객체지향개념을 이용한 유한요소 구조설계 시스템 개발

이 상 갑* · 장 승 조**

Development of Finite Element Structural Design System using Object-Oriented Concept

*Sang-Gab Lee** · *Seung-Cho Chang***

〈목 차〉

Abstract

The purpose of this paper is to develop an integrated environment system for finite element structural analysis using OOA(Object-Oriented Analysis) and OOD(Object-Oriented Design), which may reduce inconveniencies in use such as file input of macro command and improve lacks of graphic presentation in the established finite element analysis program. This paper is attempted to suggest an easy approach to object-oriented concept and convenient programming. Two languages are used together in this paper insteac of single C++ language for the development of object-oriented program. : Visual Basic with CDK(Custom Development Kit), and Borland C++ with OWL(Object Windows Library).

1. 서 론

60년대 이후 많은 유한요소 프로그램[1]들이 개발되어 왔으며, 현재는 NASTRAN, ANSYS, ABAQUS등 뛰어난 성능을 가진 범용 유한요소 프로그램들이 개발되어 구조해석에 직접 사용되고 있다. 이러한 프로그램들을 이용한 구조해석은 기존 구조해석의 이론적 검증과 미래지향적 설계기법의 창출을 가능하게 해 주었고, 최적 설계에 따른 구조물의 신뢰성 및 생산성 향상에 많은

* 정회원, 한국해양대학교 조선공학과 교수
** 한국해양대학교 조선공학과 대학원생

도움을 주었다. 그러나 국내의 경우 구조해석용 프로그램 사용에 있어서 대부분 선진 외국에서 개발된 것들을 사용하고 있어 고가의 구입비와 유지비를 부담하고 있고, 관련 기반 기술(Source)의 미공개로 국내 사용자들의 특성에 맞게 자체적으로 프로그램을 확장하거나 개선하기가 어렵다. 이러한 현실적 상황을 개선하기 위해서 프로그램의 전·후처리(Pre/Post Processing) 기능을 개발하여 외부접속(External Interface)을 통한 개선점을 찾고 있지만, 그렇게 할 경우 다단계식 처리 방식으로 실제적인 통합환경(Integrated Enviroment)이 가지는 일괄적인 처리를 할 수 없는 문제점이 있다. 또한 프로그램의 가격이 높음으로써 일부 대기업 외에는 사용이 어려운 실정이므로 산업구조의 전반적인 면에서 상호 일관성있는 발전을 유도한다는 것은 어려운 상태이다. 기존의 범용 프로그램은 광범위한 분야를 해석할 수 있지만 전문적이고 세분화되어 있는 분야에는 한계가 있다. 또한 계속적인 프로그램의 해외 의존으로 프로그램 개발과 관련한 기술종속의 악순환이 계속될 수 있다[2]. 이러한 현실적인 문제는 국내 프로그램 개발의 연구 노력을 저해하는 결과를 낳고 있다. 외국에서 개발된 프로그램을 사용하는데 나타나는 현실적인 문제점들을 개선하고 프로그램의 국산화를 통한 선진 외국과 경쟁하기 위해서는 프로그램 개발과 관련한 기술투자 및 개발 인력의 확충, 그리고 새로운 시스템(System) 설계개념 적용등이 필요하다고 할 수 있다.

특히 선진국의 경우 시스템 설계에 있어서 자체적인 기술투자 및 연구 노력으로 기존의 시스템이 가지고 있는 단점들을 개선하여 객체지향 설계개념(Object Oriented Design Concept)[3]을 유도하였고, 현재는 시험단계를 지나 객체지향 프로그래밍 언어까지 개발하여 시판중에 있다. 객체지향 시스템 설계개념은 프로그램 개발에 있어서 개개의 프로그램 구성 요소들을 현실 세계와 근접한 객체(Object)로 모델링(Modeling)하여 프로그램을 개발하기 때문에 체계적인 프로그램을 설계할 수 있고, 지속적인 개선(Upgrad) 및 추가적인 성능향상을 쉽게 할 수 있다. 또한 재사용 가능한 코드(Code)를 제작하여 유사한 프로그램을 개발할 때 바이너리(Binary) 형태로써 확장성을 제공하기도 한다. 따라서 현재의 객체지향 설계 방식은 다른 어떤 방식보다도 효율적인 것으로 인식되고 있다.

본 논문에서는 이러한 객체지향 설계 개념을 적용하여 구조 해석용 유한요소 통합 환경 시스템을 구현하는 것이 목적이다. 본 연구에서는 Bailin의 요구 명세(Object Oriented Requirements Specification) 개념과 Shlear/Mellor의 정보 모델링(Object Oriented Information Modeling) 개념을 응용하였다[4]. Bailin의 요구 명세론은 기존의 구조적 시스템 개발 방법론에서 채용하고 있는 방식에 객체지향 특성을 결합시키는 방법으로 데이타의 흐름은 구조적 기법과 유사하나 거의 같은 부류의 데이타와 서브 프로시져(Sub Procedure)을 하나로 묶어 객체를 모델링 한다는 것이 기존의 구조적 방법과의 차이점이라 할수 있다. 즉, 기존의 구조적 방법으로 시스템을 설계하던 방식을 어느정도 적용할수 있다는 장점이 있다. Shlear/Mellor의 객체지향 방법론은 실질적인 정보 모형을 찾는데서 부수적으로 발생하는 외부 프로세서의 개발이라는 점에서 수학적 해나 공학용 결과를 단일 시스템으로 설계할 때는 적용 기준을 찾기가 어렵지만 모든 문제의 영역을 정보(Information)에 기준을 두고 그에 따른 영역과 프로세스의 모델로써 충분히 표현할 수 있기 때문에 정보 형태의 데이타를 처리하는 시스템 설계에 있어서는 효과적이라 할 수 있다. 전자의 경우 요구되는 객체를 추상적 명사로 추출하고 객체간에 공유되는 자료(Data) 및 동작(Operation)과의 가시 범위를 정하는데 적합하여, 본 연구에서는 설계하고자 하는 시스템의 객체 추출 및 형성과정에 사용하였다. 후자의 경우는 신호처리와 관련한 통신 시스템 및 외부 장치를 많이 필요로 하는 대형 시스템 설계에 많이 사용되고 있으며 본 연구에서는 시스템 개개의 객체가 전처리기(Preprocessor), 해석(Solver), 후처리 기(Postprocessor)의 메시지(Message) 처리에 의하여 제어(Control)되는 객체의 생명주기(Life Cycle)를 결정하는데 사용하였다.

