객체지향 접근법을 이용한 재무계획모델의 표현

(A Representation of Financial Planning Model
Using Object-Oriented Approach)

이재식·한재홍
아주대학교 대학원 경영학과

Recently, the computer software technology is not keeping pace with the increasing demand of application software development and rapid changes in business environment.

To overcome this "software crisis", many researchers have studied the methodologies to improve the productivity in software development and the flexibility of software usage. Among these methodologies, the most promising and intensively studied methodology is the Object-Oriented Approach.

The purpose of this study is, therefore, to examine the applicability of the Object-Oriented Approach for improving productivity and flexibility in Management Information Systems development. For an application, we selected a financial planning model, especially focusing on the budgeted income statement.

In this thesis, we identified relevent objects in the budgeted income statement, and represented them in the object models. By implementing these object models using C++ language, we evaluated their adaptability in the budget-making process, and showed, by comparing them with the existing Planning & Modeling Languages such as IFPS(Interactive Financial Planning System), their practicability in Management Information Systems.

The results of this study are as follows :

First, the same object models can be used in making the budgeted income statement both in the department level and in the corporate level.

Second, the object models provide the flexibility and extensibility of an application program in case of the changes in business environment.

Third, the Object-Oriented Approach is a practical methodology to improve the productivity and cut down the maintenance cost of application software development.

I. 서론

년간 65%에서 100%의 생산성 증가율을 나타내는 컴퓨터 하드웨어와 통신 기술의 발달로 인한 정보시스템의 광범위한 확산에도 불구하고 년간 5%의 낮은 생산성 증가율을 보이는 소프트웨어 기술은 정보시스템의 광범위한 이용에 큰 애로점이 되고 있다[11].

상대적으로 낙후된 소프트웨어 기술은 각종 소프트웨어의 개발을 지연시키고, 기존에 개

발된 소프트웨어의 유지보수에 많은 시간과 비용을 투자하게 한다. 최근까지 제 4세대 언어, 구조적 프로그래밍 기법 등 프로그램 개발 및 유지보수를 위한 많은 도구(tool)가 등장하였음에도 불구하고, 미국의 경우 많은 개발지연(backlog)이 발생하고 있으며, 전체 소프트웨어 비용중 60%에서 70%에 해당하는 13 억불을 소프트웨어의 유지보수에 투자하고 있다[12]. 이렇게 소프트웨어의 생산성이 낮은 주요원인은 현재 사용하고 있는 프로그래밍 언어와 방법이 소프트웨어의 재사용성을 매우 낮게 하기 때문이다[5][14]. 특히 기업의 경우, 변화하는 경영환경에 적응할 수 있는 소프트웨어 시스템에 대한 기업의 개발욕구를 현 소프트웨어 기술이 충족시키지 못하고 있으며, 개발지연(backlog) 및 기존 시스템의 유지보수가 점차 늘어나고 있는 실정이다.

이러한 상황을 고려할 때 기업내의 소프트웨어 시스템의 설계 및 코딩의 약 60%가 중복이 되고 재사용이 가능하다는 연구보고는 기업에서의 소프트웨어 재사용성에 바탕을 둔 소프트웨어 생산성 향상기법의 중요성을 새롭게 인식시키고 있다. 국내기업의 경우 최근에 들어 소프트웨어 생산성 향상 기술에 관심을 갖기 시작하고 있으나 이를 기업정보시스템의 개발에 실제로 적용하고 있는 기업은 극히 적다[1]. 국내의 이러한 실정에 비추어 볼 때 기업정보시스템내에서의 응용프로그램 개발에 있어 재사용성을 통한 생산성 향상에 관한 연구는 반드시 필요하다.

이러한 소프트웨어의 위기에 벗어나기 위하여 소프트웨어의 재사용성을 높여 생산성을 향상시키려는 연구가 활발히 진행되고 있다. 그러한 연구중 가장 유력한 분야중 하나가 객체지향 접근법(Object-Oriented Approach)이다. 객체지향 접근법은 소프트웨어의 재사용성을 높일 수 있는 데이타 추상화(data abstraction), 상속성(inheritance), 다형성(polymorphism) 등의 특성을 갖는 프로그램 파라다임(paradigm)이다[18].

