Journal of the Korean Institute
of Industrial Engineers
Vol.24, N0.3, September, 1998

Option을 고려한 객체지향형 Product Structure 설계

The Object-Oriented Design of the Product Structure Based on Options

고석완* · 김선호** · 정석찬***

Suk-Wan Ko* · Sunn-Ho Kim** · Seok-Chan Jeong***

Abstract

As a product structure or BOM(bill of material) of products is hierarchically structured, the design based on the concept of relational data base modeling causes low performances in data search or processing. For this reason, an object-oriented approach to designing a product structure is proposed in this paper. Using Rumbaugh's OMT (Object Modeling Technique) method, classes of parts, BOM structure, options, and models are defined and their class-relationship diagrams are proposed. For the representation of the BOM structure suitable for the object-oriented paradigm, a new data architecture called the BOM item class is suggested. It is expected that the proposed data structure ensures better reusability and expandability due to the modularity.

1. 개요

일반적으로 PDM이나 MRP시스템에서는 부품에 대한 관리를 체계적으로 하기 위해서 제품구조(product structure)를 BOM(Bill of Material)이라는 체계로 생성하고, 이를 통해 부품을 관리하고 있다. PDM에서는 엔지니어링 BOM(E-BOM)을 관리하며, MRP에서는 제조 BOM (M-BOM)을 관리한다. 이러한 BOM은 시스템에서 기본적인 기능의 구현에 필수적인 요소이며, 이것을 어떻게 관리하는지에 따라 시스템의 성능에 영향을 미치게 된다.

BOM을 현장에서 제대로 운용하기 위해서는 일반적인 BOM의 구조를 효율화시켜 나가지 않으면 안된다. 즉, 모델이 다양해지고 선택사양이 많아지면 제품의 종류는 기하급수적으로 많아지며 관리해야할 BOM도 제품종류별로 모두 관리해야하는 불편함이 있다. 또한, 한 부품이나 구성품을 설계변경하게 되면 모든 BOM을 일일이 수정하여야 하는 불편함이 따르게 된다. 이러한 문제점을 해결하기 위하여 가상부품, feature, option, 대치품, 부산물이나 불량품 등을 복합적으로 고려한 BOM이 필요하다.

BOM 데이터의 구조는 tree형태로 복잡하여 관계형 DB로 구현하는 것보다 객체지향형 DB로 구현하는 것이 더욱 바람직하다. 관계 데이터 베이스와 비교해 볼 때, 객체지향 데이터 베이스에서는 구조들이 객체들 사이에 참조가 직접 저장됨으로 좀더 빨리 복잡한 구조를 조회할

* 대우정보시스템(주)

** 명지대학교 산업공학과

*** ETRI 컴퓨터 · 소프트웨어기술연구소

수 있다. 객체지향 데이터 베이스는 테이블을 통해 검색하므로, 관계형 데이터 베이스에서처럼 관계를 재구성할 필요가 없다[8]. 객체지향 데이터 베이스는 제품을 복합 객체로서 저장하며, 각 구성 요소에 대한 링크로 직접 참조를 나타낸다. 이렇게 표현하면 데이터 베이스 관리자가 직접 구성 요소 객체를 참조할 수 있으므로, 테이블을 검색하여 찾는 노력을 최소화할 수 있다. 객체지향 데이터 베이스에서는 관계형 데이터 베이스보다 훨씬 시간이 덜 걸리면서도 완벽하게 BOM 정보를 검색할 수 있다.

그동안 이 분야와 관련된 외국의 연구 사례를 보면, BOM의 구조(BOM structure)와 부품의 특성(part property)에 대해 객체지향 방식으로 개념적인 모델링을 제시하거나 객체지향 방식으로 BOM 구조와, 데이터를 처리하는 transaction 모델들을 설계하였다[2,5]. 이 외에도 관계형 DB를 이용하여 BOM processor의 효율성을 분석하였고[1] 데이터 구조가 복잡한 엔지니어링 분야에서 여러 가지로 제시된 version에 대한 개념을 통합 정리하기도 하였다 [10]. 또한, 국내의 연구 사례를 보면, 대부분이 관계형 DB 하에서 BOM의 구조나 효율성에 대해 언급하고 있다[17,18,19,20,25,26]. 객체지향 방식으로 BOM과 기술자료간의 통합을 시도한 연구도 있었다[22]. 그러나 그동안의 연구는 BOM구조나 프로세서 기능 등에 대해서는 간단한 개념만 제시하였고 실제 운용할 수 있는 수준으로 고려되지 않은 단점이 있다.

기존의 상용화된 PDM tool [12,13]들은 product structure와 configuration을 관리하는 기능이 있으나 가상부품, feature, option, 대치품, 부산물이나 블랑품 등과 같은 사항들을 전혀 고려하고 있지 못하여 사용자가 이러한 기능을 다시 개발해야하는 불편함이 있다. 또한, 저장방식으로 관계형 DB를 이용하고 있어 데이터 운용에 비효율적이다.

product structure와 configuration 관리에 관한 개념 및 기준으로서 ISO 10303의 Part 44 에 대한 표준이 제정되었다 [3]. 이것은 STEP을 위한 기준이며 기본적인 BOM 사항들만 정의 되어 있고, 가상부품, feature, option 등 세부적인 사항이 고려되고 있지 않다.

1990년대 이후 컴퓨터 하드웨어의 급속한 발전에 따라 소프트웨어의 기술적인 수준과 하드웨어의 기술적 수준과의 격차가 심하게 벌어지게 되었다. 이러한 소프트웨어의 구현기술의 발전속도를 향상시키기 위한 노력의 결과로서 객체지향 기법이 대두되게 되었다. 이 기법은 소프트웨어의 생산성과 재사용성을 향상시키고, 보다 실세계에 근접한 개념으로 프로그램의 설계와 구현을 가능하게 하였다. 이러한 객체지향 기법은 프로그램 개발 분야에서 시작되었지만 점차 발전하면서 객체 지향 방법론으로 확대되었고 이를 통해 현실세계의 분석과 재설계를 위해서도 사용되고 있다. [24]

본 연구에서는 객체지향 기법을 이용하여 option을 고려한 BOM 관리시스템을 설계하였다. 우선, BOM 관리 시스템의 객체지향적인 설계를 위해서 본 논문에서는 객체지향 방법론 중의 하나인 Rumbaugh의 OMT(Object-Modeling Technique) 기법을 사용하였다. 여기서 BOM 관리 시스템에서 사용되어지는 부품객체, 속성객체, 구조객체 등의 객체들을 정의하였다. 다음으로 정의된 각 객체들의 연관 관계를 기반으로 객체간의 구조를 정의하고 정의된 전체 구조를 객체지향 모델링 기법인 OMT 기법의 다이어그램을 사용하여 나타내었다.