본 연구에서는 VBX(Visual Basic Control)[5]와

C++[6] 언어에서 제공하는 객체지향적 프로그래밍(OOP : Object Oriented Programming) 기법을 적용하여 전체적인 객체지향 시스템을 구현하였다. 개발 프로그램은 객체지향적 특성을 가지는 언어로 제작되었지만 객체지향적 프로그래밍 보다는 최초의 시스템 설계에 적용되는 이론적 개념의 응용에 중점을 두어 개발하였다. 시스템 개발에 사용된 언어는 일반적인 C++ 언어가 가지는 단점들을 개선하여 위에서 언급한 두가지 언어를 복합적으로 사용하였고, 이러한 방법을 통하여 효율적으로 시스템을 설계하는 방법을 제시하였다. 여기에서 재사용 라이브러리(Library)의 설계는 C++ 언어로 제작하고, 기타 GUI(Graphic User Interface) 구현 및 처리속도를 요하는 부분은 VB(Visual Basic)을 사용하였다.

2. 시스템 설계개요 및 적용개념

2.1 프로그램 개발 언어 및 특징

본 유한요소 구조해석용 통합환경 프로그램은 OWL(Object Windows Library)과 CDK(Custom Development Kit)를 포함하는 Borland C++ 4.0과 MS-Visual Basic 3.0 (Professional)의 혼합 언어로 구성되어 있다. 개발 시스템의 제작 언어구성의 범위는 Fig. 2-1과 같다.

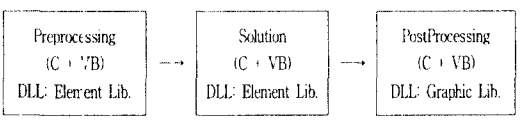

Fig. 2-1 Applied Language Scope of System

객체지향 개념을 이용한 프로그램 개발에 있어서 VB는 제한적 기능을 갖지만, 기존의 C++에 비하여 GUI를 구현함에 있어 코딩(Coding)의 양을 현저히 줄일 수 있고 그에 따른 시간 절약 및 컴파일(Compile) 과정에서 나타나는 에러(Error)를 방지할 수 있다. VB는 메소드(Method)의 제한적 사용으로 완전한 객체지향을 지원하지는 않지만 다른 응용 프로그램(Windows Application)과의 연계성을 이용하여 폭넓은 확장성을 제공하고 있다. 즉, 기존의 윈도우즈 환경이 가지고 있는 API (Application Program Interface) 함수와 DLL (Dynamic Link Library)를 이용한 C++ 언어와의 연결은 윈도우즈 응용 프로그램을 개발하는데 있어 다른 어떤 언어보다 쉽고 빠르게 개발할 수 있는 여건을 제공한다. 또한 C++ 언어로 작성된 클래스 라이브러리(Class Library)는 기존 VB로 개발된 프로그램을 확장할 때 그대로 재사용 가능하기 때문에 프로그램 관리의 효율적인 면을 기대할 수 가 있다. Fig. 2-2는 VB의 윈도우즈 환경하에서 적용되는 언어의 확장성 및 DLL과 API 함수의 적용과정과 상속성을 이용한 C++ 언어의 VB 적용과정을 그림으로 나타낸 것이다. 여기에서는 C++의 상속 과정의 일부만을 도시하였고 OWL에 준하여 나타내었다.

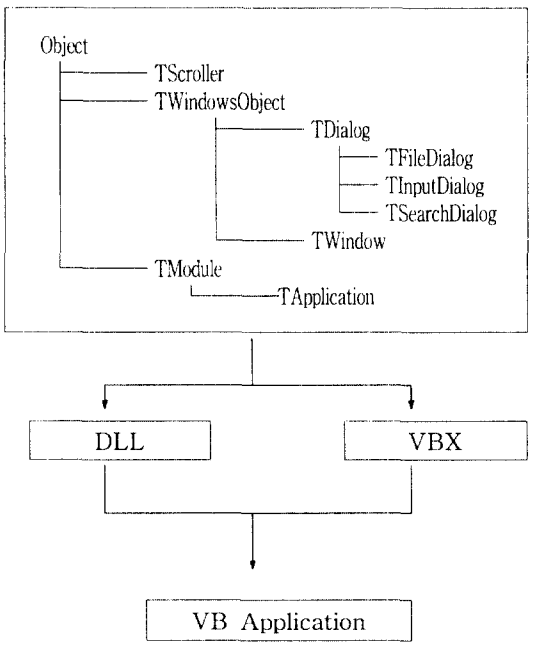

Fig. 2-2 Development Process of Application Program by Inheritance

Fig. 2-2에서와 같이 OWL의 최초 "Object"로부터 계승되어 개발된 대화상자(Dialog Box), 자료입력상자(Data Input Box) 그리고 다중윈도우(Multi-Window)등의 상속에 의하여 제작된 객체는 실질적으로 DLL이나 VBX로 제작되어 VB에서 적용된다. 최상위 "Object" 클래스는 Object-Windows 클래스에 대한 일반적 구조와 기능을 정의하는 추상 클래스(Abstract Class)로 표현된다. 이러한 상속에 의하여 정의된 객체를 사용하는 것은 윈도우즈 응용 프로그램의 사용자 접속장치(User Device Interface)를 개발할 때 생기는 많은 양의 코딩을 줄이고 프로그램의 신뢰성을 향상시킨다.여기에서 VB의 사용에 의한 사용자 접속장치는 Fig. 2-3에서와 같이 윈도우즈 접속장치를 그대로 이용한다. 즉 입력을 실행하는 마우스(Mouse)나 키보드(Keyboard), 출력을 실행하는 프린터(Printer)등의 외부 접속장치는 별도의 프로그래밍 없이 윈도우즈의 접속 장치를 그대로 이용한다. 이러한 접속 장치와 관련한 프로그래밍를 하지 않는 것은 설계시 윈도우즈 환경의 일관성 있는 장치를 활용하게 되므로 인터페이스(Interface)는 유사성을 유지하며 그에 따른 시스템의 안정성을 높일 수 있기 때문이다.[7] 그러나 프로그램을 구동하기 전에 윈도우즈에 대한 명확한 외부 접속장치를 정의해 주어야 하며 출력되는 문자의 형태(Fonts)나 출력 방향 및 출력 조건등은 별도의 코딩으로써 제작해도 무관하다. 본 연구에서 외부 장치의 접속은 윈도우즈의 접속 장치를 그대로 사용하였으며 전 후 처리의 그래픽 출력 부분에 있어서는 출력의 형태만을 지정해 주었다.