본 연구는 객체지향 접근법을 이용하여 기업경영정보시스템에 필수적인 응용소프트웨어의 생산성 향상과 환경변화에 따른 적응력을 높이는데 그 목적이 있다. 이를 위해 재무계획 모델을 객체지향 접근법을 이용하여 표현하고 기존의 모델링 방법론과 비교해봄으로써 기업의 응용소프트웨어 개발에 있어서 객체지향 접근법의 유용성을 살펴보고자 한다.

II. 객체지향 접근법

하드웨어 기술이 소프트웨어 기술보다 질적, 양적으로 발전하게된 근본적인 원인을 든다면 모듈화와 재사용기법을 적용시켜 IC(Integrated Circuit)화 했다는 점이다[6]. 따라서 소프트웨어 분야도 소프트웨어 IC를 만들수 있다면 소프트웨어의 위기를 극복할 수 있다는 것이 객체지향 접근법의 기본적인 아이디어이다.

소프트웨어 위기의 본질적인 원인은 현재 사용하고 있는 프로그래밍 언어로 작성된 소프트웨어의 재사용성(reusability)이 매우 떨어지기 때문이다. 이러한 소프트웨어 위기를 해결하려면 결국 소프트웨어를 구성하는 각 부분을 최대한 독립적으로 만들어 원하는 소프트웨어를 구성하기 위하여 필요한 몇가지를 결합하기만 한다면 곧바로 소프트웨어를 얻을 수 있도록 해야 한다. 또 부분적으로 수정할 필요가 있을 경우 그 부분만 수정하거나 교체하도록 되어야 한다. 이러한 것들을 가능하게 하는 것이 객체지향 접근법(Object-Oriented Approach)이다.

객체지향 접근법은 각종 시스템과 소프트웨어 개발의 복잡성(Complexity)을 지금까지의 알고리즘 분해(Algorithmic Decomposition), 절차지향 프로그래밍(Procedure-Oriented Programming) 대신에 객체지향 분해(Object-Oriented Decomposition), 객체지향 프로그래밍(Object-Oriented Programming)을 이용하는 방법론이다. 절차지향 접근법(Procedure - Oriented Approach)은 데이타(Data)와 절차(Procedure)로 분리되어 알고리즘(Algorithm) 위주로 표현을 하는 반면에, 객체지향 접근법(Object-Oriented Approach)은 데이타와 절차가

합쳐진 객체(Object)와 비슷한 특성을 가진 객체를 포함하는 클래스(Class)를 기본단위로 하여 표현한다[7].

이러한 객체 모델은 프로그래밍 언어, 시스템 디자인, 데이타베이스, 지식베이스, 심지어 컴퓨터 아키텍처에 이르기까지 시스템의 복잡성에 대처하기 위하여 이용되고 있다.

객체지향 접근법은 객체지향 분석(Object-Oriented Analysis), 객체지향 설계(Object-Oriented Design), 객체지향 프로그래밍(Object-Oriented Programming)로 크게 나눌 수 있다[17]. 이 세가지 개념들 사이의 관계를 살펴보면, 객체지향 분석의 결과는 객체지향 설계가 가능하도록 풀고자 하는 문제를 클래스와 객체의 측면에서 명세화한 것이고, 객체지향 설계의 결과는 객체지향 프로그래밍으로서 완전하게 시스템을 구축할 수 있는 청사진으로 이용된다.

III. 재무계획모델

재무계획(Financial Planning)은 각 기업에서 어떻게 사용하느냐에 따라 여러가지로 정의될 수 있으나 그 내용을 분류할 때 크게 두가지로 분류된다. 첫째는, 기업의 손익 상태에 촛점을 두고 수익 및 비용에 관련된 항목에 대한 예측 및 추정을 주로 하는 분야와 둘째로, 손익활동과 관련하여 장기적인 시설투자 및 단기적인 영업활동에 필요한 자금지출 및 영업활동에서 발생하는 자금수입 등에 대한 추정을 하는 등 재무적인 측면이 강조된 분야로 크게 나눌 수 있다. 이러한 점을 고려할 때 재무계획 분야에서 모델은 큰 의미를 갖는다. 즉, 재무계획 모델은 다양한 조건과 여러 투자전략하에서 미래 계획기간동안의 순이익과 현금흐름에 대한 추정을 할 수 있게 한다. 그리고 이러한 모델은 각 재무적 변수들의 상호작용의 효과와 이로 인한 기업에의 영향을 측정할 수 있게 한다. 또한 모델을 이용하므로써 많은 시간과 인력 그리고 비용을 들이지 않고 아이디어나 대체안들을 실험해보고 평가해 볼 수 있다.