2. 연구동향

2.1 BOM의 연구동향

관계 데이터 베이스와 비교해 볼 때, 객체지향 데이터 베이스에서는 구조들이 객체들 사이에 참조가 직접 저장됨으로 좀더 빨리 복잡한 구조를 조회할 수 있다. 객체지향 데이터 베이스는 테이블을 통해 검색하므로, 관계형 데이터 베이스에서처럼 관계를 재구성할 필요가 없다. Taylor [11]는 객체지향방식으로 부품구성표(bill of materials)를 표현하는 방법이 더 효율적임을 강조하였다. 관계형 데이터 베이스는 구성 요소의 리스트를 나타내는 일련의 엔트리를 테이블 형태로 나타낸다. 이들 구성 요소 각각은 차례로 다른 엔트리에 의해 정의된다. 이러한 부구성 요소는 생산품에서 대개의 기본적인 부분을 나타낼 때까지 하위 레벨 구성 요소들로 연결된다. 복잡한 생산품은 평균 6개 정도의 구성 레벨이 필요하다. 즉, 각각 다섯 개의 구성 요소이라면 모두 3000개 이상의 링크가 생기는 것이다. 이들 링크 각각은 양쪽 엔트리 모두에 특수한 값(이 경우에는 부품 번호)을 가짐으로써 형성된다. 관계형 데이터 베이스로부터 BOM을 검색하는 것은 시간

낭비다. 왜냐하면 시스템은 일일이 각 구성 요소를 다음 것과 링크하면서 검색해야 하기 때문이다. 데이터 베이스 관리자는 이러한 부품 번호가 들어 있는 적당한 테이블 각각을 검색한다. 이러한 검색을 하려면 관계형 데이터 베이스를 최적화해야 한다. 그러나 하나의 구조에서 수천 번의 검색이 필요하다면 이것 역시 시간 낭비이다. 객체 지향 데이터 베이스는 제품을 복합 객체로서 저장하며, 각 구성 요소에 대한 링크로 직접 참조를 나타낸다. 이렇게 표현하면 데이터 베이스 관리자가 직접 구성 요소 객체를 참조할 수 있으므로, 테이블을 검색하여 찾는 노력을 최소화할 수 있다. 관계형 데이터 베이스 언어에서는 필요한 조인은 미리 계산된다. 사실, 데이터 베이스에 하나의 하나의 질문을 가지고 전체 BOM을 검색할 수도 있다. 객체지향 데이터 베이스에서는 관계형 데이터 베이스보다 훨씬 시간이 덜 걸리면서도 완벽하게 BOM을 검색한다.

객체지향 개념의 특성을 이용하여 Chung and Fischer [2]는 BOM의 구조(BOM structure)와 부품의 특성(part property)에 대해 객체지향 방식으로 모델링을 시도하였다. 이 데이터 모델은 BOM class와 propoerty class로 구분되며, 다시 BOM은 component, product, aggregation object class로 세분된다. property class는 shape와 specification class로 다시 세분된다. BOM과 property 사이의 관계는 referencing, referenced_by 관계로 정의되고, component, product, aggregation object간의 관계는 composed_of, part_of로 정의된다. 또한, BOM object와 다른 object 간의 관계는 own, owned_by로 정의된다. 이러한 데이터 구조와 관계를 객체지향 개념으로 모델링을 하였고 C++를 이용하여 시스템을 구현하였다.

Trappey외 2인[13]은 Smalltalk를 이용하여 객체지향 방식으로 BOM 구조와, 데이터를 처리하는 transaction 모델들을 설계하였다. 이 구조에서는 부품 class를 제조부품(manufactured parts)과 조달부품(procured parts) class로 구분하였다. 또한, 부품가공을 위한 routing class, 부품 조달업체를 나타내는 업체 class, 포장방법을 나타내는 packaging class를 고려하였다. BOM 구조를 Model이라는 class 의 sub-class로 표현하였으며 transaction 모델로서 BOM출력, BOM트리, 부품라이브러리, 부품윈도우, 제품브라우저를 설계하였다. 여기서는 BOM의 구조에 대해서는 언급이 없으며 단지 transaction과의 관계를 주로 설명하고 있다.

Nandakumar [9]는 관계형 DB를 이용하여 BOM processor의 효율성을 분석하였다. 이 연구에서는 BOM processor의 성능을 비교하는 연구로서 SQL 언어로 프로그램된 processor와 FORTRAN과 같은 일반적인 언어로 프로그램된 processor의 기능을 비교하였다. 여기서는 dBase IV의 SQL언어를 사용하였다. 비교한 결과 SQL언어로 프로그램된 processor가 더 유연하고, 시간이 적게 소모되며, 다른 컴퓨터 시스템에서도 쉽게 이용할 수 있었다.

Katz[4]는 데이터 구조가 복잡한 엔지니어링 분야에서 여러 가지로 제시된 version에 대한 개념을 통합 정리하였다. 여기서는, version, component hierarchy, version history, equivalency 등의 모델링 프리미티브들에 대해 정의를 정리하였다. 그리고 설계 관리를 위한 기능으로 currency mechanism, dynamic configurations, workspace, change/constraint propagation, inheritance 등에 대한 개념을 설명하고 있다.

STEP의 표준으로 ISO 10303중 Part 44 에 대한 표준이 제정되었다 [3]. 이것은 제품구조 및 구성에 관리에 관한 개념 및 기준으로서 세부 내용은 다음과 같다.

- product structure schema
- product concept schema
- configuration management schema

product structure schema 는 BOM과 같은 제품의 구성관계를 제품설계 관점에서 표현하는 방법을 정의한 것이다. configuration management schema 는 특정 제품에 대한 제조나 조립 계획에 대한 사양 정보를 관리하는 방법을 정의한 것으로서 effectivity 개념과 부품 변경 개념이 적용된다. product concept schema는 제품개념 정의 하는 것으로서 제품개념은 제품에 대한 소비자들의 요구사항에 따라 나타난 제품의 사양을 의미한다.

국내에서 연구된 사례를 보면, 주로 관계형 DB를 이용하거나 근거로 하여 연구되었다. 그러나, BOM의 구조가 가장 간단한 형태로 되어 있어 현장에서 필요로 하는 변경이나 부품 대체, 옵션관리등은 전혀 고려하지를 못하고 있다.

강신영 [15]은 BOM의 기능적인 측면에서 효과를 분석하였다. 이 연구에서는 기존의 구조형 BOM과 modular BOM을 데이터베이스 관리면에서 분석함으로써, 모듈화한 BOM이 데이터 관리면에서 지니는 효과성을 분석하였다. 여기서는 1) BOM 데이터베이스의 구성 레코드 수, 2) 제품구성 변경에 따른 갱신의 용이성을 비교 분석하였다. Modular BOM이 기존의 구조형 BOM에 비해 레코드 수가 감소하며 변경도 용이한 것으로 분석되었다. 그러나 레코드의 변경이 구조형 BOM에 비해 복잡한 단계를 거쳐야 하는 단점이 있어 주의가 요망된다.