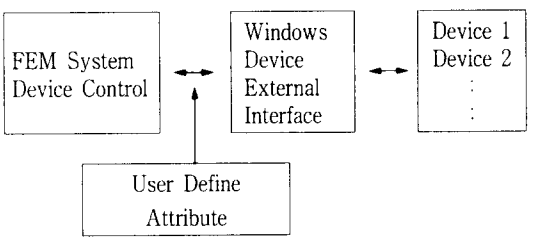

Fig. 2-3 Windows Device Using in Windows Application

프로그램의 구동은 윈도우즈 환경하에서 GUI 및 멀티태스킹(Multi-Tasking)을 구현하기 때문에 MS Windows 3.1이 구동에 필요하고 관련 DLL 파일이 포함되어 있어야 한다. 또한 프로그램의 실행시 속도감을 느끼기 위해서는 486 프로세서(Processor)를 장착한 메모리(Memory) 8MB정도의 시스템 사양을 요구한다.

2.2 객체지향 분석 및 설계

객체지향 개념을 이용하여 시스템을 설계할 때는 주어진 문제에 대한 객체지향 분석(OOA : Object Oriented Analysis)과 객체지향 설계(OOD : Object Oriented Design)가 선행되어야만 효과적인 시스템을 구성할 수 있다. 본 연구에서는 우선 객체지향 접근과 객체지향 설계개념을 정립하고, 실질적인 코딩 과정에서 객체지향 프로그래밍을 접목하여 시스템을 개발한다. 따라서 객체지향 프로그래밍 기법은 차후의 문제로 제기된다. 시스템 설계 및 객체지향 프로그래밍 방식의 적용과정과 최종적인 시스템 완성단계까지의 객체지향 개념의 효율적인 적용 과정을 Fig. 2-4에 도식하였다. Fig. 2-4에 나타낸 것과 같이 최초의 객체지향 시스템 설계에 있어서는 무엇보다 객체지향 접근 및 객체지향 해석에 바탕을 둔 시스템의 전반적인 흐름을 파악하여야 한다. 이것은 객체지향 시스템 설계에 있어서 실제적인 프로그래밍 과정보다는 설계 측면에 입각한 전체적인 구상을 중요시 하는 것으로 OOA/OOD는 이러한 구상을 현실적 상황에 맞게 전개할 수 있게 한다[8]. Fig. 2-4는 시스템 설계할 때에 최초에 고려되어야 할 시스템 설계 방법론을 제시하고, 개발하고자 하는 시스템에 적용할 객체지향적 접근 방식을 찾는다. 또한 OOA/OOD의 다음 단계에서 이루어지는 객체의 추출은 객체지향 시스템 설계에 있어 가장 어려운 문제 중의 하나라 할 수 있다. 즉, 설계 하고자 하는 시스템에서 시스템과 관련되는 부분들을 명사화 시켜 가시적인 모델링을 한다.다음 단계는 가시적으로 명사화된 객체들을 추상적인 자료형(Data Type)에 맞게 생성시키고, 동시에 자료의

가시범위와 동작(Operation)될 함수를 함께 묶는 선행 단계에 있는 실질적인 코딩과 관련된 객체를 생성한다. 추상적으로 명사화된 객체의 자료 범위와 동작될 함수가 정의되면 프로그래밍을 통하여 객체를 생성하고, 마지막으로 생성된 객체들의 메시지 전달을 포함하여 전체적으로 통합한다.

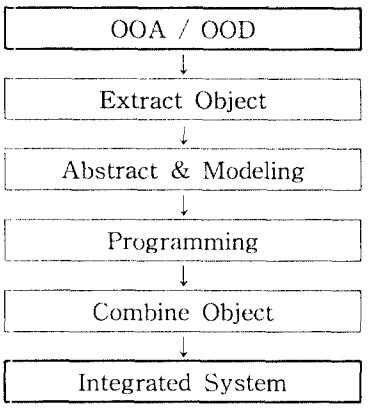

Fig. 2-4 Object-Oriented System Design Process

OOA/OOD를 통하여 주로 이루어지는 것은 객체지향 시스템설계에 있어서 방법적인 면을 나타낸다. OOA/OOD는 조합형(Combination Type)과 적응형(Adaptive Type) 두가지로 분류되는데 OOA/OOD의 방법론의 구조를 Fig. 2-5에 나타내었다. 조합형은 Booch[3], Schlaer/Mellor[4]등이 대표적 방법론으로 구조, 기능, 동작의 세가지 관점을 조합하여 분석·설계한다. 적응형은 기존의 프로그램 개발 방법론에 객체지향의 요소 기술을 포함하여 개선하는 방법이다[8].

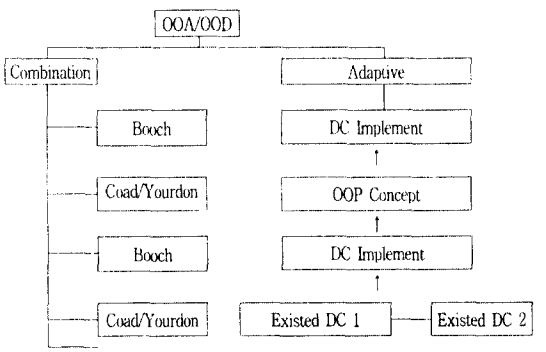

Fig. 2-5 Object-Oriented Design Methodology

본 연구에서는 실제의 개발에서 현행의 방법론과 차이가 있는 것에 한해서는 적응형 방법론을 적용하였다. 그러나 시스템 전체를 체계적으로 보는 것이 중요하기 때문에 시스템 설계의 전반적인 면은 전자의 방법을 따르고 있다.