클라인(Klein)에 의한 기업에 있어서 재무계획 모델의 이용과 개발실태에 관한 조사연구 결과를 보면 조사대상 기업의 86%가 여러가지 재무계획 모델을 사용하고 있는 것으로 나타나 있다[15]. 응답기업들은 모델의 적용 영역으로서 주로 재무예측과 추정재무제표의 작성을 들고 있다. 이렇게 재무계획모델이 많은 기업으로 확산하게 된 동기는 모델을 쉽게 구축할 수 있고, 이해하기 쉽도록 만들어진 소프트웨어 패키지(package) 때문이다.

재무계획모델의 소프트웨어 패키지는 크게 범용 프로그램 언어(General Purpose Programming Language), 스프레드쉬트(Spreadsheet), 계획 및 모델링 언어(PMLs : Planning and Modeling Languages)를 이용하여 개발된다. 이들 개발도구의 특성을 비교해보면 <표 1>과 같다.

<표 1> 재무계획모델 개발도구의 특성 비교

	범용 프로그램 언어	스프레드쉬트	계획 및 모델링 언어
적용범위	Low	Middle	High
이용의 용이성	High	Middle	Low
모델구축의 용이성	Low	Middle	High
모델의 적응력	Low	High	High

네일러(Naylor)는 재무계획모델이 1970년대에 실패하게된 요인으로 모델의 적응성(fiexibility), 빈약한 문서화, 과도한 양의 입력 데이타를 지적하였다[19]. 애그로올(Aggarwal)과 케러(Khera) 또한 모델의 적응성(flexibility) 부족을 지적하였다[4]. 또,

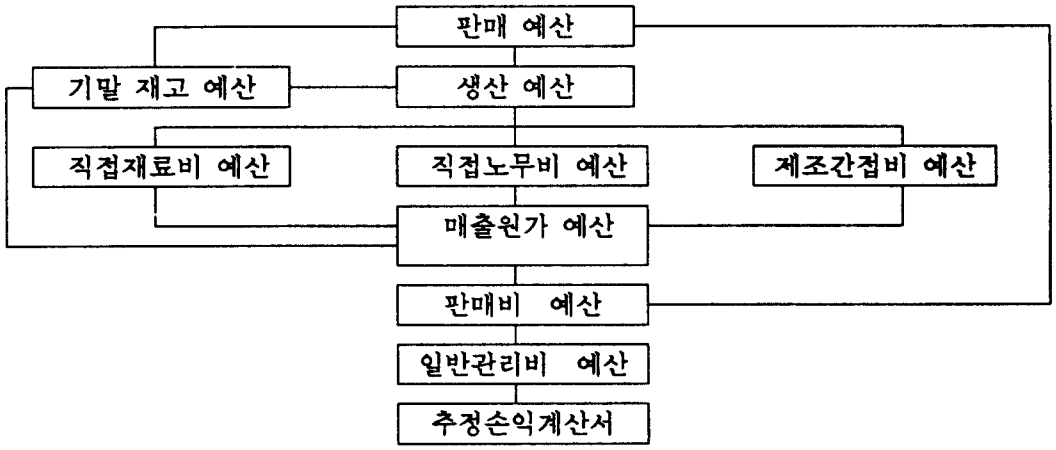
1984년 미국의 대기업을 대상으로 한 재무계획 모델의 이용실태 조사에 따르면 31개 모델 중 29개 모델이 3년이상 연속되어 사용되지 못하고 새롭게 개발되었는데 그 이유로는 충분한 적응성(flexibility)의 부족을 가장 크게 들고 있다[21].

위의 내용을 종합해 볼 때 기업내에서의 재무계획 모델이 성공적으로 정착되기 위해서는 모델구축의 용이성과 환경변화에 대응하는 충분한 적응성이 있는 모델의 구축이 반드시 필요하다.

재무계획 모델은 앞에서 살펴 보았듯이 재무예측, 추정재무제표 작성등 여러 분야에서 적용되고 있다. 기업의 추정손익예산 또한 재무계획 모델 적용분야중의 하나이다.

예산(Budget)은 통상 기업을 포함한 조직체가 계획하고 있는 행위에 대한 계량적 표현방식이다. 이 예산은 조직전체 또는 하위부서별로 세워질 수 있다. 그중 총괄예산(Master Budget)은 모든 하위부서의 목적(예산)을 요약한 것이다.