이경우[25]는 join기능을 이용하여 BOM을 생성하는 방법과 효용성을 분석하였다. BOM 전개중 다단계 정전개를, 단계별로 join 기능을 반복적으로 이용하여 수행할 수 있는 방법을 제시하였다. 처음에는 품목마스타와 BOM테이블간의 join으로 1단계 자품목 목록을 생성한다. 그리고는 1단계 자품목 목록과 BOM테이블간의 join으로 2단계 자품목 목록을 생성하고, 다시 2단계 자품목 목록과 BOM테이블간의 join으로 3단계 자품목 목록을 생성 한다. 이러한 방식이 반복적으로 수행되어 다단계 정전개가 수행된다. 이 방법은 기존의 BOM테이블을 탐색하여 전개하는 방식보다 시간과 메모리를 절약할 수 있는 장점이 있다. 여기서는 제시한 사례에서는 검색응답시간이 1/5로 줄었고, MRP계산시간이 1/3으로 줄었다. 또한 주기억장치 점유공간이 다소 줄었고 보조기억장치의 공간이 약간 늘었다.

또한, 관계형 DB에서 모델, 기능, 소비자의 선택사양들을 모두 고려하는 BOM을 제시한 경우도 있다. 이한표 [26]는 modular BOM 인 family BOM의 구현과 연관하여 사용 목적별 분류를 BOM 구성 모듈과 분리하여 간접적으로 표현하는 방안을 제시하였다. 즉, 시장에 판매되는 최종제품인 buyer model에 관한 자료만을 운영하고, 관리를 위한 basic model, function model 등의 가상적인 자료는 실체적인 대상인 buyer model을 통하여 간접적으로 표현되도록 하였다. 여기서 제품을 공통부품(family common)과 선택사양(option)으로 구분하였다. BOM구성시 공통부품은 자동적으로 포함되며 선택사양을 모델별로 선택할 수 있도록 하였다.

객체지향 방식으로 BOM과 기술자료간의 통합을 시도한 연구도 있다. 곽서경[16]은 C++ 언어를 이용하여 이러한 것을 시도하였다. 기존의 BOM과 기술자료 관리체계에서는, 서로 데이터가 연계되지 않고, 어느 한쪽에서 설계변경이나 version이 바뀔 경우 다른 한쪽에서 관계되는 자료가 변경되지 않아 내용이 일치되지 않는 단점이 있다. 이러한 문제점을 해결하기 위하여, 객체지향 방식으로 제품 class에 BOM과 기술자료를 포함하도록 설계하고 BOM 구조와 도면, 사양 등의 기술자료 class를 정의하여 연결하는 방식을 이용하였다. 이 연구에서는 BOM과 기술자료를 통합하고자 한 점에 의의가 있으나 BOM 구조나 프로세서 기능 등에 대해서는 실제 운용할 수 있는 수준으로 고려되지 않은 단점이 있다.

2.2 객체지향 DB의 활용추세

최근 CALS의 확대 적용과 더불어 객체지향 DB도 점차 시장이 확대되고 있다. 앞으로 97년 부터는 데이터베이스 부분에 ODBMS가 표준으로 채택될 예정이다[14]. 또한, 멀티미디어 환경으로 진전됨에 따라 관계형 DB는 다음과 같은 경우 한계점을 드러내고 있다[7,21]: 객체지향 DB의 장점은 다음과 같다.

- Object-oriented paradigm을 통한 각 기능의 Module화
- CAD에서의 도면 생성 또는 수정 작업과 같이 오랜 시간 동안 데이터를 처리(long transactions)를 하는 경우 내포성 트랜잭션(nested transaction)이나 약성잠금(soft lock)을 이용하여 효율적으로 관리할 수 있다.
- 설계 변경 등으로 인하여 도면을 수정하는 경우가 빈번히 발생한다. 또한 도면이 진행, 승인, 폐기 단계를 거치기 때문에 구 version의 도면을 보관해야 설계의 노하우를 유지하게 된다. ODBMS는 transient, working, released version 기능을 통하여 이러한 version 관리를 손쉽게 처리한다.
- ODBMS는 CAD, 영상, 음성, text, DB 등과 같이 내부 구조가 단순하며 데이터의 양이 많은 unstructured object (raster bit maps, audio, 긴 text file 등의 multimedia 데이터)를 처리하기 쉽다.
- Engineering BOM은 도면에 포함된 부품을 부품 관련 정보로 마스터로부터 읽어 들여 BOM을 자동 생성하고, MRP시스템의 마스터 데이터에 입력시켜 주는 시스템이다. 이 BOM은 assembly-part hierarchy 구

조를 가지고 있어 composite object hierarchy 관계를 생성한다. 또한, CAD의 형상도 point, line, surface로 구성되어 있어 복잡한 데이터 구조를 가진다. 이러한 특성들은 객체지향형 DB에 적용하기 적합한 것들이다.

그러나 객체지향형 DB가 데이터베이스 시장에 아직 영향을 미치지 못하는 이유 [21]는 첫째, 객체지향형 DB는 데이터베이스 시스템으로서 완전한 기능을 가지고 있지 못하다. 둘째, SQL을 포괄하는 수퍼셋 조회언어를 지원하지 않는 등 관계형 DB와 호환되지 않는 점이다. 이런 단점을 보완하기 위해서 관계형 DB에 객체지향 DB를 통합하는 아키텍처를 이용하고 있다. 이 방법에는 게이트웨이 방식, 관계형 DB 엔진에 객체지향 레이어를 사용하는 방식, 통합형 객체-관계형 접근방식이 있다. 이중에서 UniSQL은 통합형 객체-관계형 접근방식을 채택하고 있다.

3. 객체지향 BOM 구조설계

3.1 부품 클래스의 설계

개발하는 시스템에서는 부품의 구매에 관련된 업체정보와 부품형상에 관련된 형상/도면 정보는 제외되어 있다. 그러므로, 여기서는 객체지향 BOM시스템을 구성하는데 가장 기본이 되는 부품 클래스를 설계한다. 사용되는 Rumbaugh의 다이어그램 표기법은 그림 1과 같다. 부품 클래스에 대한 클래스 관계성 다이어그램 (class-relationship diagram)은 그림 2와 같다. 이 그림에서는 부품형상 클래스와 부품업체 클래스는 참고로 표현되어 있다.

부품 클래스에서, 부품에 대한 기본적인 정보로서 부품코드, 부품명, 부품 규격, 부품표준원가 등이 있고 이 정보들을 관리하는 함수(member function)들이 있다. 그리고, 상세한 부품 정보를 참조하기 위한 부품속성 클래스의 참조 함수들, 부품의 형상정보를 참조하기 위한 형상정보 클래스의 참조 함수들, 부품의 구매와 관련된 정보를 참조하기 위한 부품업체 클래스의 참조함수들을 포함하고 있다.

