학 술 기 사

객체지향설계법과 구조설계 전산화

김 치 경*

1. 서 론

구조설계 과정 중 구조해석 단계에서 유한요소 해석 프로그램의 사용은 일찍부터 일반화되었으며, 최근들어 도면 작성을 위한 범용 CAD 프로그램과 부재설계 단계에서 단순반복계산을 대신해 주는 정도의 프로그램이 활용되고 있다. 그러나 이러한 컴퓨터 활용 수준은 타 공학 분야, 또는 최근 컴퓨터 기술 발전에 따른 잠재적 가능성에 비할 때 기초적 단계라 아니할 수 없다. 이러한 배경에서 최근 국내외 여러 연구자들의 관심을 모으고 있는 과제 중의 하나가 구조설계 전 과정의 전산화이며, 국내에서도 몇몇 연구자에 의하여 주목할 만한 S/W들이 개발된 바 있다. 그러나 설계용 S/W 개발에 있어서는, 순차적 수치계산이 주가 되는 구조해석용 S/W 개발에서는 볼 수 없었던 난제들을 만나게 된다.

본문에서는 설계용 S/W 개발에 있어서의 주요 난제들과 그 해결 방안에 대하여 논하고, 특히 최근 관심을 모으고 있는 객체지향설계 및 프로그래밍 기법(OODP : Object-Oriented Design and Programming Method)의 개념과, 구조설계 전산화를 위한 OODP 기법 적용에 대하여 전반적으로 고찰해보고자 한다.

2. 객체지향 설계 및 프로그래밍

2.1 객체지향설계법의 기원

컴퓨터 시스템에 있어서 S/W가 차지하는 비용이 증가하고 그 중요성이 강조되기 시작한 것은 1970년대부터이다. 이 때부터, 개발된 S/W의 규모가 클수록 그 품질이 저하되거나, 개발 비용 또는 일정이 당초의 계획을 초가하는 경우가 많아지게 되었다. 이는 S/W 개발 프로젝트의 실패를 의미하며, 그 비율이 전체 프로젝트의 90%에 이를 정도로 S/W 산업은 심각한 국면을 맞게 된다. 이러한 상황은 "소프트웨어의 위기(S/W Crisis)"라고까지 표현되었으며, S/W 개발 과정에서 사용할 수 있는 체계적인 지침이나 기법 등의 부재가 그 원인임을 파악하게 된다. 이에 대처하기 위하여 대두된 학문이 "소프트웨어 공학(SE : Software Engineering)"이다. SE에서는 S/W를 코딩으로 시작하여 코딩으로 끝낼 수 있는 단순품이 아니라, 계획, 분석, 설계, 코딩, 시험 및 유지보수의 단계에 따라 체계적으로 개발되어야 하는 생산품으로 간주하고 있다. 그림 1은 S/W의 개발 주기를 보여주는 water-fall 모델이다. 이 그림은 개발 단계가 진행될수록 feedback에 많은 비

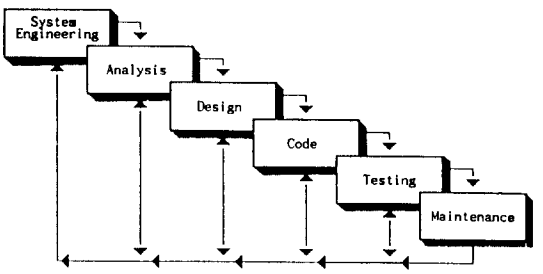

그림 1. 소프트웨어 개발 주기

* 정회원, 서울대학교 건축공학과 공학박사

용이 소요되고, 그러므로 초기 단계에서의 치밀한 문제분석 및 설계가 중요함을 강조하기 위하여 자주 인용되는 널리 알려진 모델이다.[12]

오늘날 전산학 분야에서 SE는 중요한 학문으로 인식되어 많은 S/W 개발 기법들을 제시하고 있으며, 이들중 OODP 기법은 최근 가장 주목을 받으며 실용화에 성공한 이론이다.[3][4][8]

OODP 기법은 1970년대 중반부터 발전되어 왔으며, SE에서의 연구들이 S/W의 모듈화, 추상화(Abstraction), 정보은닉(Information Hiding) 등의 중요성을 강조하면서 OODP에 대한 관심은 높아져 갔다. 1980년대에 들어서 Smalltalk, Ada, C++와 같은 객체지향 프로그래밍 언어가 급격히 발전하면서 OODP 기법은 실용화 단계에 접어들게 되었다.