기업에 있어서의 총괄예산은 판매, 생산, 마케팅, 재무, 인사 등과 같은 하부조직의 목적이 반영되어 있는 예산을 말한다. 총괄예산은 대체로 운영예산(Operating Budget)과 재무예산(Financial Budget)으로 크게 나눌 수 있는데, 운영예산은 희귀자원의 획득과 사용에 주안점을 두는 반면에 재무예산은 자원을 획득할 수 있는 자금을 어떻게 조달하느냐에 촛점을 맞추고 있다.



<그림 1> 추정손익 예산 모델

본 연구는 운영예산과 그 결과인 추정손익예산 모델에 국한하여 살펴 보고자 하는데, 추정손익예산은 <그림 1>과 같은 하위예산과 단계들을 가지고 있다. 판매예산, 생산예산, 직접재료비 예산등과 같은 하위예산들은 서로간에 상호작용을 하고 있으며 최종적으로는 추정손익예산을 지원한다.

IV. 객체지향 접근법을 이용한 추정손익예산의 객체 모델 표현

1. 추정손익예산의 객체 지향 분석

추정손익예산의 객체 모델은 다음 <표 2>와 같이 구성된다.

여기서 앞절의 <그림 1>의 추정손익예산 모델과는 다르게 예산(Budget) 객체가 추가 되었는데 그 이유는 나머지 10개의 예산 객체모델에서 공통적으로 쓰이는 데이타를 가지고 있는 기반클래스(Base Class)의 필요성 때문이다.

추정손익예산 객체 모델들은 다음 2가지 기준에 의하여 설계되었다.

첫째, 추정손익예산의 객체 모델은 최대한 일반화되어야 한다. 그 이유는 기업 예산편성

요인의 다양성 때문이다. 따라서 추정손익예산의 객체 모델은 모든 기업 예산편성에 공통되는 요소들만으로 구성된 모델이 되어야 한다. 개별기업 예산편성의 특수성은 기존의 객체 모델을 쉽게 확장시킬 수 있는 객체지향 접근법의 상속성 특성을 통하여 극복될 수 있다.

둘째, 추정손익예산을 구성하는 각각의 객체 모델은 가능한 상호독립적이어야 한다. 그 이유는 기업내 예산편성과정에 기인하는데 기업내 전체예산은 하위부서들의 예산을 반영하여 편성되기 때문이다. 따라서 추정손익예산의 객체 모델들은 전체 예산편성과정뿐만 아니라 부서별 예산편성과정을 지원해야 되기 때문에 타부서 예산과의 관계를 최소화하는 모델이 되어야 한다. 이 기준은 객체지향 접근법의 정보은폐 특성으로 만족될 수 있다.

<표 2> 추정손익예산 객체모델의 구성

예산(Budget) 객체
판매 예산(Sales Budget) 객체
생산 예산(Production Budget) 객체
직접재료비 예산(Direct Material Budget) 객체
직접노무비 예산(Direct Labor Budget) 객체
기말 재고 예산(Ending Inventory Budget) 객체
제조간접비 예산(Factory Overhead Budget) 객체
매출원가 예산(Cost of Goods Sold Budget) 객체
판매비 예산(Marketing Expense Budget) 객체
일반관리비 예산(Administrative Expense Budget) 객체
추정손익 계산서(Budgeted Income Statement) 객체

각 객체모델을 앞에서 제시한 모델 설계의 두가지 기준에 의해 표현하기 위해서는 각 하위예산의 속성과 예산들간의 상호관계를 파악하여야 한다. 예를 들어보면 <그림 2>는 직접재료비 예산의 속성을, <그림 3>은 직접재료비 예산과 다른 예산과의 상호관계를 보여주고 있다.

제품이름	재료소요량(제품 1단위당)1-M	생산량	재료소요량 1-M

재료소요량(m) = 단위당 재료소요량(m) * 생산량

재료이름	단위당 구매가격	재료총수요량	재료비용

재료총수요량(m) = 제품 1 - N 의 재료소요량(m)
재료비용 = 단위당 구매가격 * 재료총수요량

<그림 2 > 직접재료비 예산(Direct Material Budget)의 속성

데이타	데이타 전달자
제품이름	생산 예산
재료소요량 (제품단위당) 1-M	예산
생산량	생산 예산
재료이름	예산 기말재고 예산
단위당 구매가격	예산 기말재고 예산

<그림 3 > 직접재료비 예산(Direct Material Budget)과 다른 예산과의 상호관계

<그림 2>와 같은 하위예산의 속성은 모델 설계의 일반화 기준을 만족시키고 있으며, 객체에 내장될 데이타구조(Data Structure)와 데이타간의 상호관계의 밑바탕이 된다.