3. 통합 환경 시스템 구축

본 연구의 개발 시스템의 전반적인 구성은 Fig. 3-1과 같이 크게 전처리기, 해석, 후처리기로 구성되어 있다.

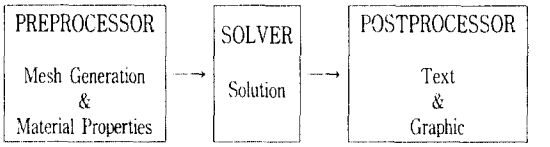

Fig. 3-1 Processing of System

객체지향 시스템 설계 방법에 대한 객체 모델링 방법은 전처리 단계에서 주로 설명하고, 해석 단계와 후처리 단계를 구성하는 객체에 대한 모델링 방법은 전처리 단계와 같은 방식으로 설계 되었기 때문에 생략하고자 한다.

시스템을 구동하게 되면 Fig. 3-2와 같이 시스템의 로고(Logo)와 함께 전체적인 시스템의 구성이 나타난다. 시스템의 명칭은 유한 요소 해석 통합 환경 시스템의 의미로 『FEAP』라 명명한다.

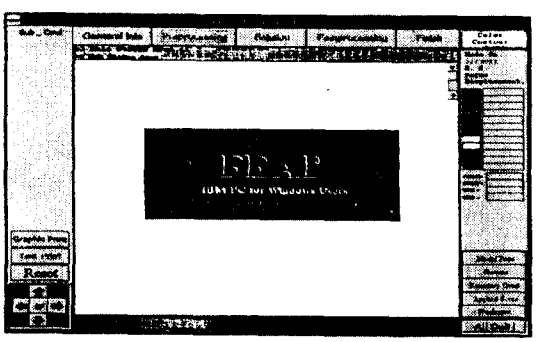

Fig. 3-2 System Initial Driving Screen

시스템의 구성은 주메뉴 영역(Main-Menu Area), 부메뉴 영역(Sub-Menu Area), 그래픽 표현 영역(Graphic Presentation Area), 텍스트 명령어 영역(Text Command Area), 그래픽 칸투어 영역(Graphic Contour Area), 텍스트 표현 영역(Text Presentation Area)으로 구성되어 있다. 주메뉴 영역은 본 프로그램의 전체적인 흐름을 제어하는 전처리 메뉴(Preprocessing Menu), 해석 메뉴(Solution Menu), 후처리 메뉴(Postprocessing Menu)로 구성되어 있으며, 그 외에 시스템의 일반적인 정보를 알려주는 정보 안내 메뉴(General Information Menu)와 각 단계를 종료하는 종료 메뉴(Finish Menu)로 구성되어 있다. 부메뉴 영역은 주메뉴 영역에 있는 임의 메뉴에 대한 메시지 응답 결과로 발생되는 동적 객체 영역(Dynamic Object Area)으로 주메뉴와 관련되는 세부적인 기능들을 포함한다. 그래픽 표현 영역은 로고가 위치한 부분으로 전·후처리가 그래픽으로 표현되며, 그래픽 칸투어 영역은 구조물의 모델링에 있어서 요소(Element), 절점번호(Node Number), 경계조건(Boundary Condition), 하중(Applied Load)들의 표현 방식을 정의하고, 후처리 단계에서는 응력분포(Stress Distribution)의 등급(Level), 각 방향의 최대 변위 및 최대, 최소 응력 값을 나타낸다.

3.1. 전처리 단계

전처리 단계에서는 해석하고자 하는 문제의 데이타(Data)를 생성(Generation)하거나 기존의 데이타를 확인(Confirmation) 및 수정(Modification) 작업이 수행되어 진다. 주메뉴 영역의 『Preprocessing』을 선택하면 전처리 과정이 실행되고, 동시에 부메뉴 영역에는 『FEAP』과 『READ』가 나타난다. 『FEAP』은 해석하고자 하는 문제에 새로운 데이타를 생성하는 경우, 『READ』는 기존의 데이타를 확인하거나 수정하고자 할 경우에 선택한다. 『FEAP』을 선택하면 8개의 부메뉴가 『FEAP』메뉴 아래에 나타나고, 해석하고자 하는 문제의 기본적인 정보를 입력하기 위해 『STARt』를 선택하면

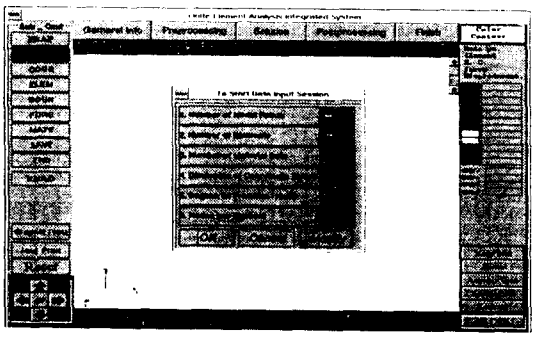

Fig. 3-3 Response Screen of 『STAR』Menu in Preprocessing Step

Fig. 3-3과 같이 또하나의 입력 윈도우가 표현되어 입력을 실행하게 되고 입력되는 데이타의 종류는 Table 1와 같다.

부메뉴에서 실행 중에 있는 메뉴는 배경색(Background Color)이 바뀌어 현재의 실행 단계를 알수 있다.

Table 1 Message and Data Type in 『STARt』

	Message	Variable	Data Type
1	Number of Nodal Points	numnp	Integer
2	Number of Elements	numel	Integer
3	Number of Material Sets	nummat	Integer
4	Dimension of Coordinate Space	ndm	Integer
5	Degree of Freedom per Node	ndf	Integer
6	Nodes per Element	nen	Integer

다음 단계로서 절점, 요소, 경계 조건, 하중, 재료 특성(Material Property)등을 입력하게 된다. 부메뉴에 있는 입력을 실행하면 입력과 동시에 텍스트 표현 영역에는 입력 데이타, 그래픽 표현 영역에는 그래픽으로 동시에 나타나게 되어 입력 데이타를 가시적으로 확인 할 수 있다. 입력 방식은 Fig. 3-4에서와 같이 대화상자(Dialog Box)를 통하여 입력이 이루어진다.