부품 클래스로부터 파생되어지는 하위클래스를 살펴보면, 모델의 정보를 가지고 있는 모델 클래스, BOM의 구조정보를 갖는 BOM구성단위 클래스, 옵션부품에 대한 정보를 가지고 있는 옵션부품 클래스 등이 있다. 이러한 클래스들은 상위클래스의 속성을 상속받기 때문에 자신의 속성뿐만 아니라 부품 클래스의 속성도 가지고 있다.

부품 클래스의 내부를 자세히 살펴보면, 우선 기본정보인 부품 코드와 부품명, 부품규격, Lead Time, 부품원가 등의 속성이 있다. 그리고 속성을 생성하고 조회, 갱신,

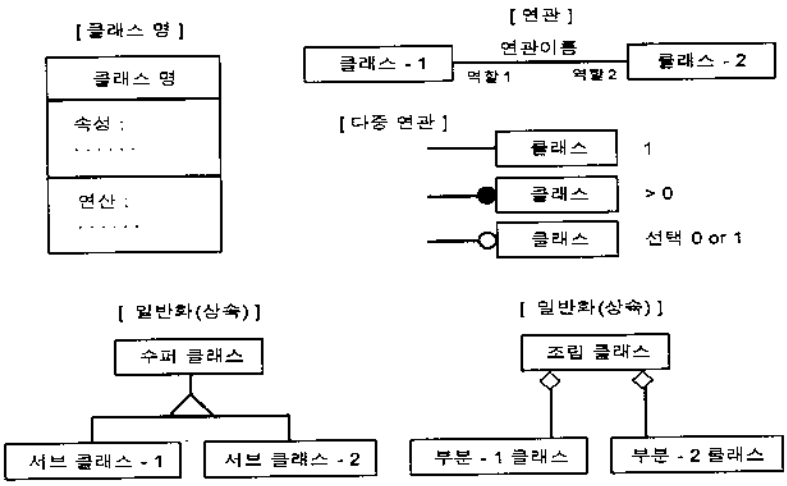

그림 1. Rumbaugh의 다이어그램 표기법 [10, 24]

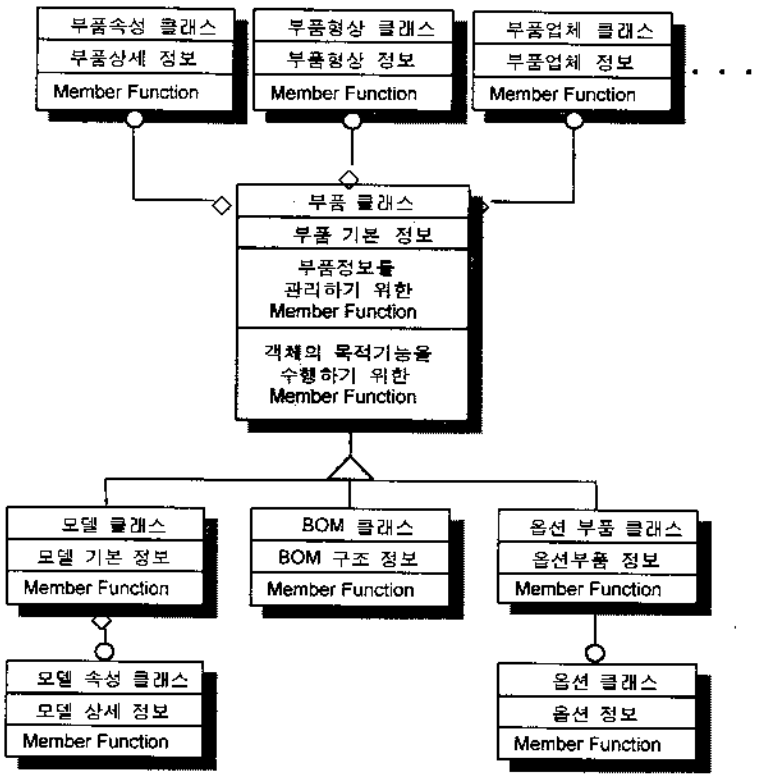

그림 2. 부품 클래스 구조

삭제하는 함수들이 있다. 이런 기본적인 속성들은 private 속성으로 정의되어 직접적으로 상속되는 객체들을 제외하고는 직접적으로 조작할 수 없게 구성된다. 그 외에 부품 속성객체, 구매업체 객체, 부품형상 객체 등을 참조하기 위한 member function들이 포함된다.

부품 클래스에서 상속되어 생성되는 모델 클래스, BOM 구성단위 클래스, 옵션부품 클래스에서는 부품 클래스의 속성과 함수들을 재정의할 필요없이 자유롭게 사용할 수 있다. 이와 같은 방법을 사용함으로써 부품의 속성이나 함수에 어떠한 변화가 생겼을 경우, 부품 객체만 수정하면 상속된 객체에서는 어떠한 추가작업 없이도 자동으로 수정된 내용을 사용할 수 있게 된다. 또한 부품이 새로운 기능을 가진 객체를 참조해야 할 경우에도 부품 객체에 새로운 정보를 참조할 수 있는 환경만을 생성해 주면 상속받은 객체에서도 사용할 수 있게 된다.

3.2 BOM 클래스의 설계

객체지향 BOM 시스템에서는 BOM의 구조를 구성하는 하나의 단위를 객체로 생성하게 되는데 이들의 집합을 BOM 구성단위 클래스 (BOM item class)라고 정의한다. BOM클래스는 BOM 구성단위 객체의 리스트로 구성되는데 그림 3은 이러한 개념을 나타낸다. BOM 구성단위 객체는 조립되는 상위부품의 리스트, 조립에 사용되는 하위부품들의 리스트, BOM 구성단위 속성정보 (상위의 부품 클래스로부터 상속받은 부품속성 정보), 그리고 수량등과 같은 하위부품의 속성 정보를 가지고 있다.