<그림 3>과 같은 예산간의 상호관계는 모델설계의 상호독립성 기준에 따르고 있으며, 이들 예산간의 관계는 데이타의 전달이라는 관점에서 데이타의 전달자와 데이타의 수신자로 구분할 수 있고, 객체 지향 접근법의 메세지의 전달이라는 관점에서는 메세지 전달자와 메세지 수신자로 구분할 수 있다. 메세지의 전달과 데이타의 전달간에는 逆關係가 존재하는데, 예를 들어 메세지를 전달하는 객체는 원하는 데이타를 전달받는 데이타 수신자가 된다. 본 연구에서는 데이타와 데이타 전달자만으로 그 관계를 표현한다.

각각의 객체 모델은 필요한 데이타를 위에서 서술한 데이타 전달자와 사용자들의 입력에 의해서만 전달받는다. 각 객체 모델들이 데이타를 전달받는 방법(어떤 전달자를 선택할 것인지 또는 사용자로부터 입력을 받을 것인지)은 응용 프로그램의 필요정도에 따라 선택할 수 있다.

2. 추정손익예산의 객체 모델 표현

본 절에서는 앞서 언급한 객체 모델의 특성요인과 객체간의 관계를 중심으로 추정손익예산의 객체 모델을 표현한다. 각각의 객체는 데이타의 구조를 나타내는 데이타 영역과 이 데이타 영역에 접근하는 함수(Function)들의 집합으로 표현할 수 있다. 객체의 데이타 영역은 객체자신의 고유한 특성을 나타낼 수 있는 데이타로 구성되며 이 데이타의 접근은 사전에 정의된 함수(Function)에 의해서만 가능하다.

객체의 함수는 객체내의 데이타를 처리(Process)하기 위해 정의된 것이며 추정손익예산의 객체에는 객체의 생성자(또는 생성함수), 객체의 소멸자(소멸함수), 저장함수, 출력함수, 데이타 전달함수가 사용된다. 생성자 함수는 객체의 인스탄스(instance)를 생성하는 함수로 같은 이름의 함수가 존재할 수 있으며, 소멸자 함수는 객체의 인스탄스를 소멸시키는 함수이다. 저장함수는 객체내의 데이타를 화일에 기록하는 함수이고, 출력함수는 객체내의 데이타를 정해진 출력형태에 따라 화면에 출력하는 함수이다. 데이타 전달함수는 다른 객체에서 메세지를 보내면 원하는 데이타를 전달하는 함수이다. <그림 4>는 직접재료비 예산 객체모델을 표현한것이다.

직접재료비 예산(Direct Material Budget) 객체
데이타 영역 [제품명, 단위당 재료소요량 (1-m), 생산량, 재료소요량(1-m)] [N] [재료명, 단위당 구매가격, 재료소요량, 재료비용] [M]
함수 생성자 1 (사용자 입력) 생성자 2 (타 객체로부터의 입력) 생성자 3 (화일 입력) 소멸자 저장 함수 출력 함수 데이타 전달함수 1 (재료별 소요량) 데이타 전달함수 2 (재료별 재료비용)

<그림 4> 직접재료비 예산 객체 모델 표현

V. 추정손익예산 객체모델의 구현 및 응용

1. 추정손익예산 객체 모델의 구현

본 연구에서는 추정손익예산 객체 모델의 구현을 위해 객체지향 프로그래밍 언어인 C++

를 사용하였다. 객체지향 프로그래밍 언어로 많이 알려진 것으로는 Smalltalk-80, C++, Ada 등이 있으나, 이중 C++를 선택한 이유는 일반적으로 널리 알려진 C 언어의 특성을 이용할 수 있다는 점과 작업수행환경에 큰 영향을 받지 않는 실행화일을 생성할 수 있다는 점 때문이다.

C++를 이용한 객체 모델의 구현은 객체가 가져야 하는 특성(데이타 구조, 함수)을 기술한 클래스(Class)를 정의하는 것이다. 클래스에서 정의하는 데이타 멤버는 클래스를 하나의 데이타 타입으로 하여 정의되는 변수에 저장되는 데이터의 구조를 지정하는 일을 하고, 멤버 함수는 인스탄스의 데이타 멤버에 저장되어 있는 값을 처리하여 정보를 만들어 주는 일을 수행하게 된다. 보통 데이타 멤버와 멤버 함수를 통칭하여 클래스의 멤버(member)라고 한다

클래스의 정의에는 각각의 멤버에 대하여 접근 권한(Access Right)을 지정할 수 있도록 하였는데, 이것은 객체지향 프로그래밍의 데이타 추상화(data abstraction)의 정도를 나타내는 것이다.