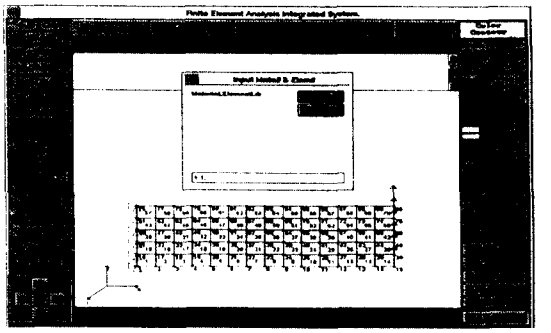

Fig. 3-4 Data Input DialogBox of 『MATE』 Menu in Preprocessing Step

입력 방식은 Table 2와 같으며 증분값(Incremental Value)에 의하여 쉽고 빠르게 데이타를 입력할 수 있다.

『READ』를 선택하면 대화상자가 나타나고, 여기에 파일명(File Name)을 입력하면 부메뉴인 『EDIT』와 『VIEW』가 『READ』아래에 나타난다. 이들은 각각 기존의 데이타를 수정 및 확인하는 작업을 수행한다. 『VIEW』를 선택하면 Fig. 3-5와 같이 『FEAP』의 부메뉴 속성(Attribute)이 변환되고 각 단계에서의 입력 데이타를 그래픽과 텍스트로 출력한다. 텍스트 표현 영역은 그래픽 표현 영역보다 차지하는 범위가 작은 반면에 스크롤(Scroll) 기능이 있어 64KB까지 순차 파일(Sequential File)의 데이타를 볼 수 있다.

Table 2 Input Format of Sub-Menu

Sub-Menu	Input Format
COORdinate	Node, GenInc, X-Coord, Y-Coord, Z-Coord
ELEMent	Elem, Mate, Node1, Node2, Node3,…, GenInc
FORCe	Node, GenInc, DOF1, DOF2, DOF3,…
BOUNdary	Node, GenInc, DOF1, DOF2, DOF3,…
MATErial	Mate, ElemLib
SAVE	Save File Name

Table 3. Object Oriented Requirements Specification and Information Modeling

단계	Bailin's Object-Oriented Requirements Specification	Shlaer & Mellor's Object-Oriented Information Modeling
1	Object Creation in Problem Area	Information Model Development
2	Definition-Active Object, -Passive Object	Definition Object Life Cycle
3	Object-Data Flow Diagram	Definition Dynamic Relationship
4	Seperate Object	Definition System Dynamic
5	Newly Creation Object Review	Process Model Development
6	Make Object Group	Definition Problem & Sub-System Area
7	Definiton Object Relationship	Integration

이제까지 기술한 전처리 단계에서의 객체 모델링 및 객체의 행동 양식 정의는 메뉴로써 표현한다. 각각의 메뉴들은 자료와 함수를 함께 공유하고 해당되는 내용에 대하여 독립적인 행동 양식을 갖고, 전달되는 메시지에 따라 반응을 한다.

전처리, 해석, 후처리 단계를 포함하는 전체적인 객체지향 통합환경 시스템의 설계는 Table 3와 같이 Bailin의 객체지향 요구 명세 방법론과 Shlear/Mellor의 정보 모델링 방법에 의한 객체지향 개발 과정을 따른다.

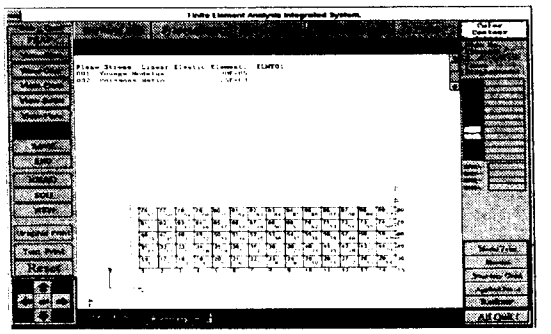

Fig. 3-5 Menu Attribute Transformation in 『VIEW』Mode in Preprocessing STEP

Bailin의 방법론에 의한 시스템 설계 과정에서 먼저 선행되는 것은 주어진 문제 영역에 대한 객체(Object)를 찾는 것이다. 이것은 객체지향 시스템을 설계할 때 반드시 거쳐야 될 부분으로 가능한 한 추상적 명사의 조건을 갖는 시스템 구성 요소 중에서 선택을 하게 된다. 전처리 단계에 있어서는 분제 해석에 필요한 자료를 추출하고 각각의 자료에 대해 추상적 명사를 유도함으로써 객체를 생성한다. 이 단계에서 고려되는 사항은 절점, 요소, 경계조건, 하중, 재료 특성을 들 수 있다. 입력 단계에서의 추상화된 객체의 세부 사항은 Table 4와 같다.

Table 4 Abstract Data in Preprocessing Step

Object Name	Index
STAR	General Data
COOR	Nodal Coordinates Data
ELEM	Nodal Connectivity, Material Sets
FORC	Force/Displacement
BOUN	Restraint Code(Boundary Condition)
MATE	Material Properties

다음 단계로써 객체는 능동적 특성을 지니는 것과 수동적 특성을 지니는 것으로 분류된다. 전자는 자신의 동작으로 메시지를 발생하여 다른 객체의 동작 여부를 제어할 수 있는 객체로, 후자는 반대의 경우의 특성으로 정의하였다. 객체란 프로그래밍 의미에서 데이타와 그 데이타를 처리하는 함수를 함께 묶어 놓은 것으로 현 단계에서는 실질적인 객체 모델링이 완성된 상태는 아니다[9]. 따라서 각각의 데이타를 처리할 수 있는 프로시져(Procedure)를 정의해 주고 그에 따라 성립된 객체의 특성을 다시 세분화하여야 한다. 이러한 작업은 객체의 메세지 처리에 의한 활동 범위(Operation Scope)와 다른 객체들과의 상호 연관성등을 고려하여 이루어진다.이러한 기능들은 추상화된 객체가 가지고 있는 자료를 처리할 수 있는 프로시져를 추가함으로써 가능하다.시스템의 데이타 흐름은 구조적 분석(Structural Analysis)과 유사하게 표현 할 수 있다. 그러나 기존의 구조적 분석이 전체적인 시스템에서 세부적으로 나뉘어지는 하향식(Top-Down) 흐름도로써 수직적이고 일괄적인 처리과정을 갖는데 비하여, Bailin의 분석 기법은 최소 단위의 세분화된 개념으로 부터 전체적인 시스템을 구성하기 위한 상향식(Down- Top) 기법을 사용한다[10]. 전처리 단계에서의 객체간의 자료의 흐름 및 가시범위는 Fig. 3-6과 같다.