객체지향 개념의 적용은 크게 대상문제 분석 및 시스템 설계 단계에서의 적용과 프로그래밍 단계에서의 적용으로 구분된다. 전자는 실제 프로그램 구현에 임하기 전단계의 작업으로서 컴퓨터와는 무관한 개념적 작업이며 객체지향설계(OOD)라 한다. 후자는 객체지향 설계 결과를 특정한 프로그래밍 언어를 사용하여 프로그램으로 구현하는 작업을 말하며 이를 객체지향 프로그래밍(OOP)이라 한다.[4][12]

2.2 기존 S/W와 객체지향 S/W

데이타+기능(함수)=소프트웨어

위의 식은 S/W를 정의하기 위해 흔히 인용되는 공식으로 데이타와 기능(함수)이 결합되어 S/W가 구성됨을 의미한다.

기존의 S/W는 기능지향적 특성을 갖는다. S/W 개발 시 흐름도 작성으로부터 출발하는 접근법은 기능지향적인 기존 S/W의 특성을 잘 보여준다. 즉 대상문제를 분석하고 프로그램을 구현하는 과정이 기능(함수)을 중심으로 수행된다. 여기에서 데이타는 부수적인 요소로 취급된다. 그러므로 하나의 S/W는 각각 나름의 역할을 하는 여러개의 함수들이 모여 완성된다.

반면에, 객체지향 S/W에서는 객체를 S/W 구성의 기본 단위로 삼는다. 객체란 S/W 영역으로 사상된 실세계의 사물로서 그 객체에 필요한 데이타와 그에 밀접하게 관련된 기능이 통합된 새로운 개념의 단위체이다. 객체지향 S/W 개발은 대상문제에 존재하는 데이타와 그에 밀접한 기능을 추출, 분류하여 객체를 정의하는 작업으로 시작된다. 이러한 작업이 객체지향설계이다.

S/W가 실세계의 현상을 컴퓨터 내에서 표현해낸 결과임을 인식할 때, 대상 사물(데이타)과 그에 가해질 수 동작(함수)을 묶어 하나의 단위체로 취급하는 접근 방법은 훨씬 더 실세계 문제 표현에 가깝고 자연스러운 방법이다.

2.3 객체지향설계법의 기본 개념

2.3.1 객체, 클래스, 멤버, 멤버함수, 메시지

전술한 대로, 객체란 S/W 영역으로 사상된 실세계의 사물로서, 객체 기술에 필요한 데이타(멤버)와 그 데이타에 직접 동작을 가할 수 있는 함수(멤버함수)가 결합된 단위체이다. 그러므로 객체는 기존 S/W에서 독립적인 존재로 취급되던 함수와 데이타보다 한 단계 상위 관점에 존재하는 취급 단위체인 것이다. 예로 보를 생각해 보자. 보에 관련된 정보, 즉 보의 형상, 사용재료, 보에 가해지는 하중, 양단에서의 단면력 등은 실세계의 보를 기술하기 위해 필요한 데이타들이다. 또한 보의 형상을 결정하는 설계 기능, 보 내부의 단면력 분포를 계산하는 해석 기능, 보의 형상을 도면으로 출력하는 도면화 기능 등은 모두 전술한 데이타 범위 안에서 수행되는 함수들이다. 보 객체는 이러한 데이타와 함수가 결합된 단위체이다.

많은 객체들은 상호 동일하거나 유사한 데이타 구조와 기능을 갖는다. 즉 각각의 보들은 서로 재료, 형상, 역학적 성능은 다를지라도 그 데이타구조나 그들에 가해질 수 있는 기능은 동일하다. 이와 같이 같은 데이타구조와 기능을 갖는 객체들의 집합이 클래스이다. 그러므로 객체(G1보)란 그가 속한 클래스(보 클래스)의 데이타구조에 특정한 값들이 배정된 결과물이다.