직접재료비 예산 객체를 구현한 Source Code는 <그림 5>와 같다.

```
class Mat_Budget  {
    protected :                                    // 접근권한
       struct PM_Budget {                          // 데이타 구조 선언
              char*          Pname ;
              unsigned long  Amount_prod ;
              float*         Amount_mat ,  Amount_Tmat ; } ;
       struct M_Budget {
              char*          Mname ;
              float          unit_pur , Total_mat , Cost_mat ; } ;
       PM_Budget*    PMbudget, PMpoi ;
       M_Budget*     Mbudget,  M_Budget* Mpoi ;
    public :
          Mat_Budget(int,int) ;                    // 생성자1 함수
          Mat_Budget(Materials*, Pro_Mat_H*, int, int, unsigned long*) ;
          Mat_Budget(char*) ;                      // 생성자 2, 생성자 3
          ~Mat_Budget() ;                          // 소멸자
          void output() ;                          // 출력함수
          void save() ;                            // 저장함수
          int Return_pmax() {return pmax; } ;      // 데이타 전달함수
          int Return_mmax() {return mmax; } ;        .......
```

<그림 5> 직접재료비 예산 객체의 구현

2. 추정손익예산 객체 모델의 응용

C++를 이용하여 구현된 추정손익예산 객체 모델은 필요에 따라 여러가지 응용 프로그램(application program)의 구축에 이용될 수 있다. 개별 객체의 생성자 함수를 이용하여 다음 <그림 6>, <그림 7>과 같이 간단히 필요한 응용 프로그램을 구축할 수 있다. <그림 6>은 개별 부서의 예산편성을 위한 응용 프로그램의 예이고, <그림 7>은 기업 전체 예산 편성을 위한 응용 프로그램의 예이다. 이 두가지 예에서 보는바와 같이 필요한 응용 프로그램은 프로그램별로 모든 코드를 작성하지않고 필요한 객체들의 함수를 선언함으로서 개발할 수 있다.

따라서 응용 프로그램 개발자는 객체의 공개된 외부기능만을 알고도 쉽게 원하는 응용 프로그램을 구축할수 있다.

```
main () {
        int p ;  cin >> p ;
        Prod_Budget production(p) ;
        production.output() ;
        production.save() ;   }
```

<그림 6> 생산 예산을 위한 응용 프로그램

```
main() {
    Budget budget(pn, mn) ;          // Budget Object의 instance 인 budget
                                        의 생성, 생성자 1을 이용
    Sales_Budget sales (budget.Return_s(), budget.Return_smax() ) ;
                  // Sales_Budget의 instance인 sales의 생성, 생성자 2를 이용
    Prod_Budget prod (budget.Return_s(), budget.Return_smax(),
                      sales.Return_sales() ) ;
    Mat_Budget mat (budget.Return_m(), budget.Return_p(), budget.Return_smax(),
                    budget.Return_mmax(), prod.Return_Amount_prod() ) ;
          .....................................
    Inco_Budget income(p, sales.Return_revenue(),cost.Return_Cost_Goods_Sold(),
                       mark.Return_marketing_expense(),
                       admin.Return_admin_expense() ) ;
    budget.output() ;  }          // Object budget의 출력
```

<그림 7> 추정손익예산 응용 프로그램

개별기업 예산편성의 특수성에 대한 적응력(flexibility)을 높이기 위하여 기존의 클래스로 부터 파생된 클래스의 정의가 <그림 8>에 예시되어 있다. 직접재료 구매 예산 객체 클래스는 기반 클래스인 직접재료비 예산 객체로 부터 파생한 클래스로서 기존의 데이타에 재료구매량과 재료 구매비용에 대한 데이타를 첨가시킨 클래스이다. 이 예에서 볼수 있듯이 객체의 상속성(inheritance) 특성을 이용하여 기반 클래스(Base Class)에서 예산편성 과정의 변화, 또는 예산편성의 특수성을 반영한 파생 클래스(Derived Class) 를 쉽고 신속하게 정의할 수 있다.