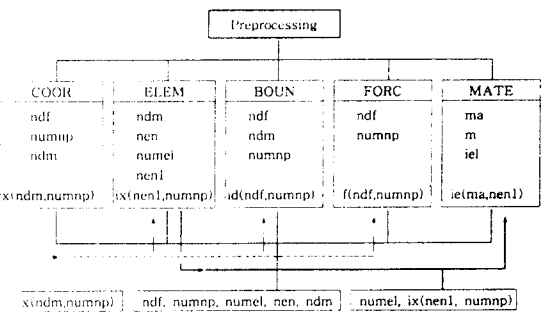

Fig. 3-6 Data Flow by Abstract Modeling

각각의 객체에 대한 모델링이 완성되면 Shlear/Mellor의 정보 모델링 방법을 사용하여 객체의 생명 주기를 정한다. Shlear/Mellor의 정보 모델링

기법은 객체에 대한 속성을 정의함으로서 유사한 것, 상속 가능한 것 등으로 세분화시킨다. 따라서 객체의 생명 주기를 정의할 때에도 속성(Attribute)에 의해 생명 주기를 갖게 설계할 수 있다. 이러한 속성은 프로그래밍에 있어서 속성의 변환으로써 객체의 활동을 다르게 하고 변형된 속성에 맞게 객체의 특성을 정할 수 있다. 본 연구에 있어서 속성 변형을 통한 객체의 활동 내역을 변화시키는 작업을 전처리 단계와 도움말 기능, 그리고 후처리 단계의 그래픽과 텍스트 결과의 출력에 적용하였다. 전처리 단계에서의 객체의 생명 주기는 Fig. 3-7과 같이 나타낼 수 있다.

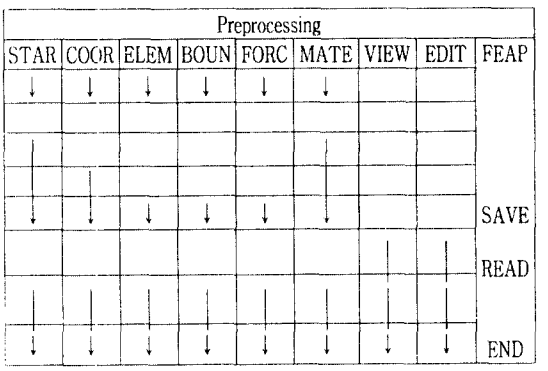

Fig. 3-7 Object Life Cycle in Preprocessing

전처리 단계에서의 객체 생명 주기는 전처리의 서브 영역(Sub Area)에 표시되는 객체들에 한에서 이루어지며 전처리 단계를 종료하는 동시에 객체의 생명 주기도 끝나게 된다. 각각의 전처리 단계에 존재하는 객체들은 자신들의 속성 변환이나 외부 메시지에 의해 생명 주기가 시작되거나 끝나는 경우도 있다. 이러한 객체에 대한 생명 주기 및 프로시저에 대한 자료 보호(Protected) 및 공유(Public)관계를 최종적으로 표현하면 전체적인 프로그램의 처리과정을 유도해 낼 수 있다[11]. 즉, 객체의 생명 주기를 정의하는 것은 자료의 입력을 통한 문제의 해를 구하고자 하는 공학 문제에 있어서 각 단계에서의 세부적인 데이타 입력 순서를 결정할 수 있을 뿐 아니라 데이타의 가시 범위, 그리고 입력 과정에서 나타날 수 있는 에러(Error)을 방지할 수 있는 효과를 얻게 된다.

시스템 전체적으로 객체의 상호 메시지 처리에 의한 생명 부여와 생명 손실의 반복으로써 객체의 기능을 제어할 수 있는 여건을 만들게 된다. 이러한 과정을 통해 객체가 처리해야 할 자료, 활동내역, 생명주기를 갖게 되면 전체적인 통합과 함께 시스템이 완성된다.

3.2. 해석 단계

본 프로그램의 해석 단계에서는 선형(Linear)과 비선형(Non-Linear) 문제를 해석할 수 있고, 주메뉴 영역에 있는 『Solution』을 선택하면 Fig. 3-8과 같이 부메뉴 영역에 『Linear』와 『NLinear』가 생성된다. 동시에 그래픽 표현 영역은 텍스트 표현 영역으로 전환되며 처리 과정에서 나타나는 메시지 및 해석의 결과가 텍스트로 표현된다.

부메뉴 영역의 『Linear』를 선택하면 Fig. 3-9와 같이 입력상자가 나타나고, 여기에 수렴허용치(Tolerance), 반복횟수(Looping Number)등의 텍스트 명령어를 입력하면 해석이 수행된다.