객체 멤버와 멤버함수는 객체 외부에서의 직접

적인 접근이 원칙적으로 금지된다. 대신 객체는 미리 정의된 메시지를 접수할 수 있다. 메시지를 접수한 객체는 접수된 메시지의 목적에 맞도록 자신의 멤버함수를 활성화시키고, 활성화된 멤버함수는 객체 멤버에 필요한 동작을 직접 가한다. 이러한 개념은 정보은닉(Information Hiding)을 실현시켜 예기치 못한 부수효과(Side Effect)를 막아줌으로서, S/W 모듈의 부품화를 가능케 하고 개발 효율 및 신뢰성을 높혀준다. 또한 이러한 개념은 하나의 방대한 대상문제를 여러개의 다루기 쉬운 소규모 문제로 분해(모듈화)해줄 뿐만 아니라, 모듈간의 상호 인터페이스가 단순화되는 장점이 있다.

2.3.2 상속성과 다형성

상속(Inheritance)은 타 기법과 구별되는 OODP이 가장 독특한 특성이다. 상속이란 한 클래스의 구조(멤버구조, 멤버함수, 메시지)가 일부만 변경 또는 추가되면서 타 클래스로 복사되는 현상을 일컫는다. 예로 부재 클래스로부터 보클래스가 파생될 수 있으며, 다시 보 클래스로부터 철근콘크리트보 클래스와 형강보 클래스가 파생될 수 있다. 이 때 보 하중 데이타구조, 보 단면력 데이타구조 등은 보 클래스에서 한번만 정의되어 철근콘크리트보와 형강보 클래스에 공히 상속되지만, 보 형상 데이타구조는 각각 새로이 정의되어 추가되어야 한다. 또한 멤버함수 정의에 있어서도 해석 기능은 보 유형에 관계없이 동일한 함수로 수행될 수 있으므로 보 클래스에서 한번만 정의되지만, 설계 기능은 보 유형마다 다른 함수로 새로이 정의되어야 할 것이다.

이러한 기법은 객체지향설계법에서 모듈의 재사용성을 향상시켜주며, 단순한 클래스 개발로부터 출발하여 계속적으로 파생클래스들을 추가해 가는 방법으로 방대한 문제를 점진적으로 개발해 나갈 수 있도록 해준다.

다형(Polymorphism)이란 한 함수가, 수행되는 시점의 상황에 따라 달리 동작하는 현상을 일컫는다. 전산 용어로 이러한 현상을 run-time binding이라 하며, compile-time binding에 상반되는 개념이다. 즉 프로그램 수행 진로가 수행 중에 결정된다는 것이다. 예로 골조 객체는 여러개의 보와 기둥 객체가 결합되어 이루어진다. 그러므로 골조 객체의 설계기능은 보 객체들에게 설계를 요구하는 메시지를 전달하게 된다. 그런데 보 객체의 유형은 철근콘크리트보일 수도 있고 형강보일 수도 있다. 이 때 골조 객체는 보 객체의 유형을 구분할 필요가 없다. 왜냐하면 같은 설계 메시지라도 접수한 보 객체의 유형에 따라 필요한 설계 작업이 해당 객체 내부에서 적절하게 수행되기 때문이다. 이러한 개념에 의해 새로운 보 객체 유형(프리캐스트 보)이 추가되더라도 골조 객체의 설계기능은 수정할 필요가 없다.

2.3.3 객체지향 프로그래밍 언어

이상에서 기술한 객체지향설계법은 단지 대상문제를 분석하고 S/W를 설계하는 기법이다. 이러한 개념을 사용하여 설계된 S/W의 구현은 일반적으로 Smalltalk, Modula-2, Ada, C++ 등의 객체지향 특성을 갖는 언어를 사용하는 것이 효율적이다. 객체지향 언어들은 위에서 언급된 객체지향설계법의 여러가지 개념들이 잘 구현될 수 있는 데이타구조와 문법을 보유하고 있다.

Smalltalk는 객체지향 개념을 바탕으로 개발된 언어로서, 객체지향 개념이 가장 충실히 지원될 수 있는 언어로 평가받고 있다. 그러나 최근들어 Smalltalk의 발전은 C++ 등장과 빠른 발전에 비해 주춤하는 느낌이다.

Ada는 미 국방부의 지원하에 S/W 재사용성의 향상에 의한 개발 비용 절감을 위하여 많은 노력이 투입되어 개발되었으며, 앞으로 가장 주목받을 객체지향 언어의 하나로 기대된다.