일반적으로 BOM의 구조는 그림 4와 같은 트리 형식으로 표현되는데 기존의 관계형 DB 구축방법에서는 그림 5와 같이 하나의 테이블에 상위부품코드, 하위부품코드, 하위부품수량, LLC(the lowest level code), 기타 하위부품에 대한 속성정보 등의 칼럼으로 표현하고 있다. 그러나 이 구조를 객체지향 기법을 이용하여 BOM 구성단

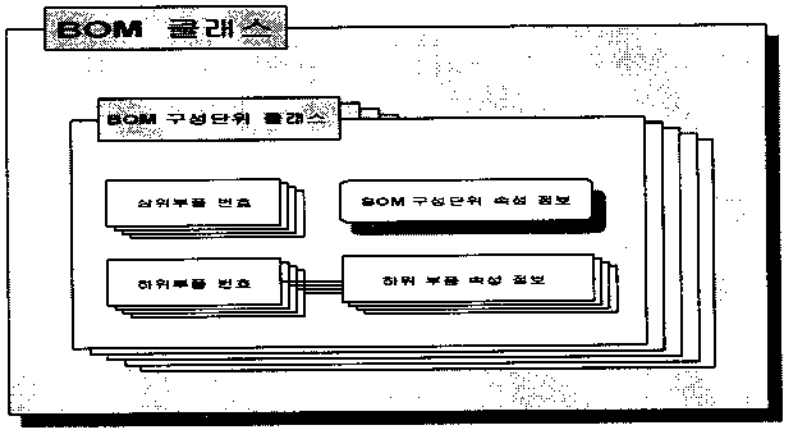

그림 3. BOM 클래스의 BOM 구성단위 객체

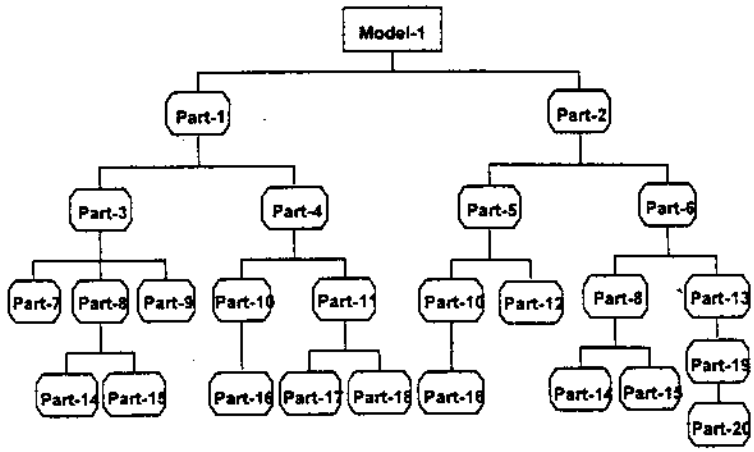

그림 4. BOM TREE 예제

BOM Table			
상위 부품 코드	하위 부품 코드	상위 부품 코드	하위 부품 코드
X	Model	Part 8	Part 14
Model	Part 1	Part 8	Part 15
Model	Part 2	Part 10	Part 16
Part 1	Part 3	Part 11	Part 17
Part 1	Part 4	Part 11	Part 18
Part 2	Part 5	Part 13	Part 19
Part 2	Part 6	Part 19	Part 20
Part 3	Part 7	Part 7	X
Part 3	Part 8	Part 14	X
Part 3	Part 9	Part 15	X
Part 4	Part 10	Part 9	X
Part 4	Part 11	Part 16	X
Part 5	Part 10	Part 17	X
Part 5	Part 12	Part 18	X
Part 6	Part 8	Part 12	X
Part 6	Part 13	Part 20	X

그림 5. 관계형 데이터베이스의 테이블로 표현한 BOM TREE

위 객체로 표현하면 그림 6과 같이 나타낼 수 있다. 이 그림의 BOM 구성단위 객체내에서 좌측 부품코드가 BOM 구성단위 객체의 이름을 나타내고 있으며, 우측 상단에 있는 리스트가 상위부품의 리스트를 나타내고 우측 하단에 있는 리스타가 하위부품의 리스트이다. 예를 들면, BOM 구성단위 객체-8의 이름은 "Part-8"이며, 상위부품이 "Part3"과 "Part16"이고, 하위부품이 "Part14"와 "Part15"로 구성되어 있다. "X"는 상위부품이나 하위부품이 없다는 의미이다.

이 구조에서는 상하위 부품들을 직접 연결하여 주기 때문에 BOM 구조가 복잡하면 복잡할수록, 공용부품의 수가 많을수록 BOM 전개나 검색이 관계형 구조에 비하여

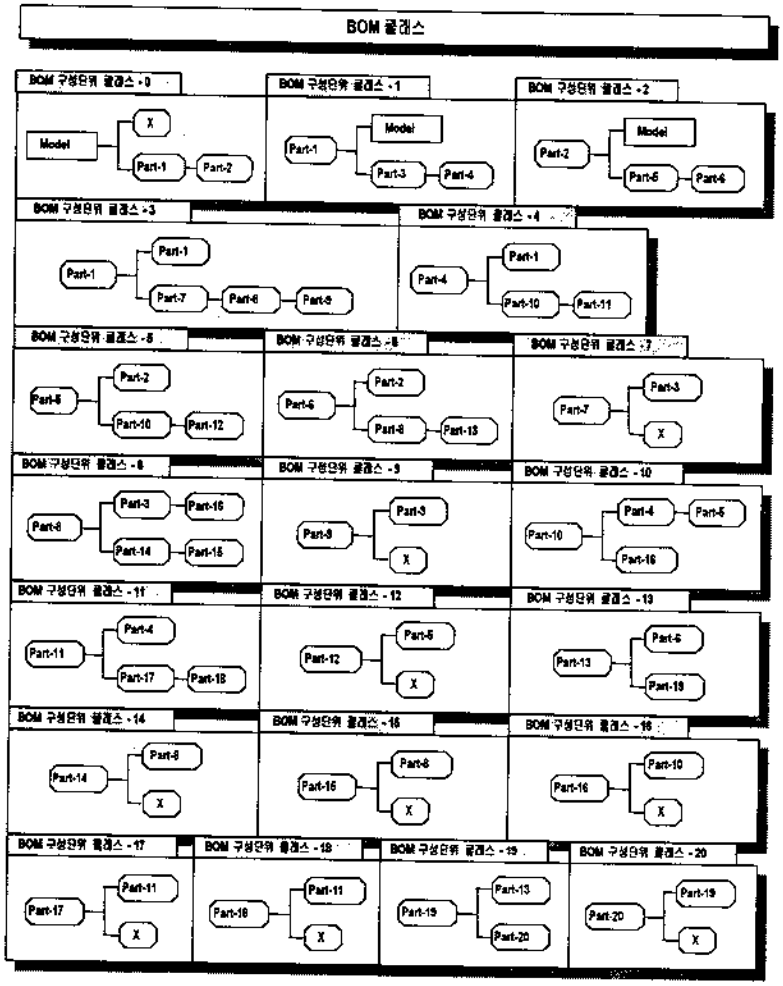

그림 6. 객체 지향 기법으로 표현한 BOM TREE

빠르다는 것을 쉽게 예상할 수 있다.