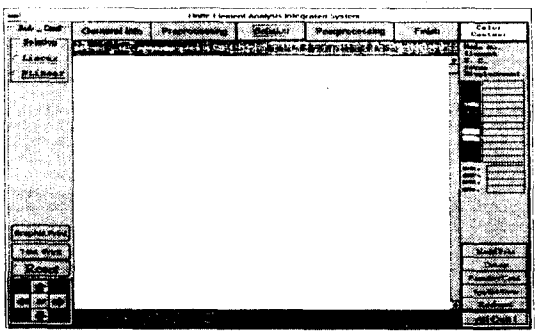

Fig. 3-8 System Driving Screen in Solution Step

『NLinear』를 선택하면 선형의 경우와 달리 하중증분형식(Load Incremental Type)과 변위증분형식(Displacement Incremental Type)의 두가지

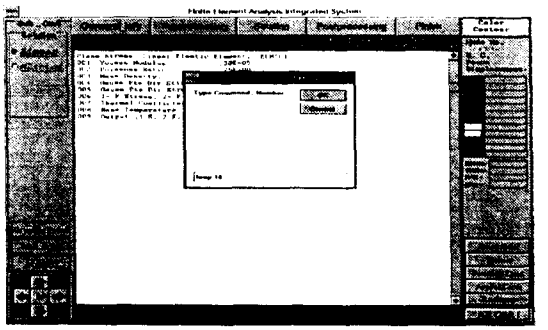

Fig. 3-9 Input Command Box of 『Linear』Menu in Solution Step

메뉴가 생성되고 각각의 경우에 대하여 데이타 입력상자가 나타나게 된다.

문제 해석을 위해 해석 단계에서 필요로 하는 입력이 끝나면 시스템은 해석을 시작하고 해석이 진행되는 동안 순간 순간의 결과값은 텍스트로써 출력된다. 출력되는 결과값은 파일로 동시에 저장되며, 선형 해석의 경우에는 실행이 종료된 후 그리고 비선형 해석의 경우 중간단계에서도 후처리 기능으로 전환되어 결과를 그래픽으로 표현할 수 있다. 비선형의 경우 각 증분마다 실행이 끝나면 다시 대화상자를 통하여 계속적인 작업 여부를 확인하고, 계속 작업을 수행한다면 지금까지의 내부이력변수치(Internal History Variables)를 저장하고

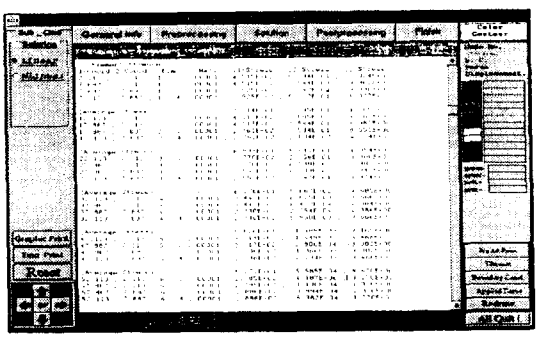

Fig. 3-10 Solution Result in Text Presentation Area During Solution

전단계와 같은 순서로 다시 실행된다. 해석 진행중에 나타나는 결과값은 Fig. 3-10과 같이 텍스트 표현 영역에 나타난다.

3.3 후처리 단계

후처리 단계는 선형해석이 종료된 후, 그리고 비선형 해석이 수행될 경우에는 해석 중간단계에서 결과를 그래픽으로 표현하여 가시적인 결과를 확인할 수 있게 설계되었다. 이러한 방식은 잘못된 중간 입력값이나 해석 방법으로 인한 지속적인 해석시간의 낭비를 방지할 수 있다. 후처리 방식은 구조물의 변형, 응력분포, 반력등에 대한 칼라 칸투어 표현(Color Contour Presentation)으로 나타낼 수 있고, 해석된 결과를 그래프(Graph)로도 표현할 수 있다. 그래픽의 확대(Zoom In) 및 축소(Zoom Out)가 가능하며 Fig. 3-11의 주윈도우상의 좌측 하단에 위치한 4개의 방향키(Direction Key)와 크기 조정(Scale Factor)버튼으로 동작된다. 4개의 방향키 버튼은 각각 윈도우즈의 키눌림(Key Press) 메시지와 좌측 마우스 클릭(Left Mouse Button Click) 메시지에 응답하며 지속적으로 누르고 있으면 그래픽 화면 크기 값이 조건식에 의해 증가하거나 감소하는 원리로 동작한다. 그래픽 결과는 비트맵(Bitmap) 또는 메타파일(Metafile)로 저장할 수 있고 기존의 저장양식(Save Format)과 같은 그림을 불러 올 수 있다.

『Postprocessing』을 선택하면 부메뉴 영역에 『Text』와 『Graphic』이 생성되고, 또 다시 『Graphic』을 선택하면 Fig. 3-11과 같이 『Graphic』아래에 각각의 그래픽 결과를 보기위한 4개의 부메뉴가 생성된다.

텍스트 표현 영역에는 해석 단계의 마지막 결과가 텍스트로, 그래픽 표현 영역에는 전처리 단계에서의 모델링이 그래픽으로 나타난다. Fig. 3-12는 『GR-DISP』를 선택한 후의 변형된 모양을 그래픽 표현 영역에, 동시에 칼라 칸투어 영역에는 각 방향의 최대 변위값을 나타내고 있다.

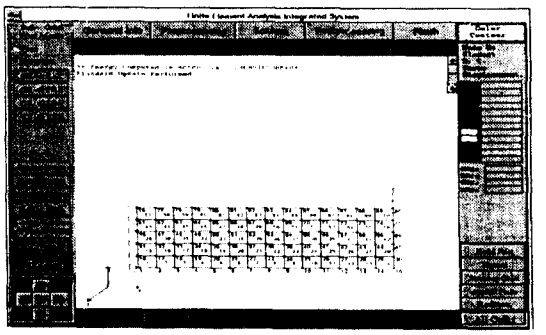

Fig. 3-11 System Driving Screen in Postprocessing Step

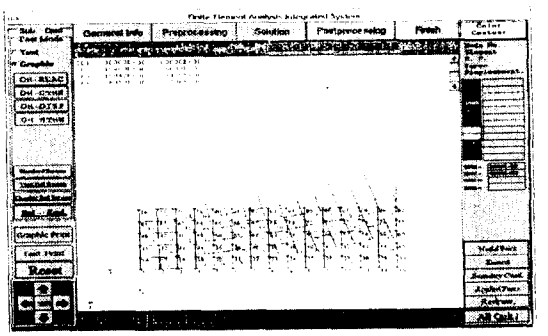

Fig. 3-12 Deformation Configuration in Postprocessing Step

Fig. 3-13은 『GR-STRE』선택하여 나타난 3개의 각 응력 성분 부메뉴중에서 『11-Stress』을 선택한 후의 응력분포를 칼라칸투어로 표현된 모양을 보여주며, 칼라 칸투어 영역에도 색표현 값과 함께 응력의 최대, 최소치가 나타낸 것을 보여 주고 있다.