C++는 1980년 경에 AT&T의 Bjarne Stroustrup에 의하여 "C with Classes"라는 이름으로 처음 개발되었으며, 1985년에 AT&T C++ 1.0이 발표되었다. C++는 C언어의 문법을 대부분 사용하기 때문에 C를 따오고, C보다 향상되었으므로 ++를 사용하였다. 한다. C++는 클래스, 상속, 연산자중복, 가상함수(다형) 등과 같은 특징을 갖추고 있어 객체지향 프로그래밍에 적합할 뿐

만 아니라, C언어가 제공하는 포인터 연산을 적절히 사용하면 계산집약적인 구조해석 문제의 구현에도 기존의 베이직 또는 포트란 프로그램보다 월등한 계산효율을 갖기 때문에 앞으로 우리 분야에 가장 적합한 언어라 판단된다.

3. 구조설계 전산화를 위한 객체지향적 접근

3.1 구조설계 전산화에서의 난제

구조설계용 S/W 개발을 위해서는 해결되어야 할 난제들이 있다. 이 절에서는 이들 난제와 객체지향 개념을 중심으로 그 해결 방안에 대하여 논하고자 한다.

1) 모듈간 상관관계의 복잡성 : 더 많은 기능이 통합될수록, 각 단위 기능 또는 데이타간의 상관관계가 기하급수적으로 복잡해져 취급불능의 상태로 접어들게 된다. 즉 1000줄로 구성되는 프로그램을 위한 개발노력은 100줄로 구성되는 프로그램 개발노력의 10배가 필요한 것이 아니라 수십배, 때로는 수백배의 노력이 요구된다. 이는 인간이 한번에 고려할 수 있는 사고의 범위와도 관계가 있다. 이에 대한 현재로서의 최선의 선택이 OODP의 적용이라 하겠다. OODP의 가장 핵심적인 개념은 대상문제를 취급하기 쉬운 소규모의 독립적인 객체로 분해하여 취급하고, 이들 객체 사이의 상관관계를 최대한 단순화(오로지 정의된 메시지만으로 연결된다) 시킨다는 점이다.
2) 방대한 설계 데이타 관리 : 구조설계 과정에서 발생, 취급되는 다양하고 방대한 양의 데이타들을 하나의 데이타베이스가 전담하도록 한다.
3) 시스템, 사용자간의 의사교환 문제 : 최근 대규모 S/W에서 배우고 사용하기 쉬운 사용자접속장치의 제공 여부는 S/W의 활용성을 좌우하는 중요한 요소로 간주된다. 특히 도형적 요소가 많은 건축, 토목 구조물의 경우 잘 고안된 그래픽 사용자접속장치는 컴퓨터와 사용자간의 정보교환을 원활하게 해줌으로써 작업효율 향상에 크게 기여한다. 현재 객체지향 개념이 가장 활발히 사용되고 있는 분야가 바로 그래픽 분야이다.
4) 지능적 의사결정 문제 : 최근 이 분야의 전산화를 위하여 인공지능, 전문가시스템 등이 연구되고 있으나, 실용화 수준에는 못 미치고 있다. 그러므로 현 단계에서 이러한 문제들을 모두 S/W로 처리하고자 함은 무리이며, 대신 지능적 의사결정이 필요할 때마다 시스템이 여러 대안을 창출하여 그에 관련된 정보와 함께 잘 고안된 사용자접속장치를 통하여 사용자에게 제시하면 사용자가 선택, 수정을 통하여 최종 결정을 내리는 방법이 현재로서 가장 실용화에 성공할 수 있는 방법이라 생각된다.

3.2 객체지향 기법 적용의 타당성

대상문제에 적합한 S/W 개발 기법의 선택도 중요한 문제이다. 구조설계 문제의 특성 중 OODP 기법 적용을 고무하는 특성을 정리하면 다음과 같다.

