이다.

- 명령어 입력 창: CEMTool의 명령어 입력 창은 MATLAB의 명령어 입력 창과 같은 역할을 한다. 사용자는 명령어 입력 창에서 대화식(Interactive)으로 명령어를 수행하고 결과를 본다.
- 클래스 편집 창: 클래스 편집 창은 거대하고 복잡한 시스템을 시뮬레이션 하기 위해 무엇보다 중요한 부분이다. 코드의 한 부분을 잘 작성하는 것보다 시스템을 모듈별로 잘 나누어 이들의 유기적인 결합관계를 클래스 위주로 편집할 수 있게 해주는 부분이다.
- 코드 작성 창: 코드 작성 창은 클래스의 속성을 코드로 편집하는 부분이다.

출력 창은 CEMTool에서 처리한 결과를 텍스트와 그래픽으로 보여준다. 메시지 출력 창, 전역 변수 창, 그래프 출력 창으로 나누어져 있다.

- 메시지 출력 창: 메시지 출력 창은 코드를 컴파일 하거나 실행 도중에 발생하는 경고와 오류 메시지를 창에 보여준다.
- 전역 변수 창: 전역 변수 창은 코드를 실행한 후 생성된 전역 변수의 값을 창에 보여준다.
- 그래프 출력 창: 그래프 출력 창은 2차원과 3차원 좌표계에서 선이나 면으로 자료를 창에 그래프로 표시해 주는 역할을 한다. 그림 3는 CEMTool이 그래프를 창이나 프린터에 출력하거나 파일로 저장하는 부분의 구조이다.

reset.m

```
% Reset counter register.
function reset(void)
counter_reg=0;
```

increment.m

```
% Increase counter register.
function result = increment(void)
counter_reg=counter_reg+1;
result=1;
```

decrement.m

```
% Decrease counter register while greater than zero.
function result = decrement(void)
if(counter_reg>0)
   counter_reg=counter_reg-1;
   result=1;
else
   result=0;
end
```

main.m

```
% main routine
counter_reg=0;      % Make variable that used in counter.
reset(0);           % Initialize counter.
increment(0);       % Count one.
```

그림 4: MATLAB으로 작성된 Counter 예제.
Fig. 4 Counter sample in MATLAB.

counter.class

```
private:
   counter_reg=0;
public:
   function reset(void) //Reset counter register.
      counter_reg=0
   end

   function increment(void)  //Increase counter register.
      counter_reg=counter_reg+1
   end

   function decrement(void) //Decrease counter register.
      if(counter_reg>0)
         counter_reg=counter_reg-1
         return 1
      else
         return 0
      end
   end
```

main.cem

```
//main routine
class counter old_counter        // Define counter class.
old_counter.reset(0)             // Initialize counter.
old_counter.increment(0)         // Count one.
```

그림 5: CEMTool로 작성된 Counter 예제.
Fig. 5 Counter sample in CEMTool.

4. Matlab 과 CEMTool의 프로그램 비교

그림 4와 그림 5는 MATLAB과 CEMTool을 사용해 작성된 계수기(Counter)를 사용하는 시스템을 모델링 한 예제 프로그램이다. 계수기는 세 가지의 기능을 가지도록 프로그램을 작성한다. 계수기의 값을 1 증가시키는 기능, 1 감소시키는 기능, 계수기를 0 으로 초기화하는 기능이 그것이다.

MATLAB에서는 그림 4와 같은 형태로 코딩 된다. MATLAB에서 함수간에 자료를 전달하기 위해서는 전역 변수를 이용 하거나 함수로 전달되는 매개변수를 이용하는 방법이 있다. 여기서는 현재의 카운트 값을 기억하는 변수가 필요하기 때문에 하나의 계수기에 대해 counter_reg 라는 전역 변수를 만들었다. 하지만 시스템에서 사용하는 계수기가 늘어나면 의당 전역 변수도 새로 만들어야 한다. 전역 변수를 새로 만들었다면 이를 사용하는 함수도 새로운 이름으로 만들어야 한다. 카운트가 늘어날수록 개발자는 시뮬레이션 프로그램에 새로운 변수와 함수를 추가해야만 한다.

하지만 OOP 방식을 사용하는 CEMTool로 모델링 된 시스템의 계수기는 그림 5와 같이 하나의 클래스 형태로 만들어 져 있다. 전역 변수를 쓸 필요도 없으며 시스템에서 사용하는 계수기의 종류가 늘어날수록 계수기 클래스를 다시 만들 필요도 없다. 단지 계수기 클래스를 새로운 이름으로 선언하면 된다.
새로운 계수기가 필요하다면 *class counter new_counter* 라는 문장을 프로그램 내에 추가하면 된다.

5. 결론

본 논문에서는 현재 프로그래밍 표준으로 자리를 잡아가고 있는 OOP 방식을 CACSD 패키지에 도입 하였을때 얻을 수 있는 이점과, OO-CACSD 패키지인 CEMTool의 구현 과정을 살펴 보았다. OOP 방식을 지원하는 CEMTool은 다른 CACSD 패키지들에 비해서 객체 중심의 사고, 프로그램의 이식성과 호환성, 클래스 중심의 GUI 환경 등의 이점을 사용자에게 제공한다.

하지만 현재 CEMTool이 가지고 있는 내장 함수들은 클래스 구조로 모듈화 되어 있지 않고 순차적 프로그램 방식에서 사용하던 함수들과 마찬가지로 각각 하나씩 나누어져 설계되어 있다. 앞으로 개발되는 CEMTool이 진정한 OO-CACSD 패키지가 되기 위해서는 내장 함수들도 클래스 구조로 바뀌어야만 할 것이다. 그리고 기존의 SPP 언어로 작성된 코드를 OOP 언어로 자동으로 변환하는 기능이 추가되어야만 할 것이다.

6. 참고문헌

[1] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers*, Addison-Wesley Publishing Company, Massachusetts, 1986.

[2] H. El-Rewini and S. Hamilton, "Object technology," *IEEE Computer*, vol. 28, no. 10, pp. 58–72, October 1995.

[3] D. R. Gentner and J. Grudin, "Design models for computer-human interfaces," *IEEE Computer*, vol. 29, no. 6, pp. 28–35, June 1996.

[4] J. James, F. Cellier, G. Pang, J. Gray, and S. E. Mattson, "The state of computer-aided control system design (CACSD)," *IEEE Control Systems*, vol. 15, no. 2, pp. 6–8, April 1995.

[5] E. H. Khan, M. Al-A'ali, and M. R. Girgis, "Object-oriented programming for structured procedural programmers," *IEEE Computer*, vol. 28, no. 10, pp. 48–57, October 1995.

[6] M. Marcus, *Matrices and MATLAB*, Prentice-Hall International, Inc, New Jersey, 1992.

[7] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, New York, 1992.

[8] R. Rutz and J. Richert, "Camel: An open cacsd environment," *IEEE Control Systems*, vol. 15, no. 2, pp. 26–32, April 1995.