3.3 옵션 클래스의 설계

생산되는 제품의 한 모델이 여러개의 옵션(option)을 가지고 있는 경우에 각 옵션마다 BOM을 새로 구성하여 관리하는 것은 데이터의 양이 많아지고 부품 소요량을 계산하는데도 매우 번거로워진다. 이러한 경우 표준이 되는 BOM을 설정한 후 옵션에 따라 바뀌는 부품들을 객체로 생성해서 별도로 관리하고, 옵션 선택시에 BOM상의 부품을 교체하여 관리하는 것이 효율적이다.

여기서는 모델별 옵션을 고려하기 위해 그림 7과 같이 1) 모델 클래스, 2) 옵션 클래스, 3) 옵션부품 클래스가 고려된다. 우선, 모델 클래스는 모델코드, 모델명, 모델별로 옵션 종류를 나타내는 옵션 객체 리스트로 구성되어 있다. 옵션클래스는 옵션코드, 옵션명, 옵션에 대한 상세설명, 각 옵션에 따라 교체대상이 되는 옵션부품 리스트로 구성되어 있다. 옵션부품 클래스는 부품 객체로부터 상속받아 생성되며, 교체대상인 부품 코드, 대체부품인 옵션교체부품 코드, 옵션교체부품 수량 등의 정보로 구성되어 있다. 옵션부품 객체는 옵션 클래스에 리스트로 관리되며, BOM상의 구조를 검색하여 옵션에 따라 부품을 교체시키는 기능을 포함하고 있다. 옵션부품 객체는 우선

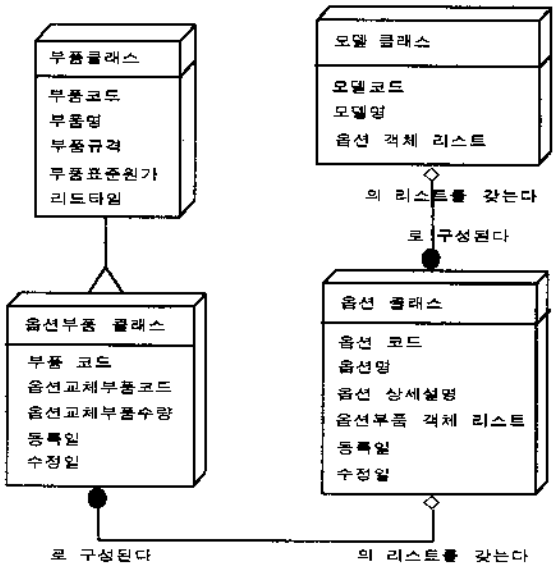

그림 7. 옵션관련 클래스 구성도

옵션이 적용될 모델을 선택하고, 다음으로 모델 클래스에 등록된 옵션별 교체대상 옵션부품을 선택한 후, 교체부품인 옵션교체부품과 옵션교체부품 수량을 입력하여 생성된다.

예를 들어, 그림 4의 한 모델이 그림 8과 그림 9의 두 가지 옵션을 가지고 있다고 하자. 그림 8에서는 교체대상인 옵션부품 "Part-9", "Part-10", "Part-13", "Part-16"은 옵션교체부품 "P-9-1", "P-10-1", "P-13-1", "P-16-1"으로 각각 교체된다는 의미이며 검게 표시되어 있다. 그림 9에서는 교체대상인 옵션부품 "Part-5", "Part-8"은 옵션교체부품 "P-5-1", "P-8-1"로 각각 교체된다. 옵션부품이 교체될 경우 그 하위부품들은 자동적으로 교체된다.