1) 건축, 토목 구조물 및 구조설계과정은 구조물 구성요소(보, 기둥, 골조 등) 단위의 상호 독립적인 소규모 문제로 분해될 수 있는 특성이 있다. 이러한 경우 OODP 기법이 효과적으로 적용될 수 있다.
2) OODP 기법의 상속 개념이 구조물 구성요소간의 관계 표현에 무척 효과적으로 사용될 수 있다.(3.3절 참조)
3) 설계용 S/W는 구조설계 규준의 개정이나 그 밖의 여러 이유로 인하여 그 안에 담고 있는 정보 및 기능이 계속적으로 유지, 보수, 확장되어야 하는 특성을 갖는다. OODP 기법은 객체 상호간의 독립성을 최대한 확보해 줌으로써 유지, 보수, 확장이 빈번히 요구되는 경우에 적합하다.
4) 구성요소 단위로 분해되어 있는 각 객체들을 적절히 결합함으로써 다양한 유형의 구조물 표현이 가능하다. 이는 마치 부품들을 골라 맞춰 집을 지어나가는 집짓기 장난감처럼,

독립적인 구성요소 객체들을 결합시켜 전체 구조물을 S/W 영역에서 표현해낼 수 있음을 의미한다.

5) 단계별 실용화 및 단위 기능 사용 : 객체지향 S/W는 독립적인 객체 단위로 구성되기 때문에 단계별 실용화 및 단위 기능만의 사용이 용이하다. 즉 철근콘크리트보 객체가 완성되면 나름의 사용자접속기능과 함께 보설계 기능이 바로 실용화될 수 있으며, 개발자는 다시 보 객체와 기둥 객체를 결합하여 골조 객체를 구현하고 실용화한다.

3.3 구조물 구성요소 클래스 체계

그림 2는 필자가 건축구조설계 통합시스템 개발을 위하여, 객체지향설계법을 사용하여 건축구조물 및 구조설계과정을 분석하고 이에 포함되는 건축구조물 구성요소 클래스 체계를 구축한 결과이다. 구성요소 클래스(XCompo)를 최상위 베이스 클래스로 하여 3차원구조물, 2차원 수평 및 수직 서브시스템, 부재 그룹으로 대분되고, 이어 몇 단계의 상속 과정을 거쳐 각 구성요소 유형 클래스가 구체화되어감을 볼 수 있다.[2][6][7][9][10][13][15]

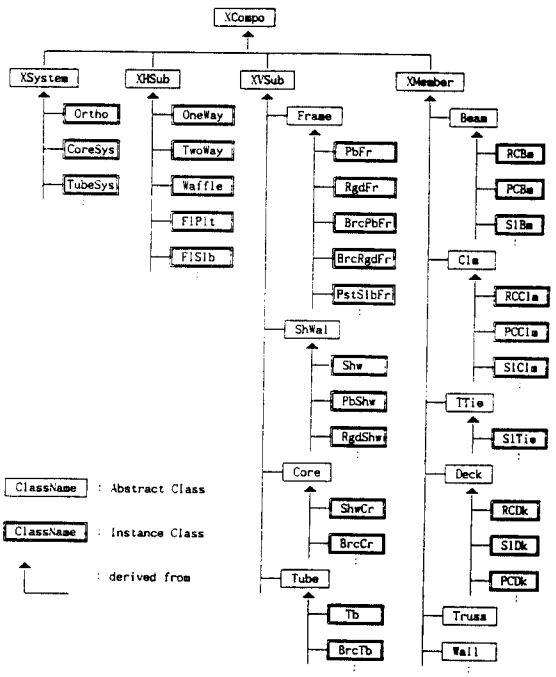

그림 2. 건축구조물 구성요소 클래스 상속도

3.4 유한요소법과 객체지향 개념

유한요소해석 프로그래밍은 그 구조가 무척 복잡하고 그러므로 내부 오류(bug)가 존재하기 쉬운 특성을 갖는다. 이런 이유로 프로그램의 바탕이 되는 이론 개발보다는 도리어 프로그래밍에 더 많은 노력과 유지, 관리 비용이 드는 사례를 종종 볼 수 있으며, 완성된 프로그램일지라도 그것의 신뢰도를 확인하는 작업이 수월치 않다. 또한 새로운 해석기법의 개발, 새로운 유한요소의 개발, 새로운 수학적 재료 모델의 개발등은 모두 기존 프로그램의 수정, 확장을 계속적으로 요구한다. 그러나 기존의 유한요소해석 프로그램에서는 요구되는 기능을 추가하기 위하여 전체 프로그램을 손봐야 하거나, 심지어는 이 작업이 불가능한 경우도 종종 있다. 여기에 계산집약적인 유한요소해석 프로그램의 특성으로 인하여 계산효율 향상에 막대한 노력이 투여되어 왔다. 그러나 이제는 계산효율 향상은 그리 중요한 이슈가 되지 못한다. 과거에 슈퍼컴퓨터라 불리우던 컴퓨터의 수십배 계산성능을 갖는 컴퓨터를 이제는 개인이 책상 위에 놓고 사용할 수 있다.