후처리 단계에서는 객체의 동적 특성(Dynamic Property) 및 메시지 전달에 의한 속성 변형을 이용하여 동일한 객체에 대하여 행동을 달리하는 특성을 가지고 있다. 결과의 표현 방식에 있어서도 전체적인 그래픽 표현, 그래픽과 텍스트의 동시 표현과 전체적인 텍스트 결과를 표현할 수 있는 기능을 갖추고 있다.

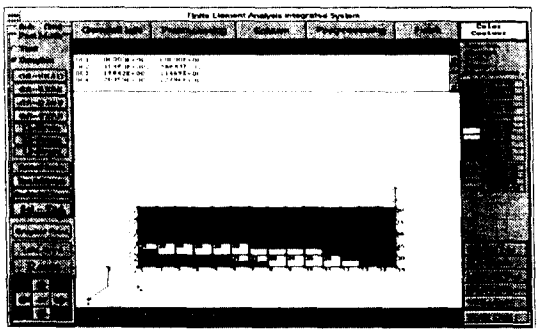

Fig. 3-13 Stress Distribution in Postprocessing Step

4. 결 론

본 연구는 객체지향 시스템 설계에 있어서 일반적으로 C++ 단일언어로 개발되던 방식을 탈피하고, 객체지향 설계 개념에 바탕을 둔 VB와 C++를 사용하여 구조 해석용 유한요소 통합환경 시스템을 설계하였다. 적용 개념에 있어서는 프로그래밍 위주의 객체지향 보다는 전체적인 시스템 설계에서 선행되어야 할 객체지향 접근 방식에 중점을 두었고, 구조해석용 유한요소 프로그램 개발에 있어서 실세계 모델링에 근접한 객체의 추출 및 생성 방법등을 Bailin의 요구 명세 방법론과 Shlear/Mellor의 정보 모델링 기법을 응용하였다. 개개의 객체 모델링을 통한 시스템 구축은 실제적인 시스템의 요소분할 형식을 가짐으로써 사용자의 입장에서 프로그램을 자신에 맞게 수정하거나 개선하기 쉽게 개발되었고 지속적인 개선이 가능하게 설계되었다. 따라서 해석하고자 하는 문제의 요소 라이브러리만을 추가해 주면 해석 범위는 쉽게 확장될 수 있다. 이 때 제작되는 라이브러리는 VB코드를 그대로 적용할 수도 있으며 다중처리 환경을 이용할 경우 DLL로 제작하여 사용할 수도 있다.

또한 GUI의 구현으로 데이타 생성과 동시에 그래픽 모델링 과정을 볼 수가 있어 입력 결과의 인식이 매우 편리해 졌고 데이타 입력 방식도 증분에 의한 자동 데이타 생성을 가능하게 하여 입력 과정을 매우 간결하게 하였다. 윈도우즈 환경이 갖는 사건중심적 프로그래밍 기법은 사용자 중심의 편리성을 유도 하였고 기존 텍스트 입력 방식의 일부를 채용하여 매크로 입력 방식을 사용하던 사용자들도 큰 불편없이 사용할 수 있도록 하였다.

전체적인 시스템 구현에 있어서 기존의 객체지향 이론적 방법을 유한요소 프로그램 개발에 그대로 적용하기에는 다소의 문제점이 있었으며, 적용할 이론이 어디서 부터 유도 되었는지를 파악하는 것이 중요하다고 볼 수 있다. 즉, Shlear/Mellor의 정보 모형을 통한 객체지향 시스템 구축은 많은 외부 장치들을 개개의 요소들로 객체를 형성하여 전체적인 시스템을 설계하는 방법으로 대부분 신호 발생 및 처리에 따른 시간의 단계적 제어등과 같은 시스템 설계에서 사용되는 예가 많았다. 그렇기 때문에 단순히 출력장치나 입력장치를 주로 외부 장치로 간주하는 시스템이나 수치해석과 같이 공학적 문제의 해석을 주로 다루는 시스템 개발에 있어서는 객체지향 분석 방법론을 그대로 적용하기에는 어려움이 있다고 할 수 있다. 그러나 객체 모델링 기법이나 설계 분석기법은 실제 공학용 시스템 설계에 있어서도 근본적으로 의미가 같음을 알 수 있었다.

후 기

본 연구는 (주)통일보일러의 산학협동 연구지원에 의해 이루어 졌음을 밝히며 이 지원에 대해 감사한다.

참 고 문 헌

[1] O.C. Zienkiewicz and R.L. Taylor, The Finite Element Method, Volume 1 and 2, 4th Ed., McGraw-Hill, 1989.

[2] 이 재환 외, "전산구조공학 응용을 위한 조사 연구 : 전산선박 구조공학의 모델링 기술", 대한조선 학회, pp. 93-105, 1993.

[3] Grady Booch, Object-Oriented Analysis and Design with Application, 2th Ed., The Benjamin/Cummings Publishing Company, Inc., 1993.

[4] Shally Shlear & Stephen J. Mellor, Object Life Cycle, Yourdon Press., 1994.

[5] Daniel Appleman, Visual Basic Programmer Guide to the Windows API, Ziff · Davis Press, 1993.

[6] Object Windows for C++ Programmer Guide, Borland International, Inc., 1993.

[7] Gary Entsminger, Visual Basic 3.0 Programming, SAMS, 1994.

[8] 양 해술, "객체지향 패러다임에서의 분석과 설계", Unix C, pp. 120-126, April 1994.

[9] 사 종엽 외, "객체 지향형 프로그램에 의한 유한 요소법 코드 개발에 관한 연구", 영남대학교 공업기술연구소 논문집, 제 22권, pp. 75-86, 1994.

[10] James Martin, Principles of Object Oriented Analysis and Design, Prentice Hall, 1993.

[11] Meyer & Bertrand, Object-Oriented Software Construction, Prentice Hall, 1988.

[12] H.M. Deitel and P.J. Deitel, C++ HOW TO PROGRAM, Prentice Hall, 1994.