이러한 특성은 객체 사이의 상호독립성을 최대한 보장함으로써 환경변화에 따른 영향을 최소화할 수 있고 소프트웨어의 적응력을 향상 시킬수 있다.

```
class Mpur_Budget : public Mat_Budget  {  // Mpur_Budget class가 Mat_Budget
      protected :                            에서 파생되었음
                struct MP_Budget {
                                int          end_Minv ;
                                float        M_need ;
                                int          int_Minv ;
                                float        Amount_purchase ;
                                float        PurchaseCost ; } ;
                          MP_Budget* MPbudget ;
                          MP_Budget* MPpoi ;
     ........
Mpur_Budget::Mpur_Budget(int p, int m) : Mat_Budget (p,m) {
                          // ....................      } ;
       // Mpur_Budget이 Mat_Budget으로부터 파생된 클래스 임으로 Mpur_Budget
          의 instance를 생성하기 전에 Mat_Budget을 생성한다.
.....
```

<그림 8> Material Budget Object 클래스로부터 파생된
Material Purchase Budget Object 클래스의 정의

3. PLANNING AND MODELING LANGUAGES로 구축된 모델과의 비교

계획 및 모델링 언어는 English-Like 프로그래밍 언어로서 재무용어를 직접 이용하여 모델을 쉽게 구축할 수 있으며, 데이타처리 시간을 줄일 수 있고 상위경영층이 모델을 쉽게 이해할 수 있다는 장점이 있다. 그러나 데이터의 입력과 출력에 있어 많은 제약이 있으며, 일반적인 프로그래밍 언어보다는 개선은 되었으나 여전히 환경변화에 따른 모델의 적응력과 확장성에 문제를 가지고 있다.

계획 및 모델링 언어중 국내에서 많이 알려진 IFPS(Interactive Financial Planning System)를 이용한 추정손익예산 모델과 본 연구에서 구현된 추정손익예산 객체 모델을 비교해 보았다. 그 결과 IFPS를 이용한 모델 구축은 위에서 언급한 PMLs의 장점을 가졌음에도 불구하고 기존의 재무계획 모델을 구축하는 소프트웨어 팩키지가 갖고 있는 문제점, 즉 모델의 적응력(flexbility)과 확장성(extensibility)은 완전히 극복하지 못하고 있다. 예를 들어 판매하는 제품이 하나 늘었을 경우 IFPS를 이용한 추정손익예산 모델은 모델 전 부분에 걸쳐 추가 또는 수정이 필요하였다. 또한 IFPS를 이용하여 부서별 예산 편성을 위한 모델을 구축할 때 전체 예산 편성 모델의 코드(code)를 이용하지 못하고 다른 모델 이름으로 같은 코드(Code)를 재작성하여야만 한다. 이러한 방법은 많은 시간과 비용를 발생하여 소프트웨어의 생산성을 떨어 뜨린다. 그러나 앞절에서 볼 수 있듯이 객체지향 접근법을 이용하여 구축한 모델은 전체예산 편성과정 뿐만 아니라 부서별 예산편성에 모델의 수정 없이 그대로 이용할수 있다. 또한 <그림 8>에서 볼 수 있듯이 객체 모델의 특성인 상속성(inheritance) 은 부서별 예산 편성시 고려되어야 하는 개별 특성을 손쉽게 모델화할 수 있으므로 모델의 재사용성을 증가시킬 뿐만 아니라 궁극적으로 소프트웨어의 생산성을 향상시킬 수 있다.

VI. 결론

본 연구의 목적은 기업내의 응용소프트웨어 개발에 객체지향 접근법을 이용하여 기업 정보시스템에 필수적인 응용 소프트웨어의 생산성 향상과 환경 변화에 따른 적응력을 높이기 위한 것으로써, 그 구체적 목적은 첫째, 재무계획 모델중에서 추정손익 예산을 객체지향 접근법의 핵심요소인 객체모델(Object Model)로 표현하고 구현하는 것이고 둘째, 이 객체모델을 예산편성 과정에 적용해보고, 계획 및 모델링 언어(Planning & Modeling Languages)로 구현된 모델과 비교하여 기업 경영정보시스템 개발에 있어 객체지향 접근법의 활용가능성을 모색해 보는 것이었다.

결론적으로 객체지향 접근법을 이용한 추정손익예산 객체모델은 IFPS를 이용한 추정손익예산 모델에 비해 다음과 같은 장점이 있다.