그러므로 이제 유한요소해석 프로그램 개발에 있어서의 초점은, 어떻게 하면 고성능을 갖는 프로그램의 개발, 수정, 유지관리, 기능 확장을 최대한의 신뢰도를 확보하면서 최소한의 노력을 투입하여 수행할 것인가 하는 점이다. 이러한 배경에서 필자는 OODP 기법의 적용을 권하고 싶다. 전술한 OODP 기법의 특성과 장점들이 해석프로그램 개발에도 공히 적용된다. 특히 객체지향 프로그래밍 언어의 하나인 C++를 사용하면, 포인터에 의한 수치연산을 통하여 기존 Fortran으로 코딩된 프로그램에 비할 때 계산효율 면에서도 도리어 탁월한 성능을 발휘하는 것으로 밝혀지고 있다.[1][5][11]

3.5 사용자접속장치

사용자접속장치의 필요조건은 배우고 쉽고 사용하기 쉬우며 그러면서도 사용자와 컴퓨터간의 정보의 교환이 자유로와야 한다.

필자는 사용자접속장치의 운영환경으로 MS-Windows를 선호한다. 건축구조 분야 컴퓨터 환경의 주종을 이루는 IBM-PC에서 그래픽과 그 밖의 사용자접속환경이 가장 탁월한 OS로 판단된다. 또한 최근 발표된 MS-Windows NT가 보급될 경우 PC와 워크스테이션 사이의 구분없는 이식성이 예상되고 있으며, NT의 강력한 내장네트워킹으로 기능으로 컴퓨터 네크워킹의 표준이 될 가능성도 점쳐지고 있다.