옵션교체부품에 대한 BOM구성단위 객체는 그림 10과 같이 BOM클래스의 하부 클래스인 BOM구성단위 클래

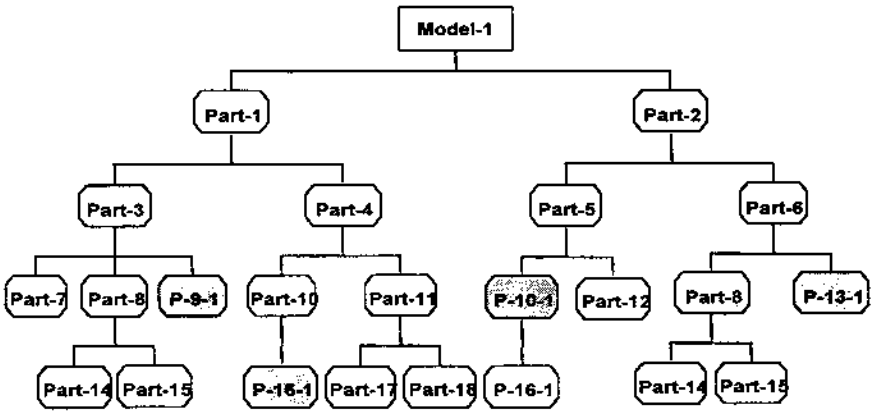

그림 8. 옵션예제 - 1

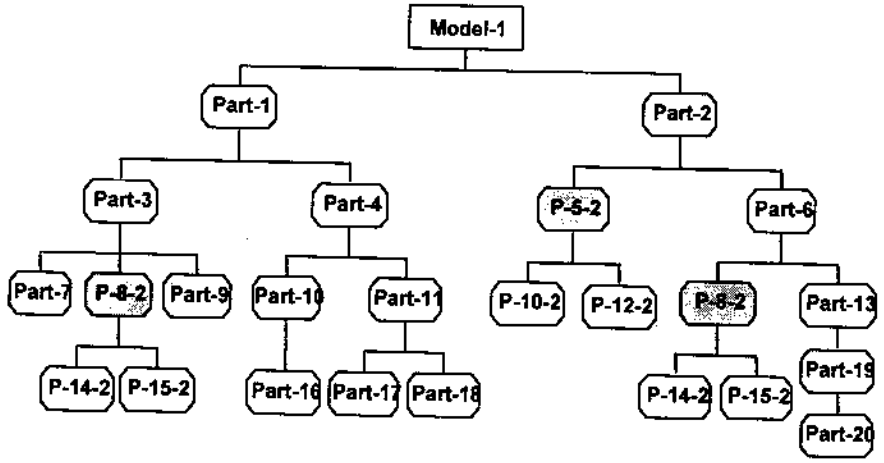

그림 9. 옵션예제 - 2

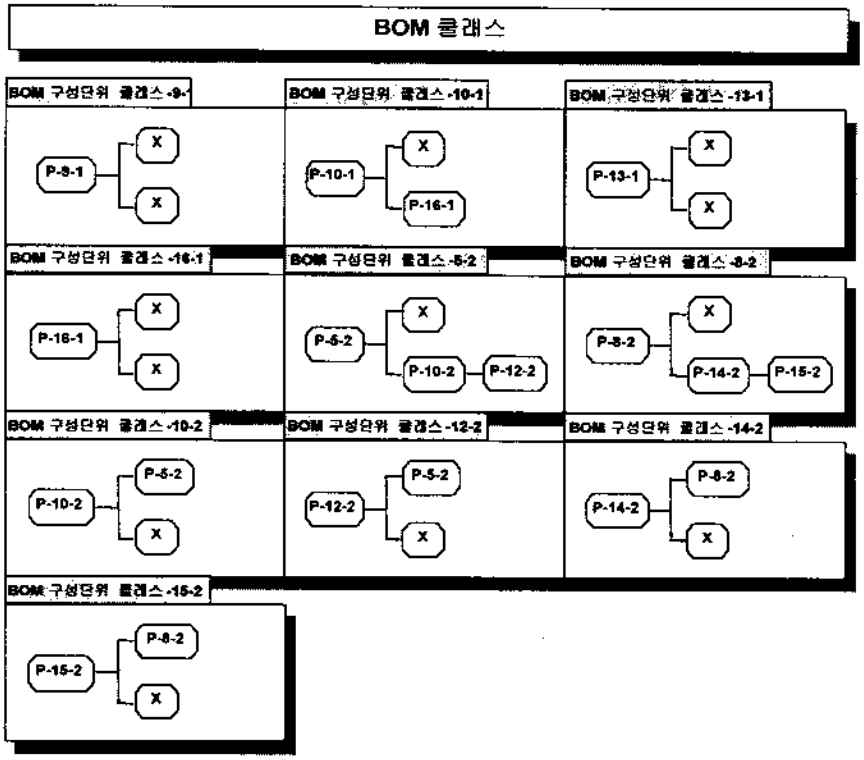

그림 10. 옵션교체부품의 BOM구성단위 객체

스의 구조에 맞게 생성된다. 여기서 옵션교체부품의 상위 부품은 정해지지 않았기 때문에 "X"로 표시되어 있다. 이

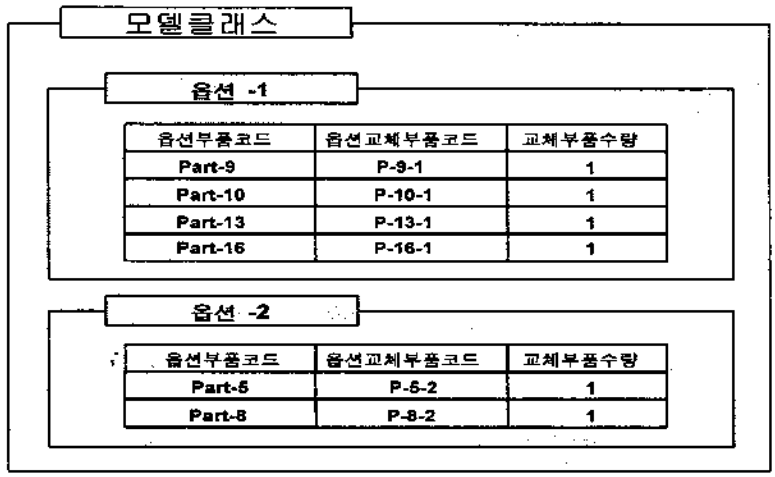

그림 11. 모델객체에 정의된 옵션 및 옵션교체부품 리스트

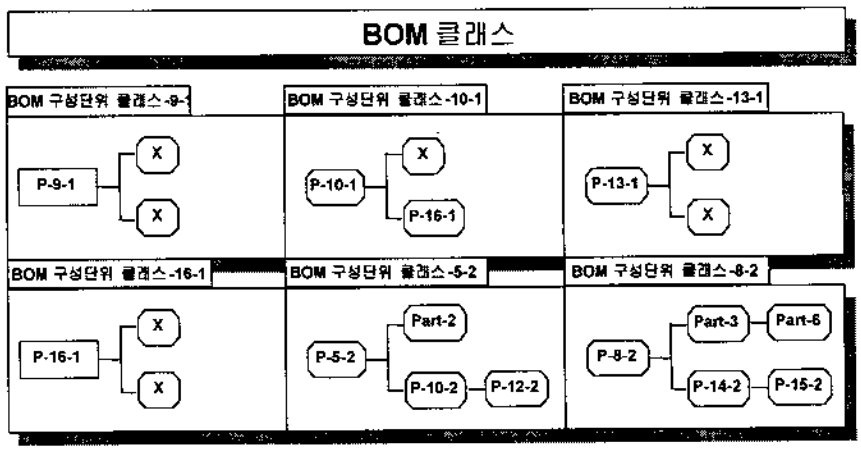

그림 12. 옵션-1을 선택한 경우의 옵션교체부품의 BOM구성단위 객체의 변화

것은 옵션교체시 상위부품 코드로 변환된다.

해당 교체부품의 입력을 마친후 사용자는 원하는 모델에 옵션을 입력하여 옵션객체의 리스트를 생성하게 된다. 하나의 모델에는 다수의 옵션이 존재할 수 있으며 각 옵션에는 교체될 부품의 리스트가 등록된다. 예를 들면 그림 11에 나타난 바와 같이 옵션-1에서는 "part-9", "part-10", "part-13", "part-16",이 "p-9-1", "p-10-1", "p-13-1", "p-16-1"로 각각 교체된다.

사용자가 BOM 전개시 주어진 옵션 리스트 중에서 하나를 선택하게 되면 해당하는 교체부품들을 검색하여 BOM 클래스에 존재하는 BOM 구성단위 객체들의 상위부품 리스트와 하위부품 리스트를 수정하여 옵션에 따른 부품의 교체를 실행하게 된다.

옵션으로 부품 교체가 이루어진 후의 BOM구성단위 객체는 그림 12와 그림 13과 같이 옵션교체부품의 상위부품이 변경된다. 예를 들면, 교체된 "p-9-1"의 상위부품이 "X"에서 "part-3"로 바뀌게 된다. 이 변화는 옵션 선택시 임시적으로 이루어지게 되고 옵션 기능을 마치게 되면 원래의 상태로 복구된다.

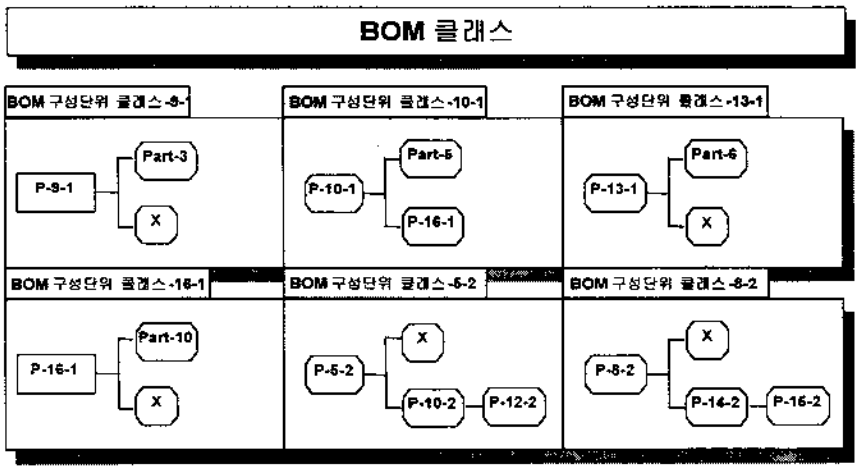

그림 13. 옵션-2를 선택할 경우의 옵션교체부품의 BOM구성단위 객체의 변화

3.4 모델 클래스 설계

모델은 다량으로 생산되는 하나의 제품을 말하며, 모델 클래스는 이러한 모델의 기본 정보를 관리하며, 모델의 생성과 삭제, 갱신의 기능을 가지고 있다. 모델 클래스는 모델의 속성정보 클래스와 모델의 BOM 클래스, 모델의 BOM이 포함하는 옵션들의 리스트를 포함하며 하나의 모델에 대한 모든 정보를 생성하고 처리하는 역할을 한다.