첫째, 추정손익예산 객체모델은 모델의 재사용성을 향상시키고 결과적으로 응용프로그램 개발의 생산성을 향상시킨다.

둘째, 추정손익예산 객체모델은 환경변화에 빠르게 적응할 수 있다. 따라서 유지보수에 드는 시간과 비용을 절감할 수 있다.

셋째, 추정손익예산 객체모델은 전체 예산편성과정 뿐만 아니라 부서별 예산편성과정을 추가 작업 없이 지원할 수 있다. 따라서 기업의 예산편성에 필요한 전체와 부서별 예산편성 과정의 조화를 충분히 지원할 수 있다.

이러한 점들을 고려해 볼 때 소프트웨어 생산성 향상과 유지보수 용이성이 절대적으로 필요한 기업의 경영정보시스템 설계 및 구축에 있어서 객체지향 접근법의 활용 가능성은 매우 높다.

참고문헌

[1] 이광순, "국내기업정보시스템 3단계초 수준", 경영과 컴퓨터, 1991.9.

[2] 하수철, "C++ 객체 지향 프로그래밍을 위한 도형화 기법", 홍익대학교 박사 학위논문, 1990.

[3] Abott R. J., "Program Design by Informal English Descriptions", Communication of the ACM, Vol.26, No.11, Nov., 1983, pp. 882-894.

[4] Aggrawal R., and I. Khera, "Using Management Science Models : A Critique for Planners", Managerial Planning, Vol. 28, No. 4, 1980, pp. 12-15.

[5] Apte U., C. S. Sanker, M. Thakur and J. E. Tuner, "Reusability based Strategy for Development of Information Systems : Implementation Experience of a Bank," MIS Quarterly, Dec., 1990, pp. 421-433.

[6] Booch G., "Object-Oriented Development", IEEE Transaction on Software Engineering, June, 1985, pp. 211-221.

[7] Booch G., Object Oriented Design with Application, The Benjamin / Cummings Publishing Company, 1991.

[8] Cox B. J., "Object-Oriented Programming," SIGPLAN Notices, Vol.17, No.9, Sep., 1982, p. 51.

[9] Geoffry S. Howard, "Object Oriented Programming Explained," Journal of Systems Management, July, 1988, pp. 13-19.

[10] Glassey C. R. and S. Adiga, "Berkely Library of Objects for Control and Simulation of Manufacturing (BLOCS/M)", Applications of Object-Oriented Programming, Addison-Wesley, 1990.

[11] Gruman G., "Early Reuse Practice Lives Up To Its Promise," IEEE Software, Nov., 1988, pp. 87-91.

[12] Hanson S. J. and R. R. Rosinski, "Programmer Perceptions of Productivity and Programming Tools," Communications of the ACM, Feb., 1985, pp. 180-189.

[13] Higgins J.C., and R. Finn, "Managerial Attitudes Toward Computer Models for Planning and Control," Long Range Planning, Sep., 1976, pp.107-112.

[14] Karami Jahanir, "An Asset-based Systems Development Approach to Software Reusability," MIS Quarterly, June, 1990, pp. 179-199.

[15] Klein R., "Computer-Based Financial Modeling," Journal of Systems Management, Vol.33, 1982, pp. 6-13.

[16] Laursen J. and R. Atkinson, "OPUS : A Smalltalk Production System," OOPSLA'87, Orlando Florida, 1987, pp. 377-387.

[17] Mellor S., A. Hecht, D. Tryon, and W. Hywari, "Objecte-Oriented Analysis: Theory and Practice, Course notes, in Object-Oriented Programming Systems, Languages, and Applications," San Diego, CA: OOPSLA'-88, Sep., 1988, p. 13.

[18] Meyer B., "Reusability : The Case for Object-Oriented Design," IEEE Software, Mar., 1987, pp. 50-64.

[19] Naylor T. H., "Strategic Planning Models", Managerial Planning, Vol.30, No. 1, 1983, pp. 3-11.

[20] Sefik M., and D. Bobrow, "Object-Oriented Programming : Themes and Variations," AI Magagine, Vol.6(Winter), 1986, p.41.

[21] Shim J. K and J. G. Siegel, Handbook of Financial Analysis, Forecasting, and Modeling, Prentice Hall, 1988, pp. 348-350.

[22] Sommerville I., Software Engineering, 2nd ed, Workingham, England: Addison-Wesley, 1985.

[23] Thomas D., "What's in an Object," Byte Magazine, March, 1989, pp. 231-24