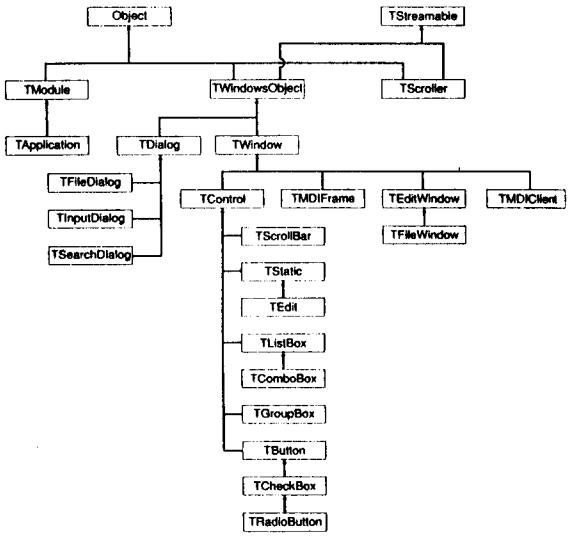

그림 3. Object Window Class Hierarchy

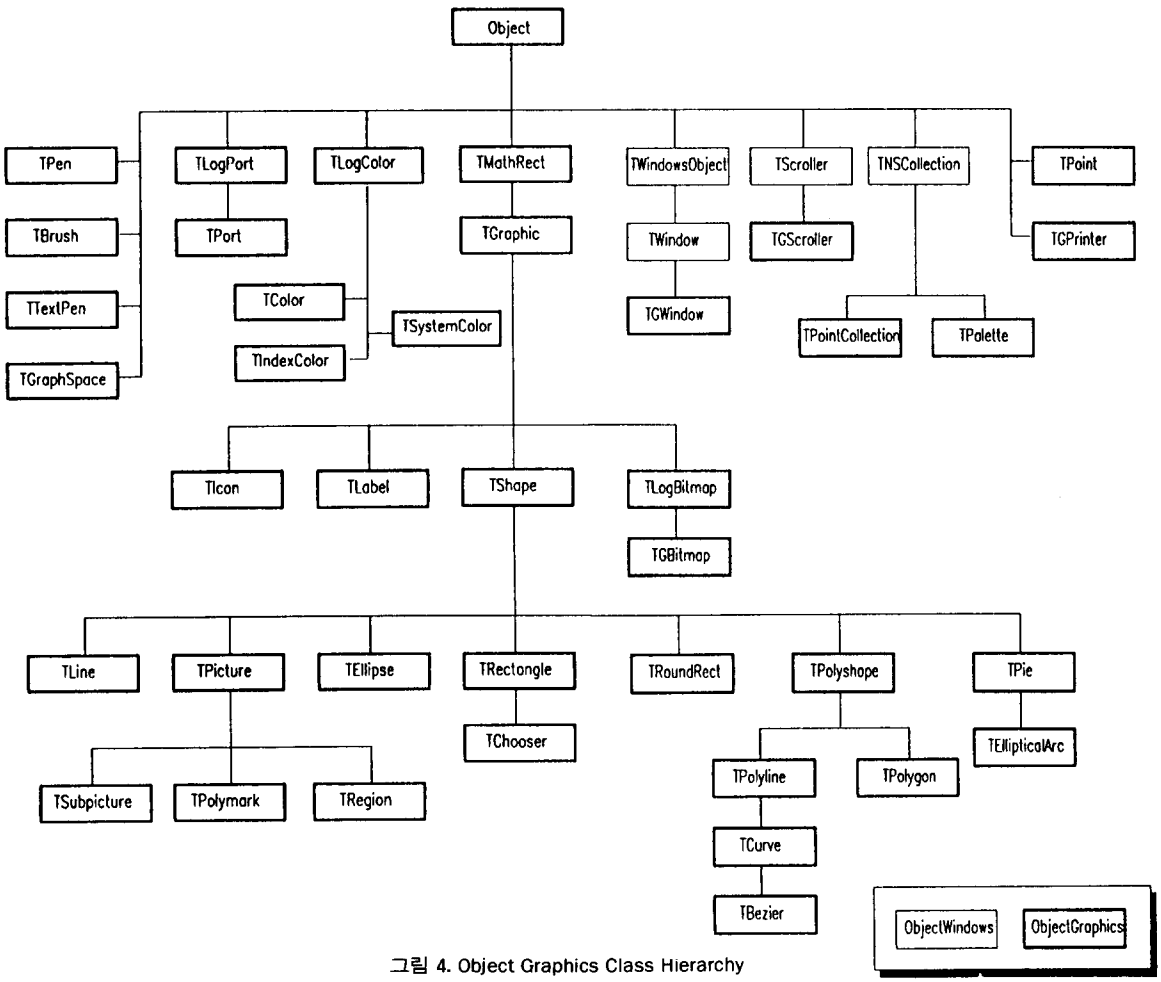

그림 4. Object Graphics Class Hierarchy

필자는 윈도우의 모든 기본 동작을 제공하는 Object Window Class Library(OWL, 그림 3 참조)와 그 안에 그려지는 각종 그래픽 처리기능을 제공하는 Object Graphics Class Library(OGL, 그림 4 참조)을 사용하여, 객체지향 그래픽 사용자접속장치를 쉽게 구현하고 있다. 이 두 라이브러리에 바탕을 둔 사용자접속기능은 사용법에 일관성을 줄 수 있어 따로 사용법을 배우지 않더라도 사용자가 동작을 예측할 수 있는 장점이 있다. 예로서, 그림 5는 보 요소의 하중 케이스별 보 단면력 및 하중과 그에 따른 단면력 분포를 편집하고 제시하는 화면이다. 이들 화면에서 보여지는 각종 도형들은 단순한 그림이 아니라 각각 독립된 객체들로서 추가, 삭제, 수정 등이 가능하고, 또한 나름의 구조 관련 정보와 연계되어 있다.[6][9][15]

4. 결 론

이상에서 객체지향설계 및 프로그래밍 기법의 기본적인 개념과 구조설계 전산화에 관한 원론적인 사항에 대하여 고찰하여 보았다.

OODP는 이미 S/W 개발 분야에서 일반화된 기법이며, 한 때 큰 관심을 모으다 결국 실용화에 실패한 인공지능 또는 전문가시스템과는 달리, S/W 개발 및 유지관리에 큰 발전을 가져오고 있다. 객체 단위에 의한 데이타 및 함수의 통합모듈화, 추상화 및 정보은닉에 의한 객체의 독립성 확보, 객체지향 프로그래밍 언어의 탁월한 이식성,

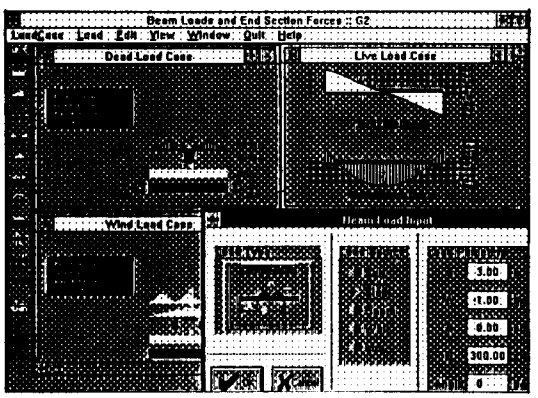

그림 5. 보 하중 및 단면력 편집

객체 추가에 의한 대규모 S/W의 점진적 개발, 상속성 및 다형성에 의한 프로그램 코드의 재사용성 등은 S/W의 부품화를 가능케 하고 S/W 신뢰도를 크게 높혀 주고 있다. OODP는 구조설계 전산화에도 적합한 기법으로 판단된다.

OODP 기법은 전산전문가들의 도구가 아니며 각 분야 전문가들이 자기 분야의 전산화를 위해 사용하는 도구이다. 과거 우리가 fortran을 익혀 구조해석용 S/W를 개발했듯이 이제는 OODP 기법을 익혀 적극적으로 우리 문제에 활용한다면, 우리 분야 전산화 수준을 높히는데 크게 도움이 될 것으로 기대된다.

참 고 문 헌

1. Abdalla, J.A., & Yoon, C.J., "Object-Oriented Finite Element and Graphics Data-Translation Facility", *Jrnl. of Comput. in Civil Eng.*, Vol.6, No.3, pp.302-322, 1992.
2. Baugh, J.W., "Data Abstraction in Engineering Software Development", *Jrnl. of Comput. in Civil Eng.*, Vol.6, No.3, pp.282-301, 1992.
3. Cattell, G.G., *Object Data Management : Object-Oriented and Extended Relational Database Systems*, Addison-Wesley, MA, 1991.
4. Cox, B.J. & Novobilski, A.J., *Object-Oriented Programming : an Evolutionary Approach*, 2nd Ed., Addison-Wesley, Ma, 1991.
5. Forde, B.W., Foschi, R.O., & Stiemer, S.F., "Object-Oriented Finite Element Analysis", *Comput. & Struc.*, Vol. 34, No.3, pp.355-374, 1990.
6. Hong, S.M., & Kim, C.K., "Integrated Information Management Expert System for Structural Engineering of Buildings", *Microcomputers in Civil Eng.*, Vol.8, No.1, pp.9-26, 1993.
7. Howard, H.C., Abdalla, J.A., & Phan, D.H.D., "Primitive-Composite Approach for Structural Data Modeling", *Jrnl. of Comput. in Civil Eng.*, Vol.6, No.1, pp.19-40, 1992.
8. Hughes J.G., *Object-Oriented Databases*, Prentice-Hall, 1991.
9. Kim, C.K., & Hong, S.M., "Development of an Integrated System for Structural Design of

Buildings Using OODP Method", *Proc. of 5th Int. Conf. on Comput. in Civil and Bldg Eng.*, pp. 521－528, CA, 1993.

10. Law, K.H., "A Formal Approach for Managing Building Design Information in a Shareable Relational Framework", *Proc. of the 4th Int. Conf. on Comput. in Civil and Bldg Eng.*, pp. 339－346, Tokyo, 1991.
11. Lawrence, K.L., Yee, V., Knipe, R., & Nambiar, R.V., "C Routines for FEM Applications", *Comput. & Struc.*, Vol.44, No.5, pp.1149－1167, 1992.
12. Pressman, Roger S., *Software Engineering : A Practitioner's Approach*, 2nd ed., McGraw-Hill, 1987.
13. Sause, R., & Powell, G.H., "Object-Oriented Approaches for Integrated Engineering Design Systems", *Jrnl. of Comput. in Civil Eng.*, Vol.6, No.3, pp.248－265, 1992.
14. Scholz, S.P., "Elements of an Object-Oriented FEM＋＋ Program in C＋＋", *Comput. & Struc.*, Vol.43, No.3, pp.517－529, 1992.
15. 김치경, 홍성목, "객체지향설계법을 이용한 건축구조 통합시스템 개발에 관한 연구", 대한건축학회논문집, 8권 11호, pp129－140, 1992.