그림 14는 모델 클래스의 구조를 그림으로 표현한 것

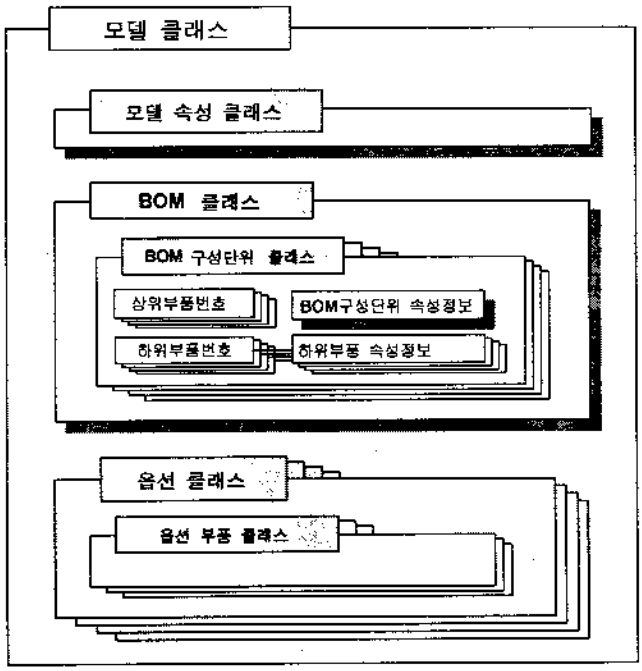

그림 14. 모델 클래스의 구조

으로 각 모델은 하나의 BOM 객체를 포함할 수 있으며, 모델에 존재하는 옵션들의 객체 리스트를 포함하고 있다. BOM 객체는 앞에서 설명한 것과 같이 구성단위 객체의 리스트로 이루어져 있고 각 옵션 객체들은 옵션부품 객체의 리스트로 구성되어 있다.

이상과 같은 구조에서 볼 때 각 모델 객체는 해당 모델이 가질 수 있는 모든 정보에 접근이 가능하게 되며 필요시 포함될 객체들을 추가시켜 보다 편리하게 모델 정보를 확장시킬 수 있다.

3.5 객체지향 BOM의 전체 구조

먼저 기존의 관계형 BOM시스템의 구조를 보면 그림 15와 같이 일반적으로 부품 테이블, BOM 구조 테이블, 모델정보 테이블, 부품 이력 테이블, 부품업체 정보 테이블, 대치품 정보 테이블, 옵션 정보 테이블 등을 가진 데이터베이스와 부품에 대한 관리기능, BOM 생성/관리 기능, 모델에 대한 관리 기능, 옵션에 대한 관리기능, 대체부품에 대한 관리기능, 이력정보에 대한 관리 기능 등으로 이루어져 있다. 이와 같은 기존의 시스템의 각 관리 기능들은 독자적인 영역을 가지며 다른 영역에 대해 내부적으로 어떠한 직접적이고 자동적인 영향력을 가질수 없다.

객체지향적 BOM 시스템에서는 위에 표현된 각 기능들이 객체로 생성되는데 이는 부품객체, BOM구성단위 객체, 부품업체 객체, 모델 객체, 옵션 객체 등으로 이루어 진다. 각 객체에는 관리되는 데이터들이 포함되고, 객체마다 데이터를 관리하는 member function과 객체의 고유한 기능을 수행하는 member function들이 포함된다.

부품객체에는 부품의 생성, 삭제, 갱신에 관련된 함수가 포함되고 또 부품의 속성을 관리하는 객체를 참조하기 위한 함수들도 포함된다. BOM 구조 객체에는 BOM의 기본 구조를 구성하고 저장하는 함수들과, BOM구조 생성에 필요한 정보를 가진 객체를 참조하는 함수들이 있다. 모델 객체에는 모델의 생성과 삭제를 관리하는 함수들과 모델이 가지고 있는 BOM 구조를 관리하는 함수들로 구성된다.

이와같이 각 객체가 생성되면 각 객체사이의 관계가 설정되며 각 객체간의 상속성과 참조성이 부여된다. 이러한 특성을 고려하여 지금까지 설계된 BOM의 구조들을 통합하면 그림 16과 같다. 그리고 도출된 클래스들이 표 1에 정리되어 있다.

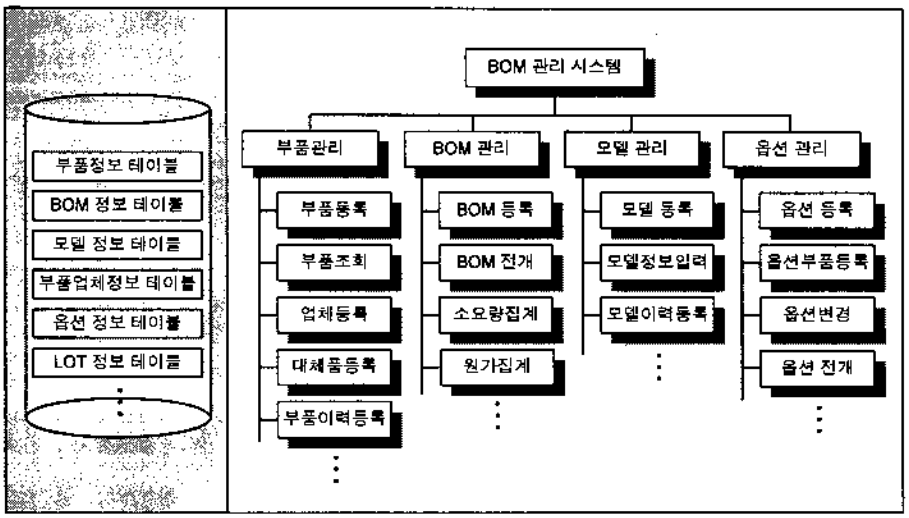

그림 15. 일반적인 BOM 관리 시스템의 구성

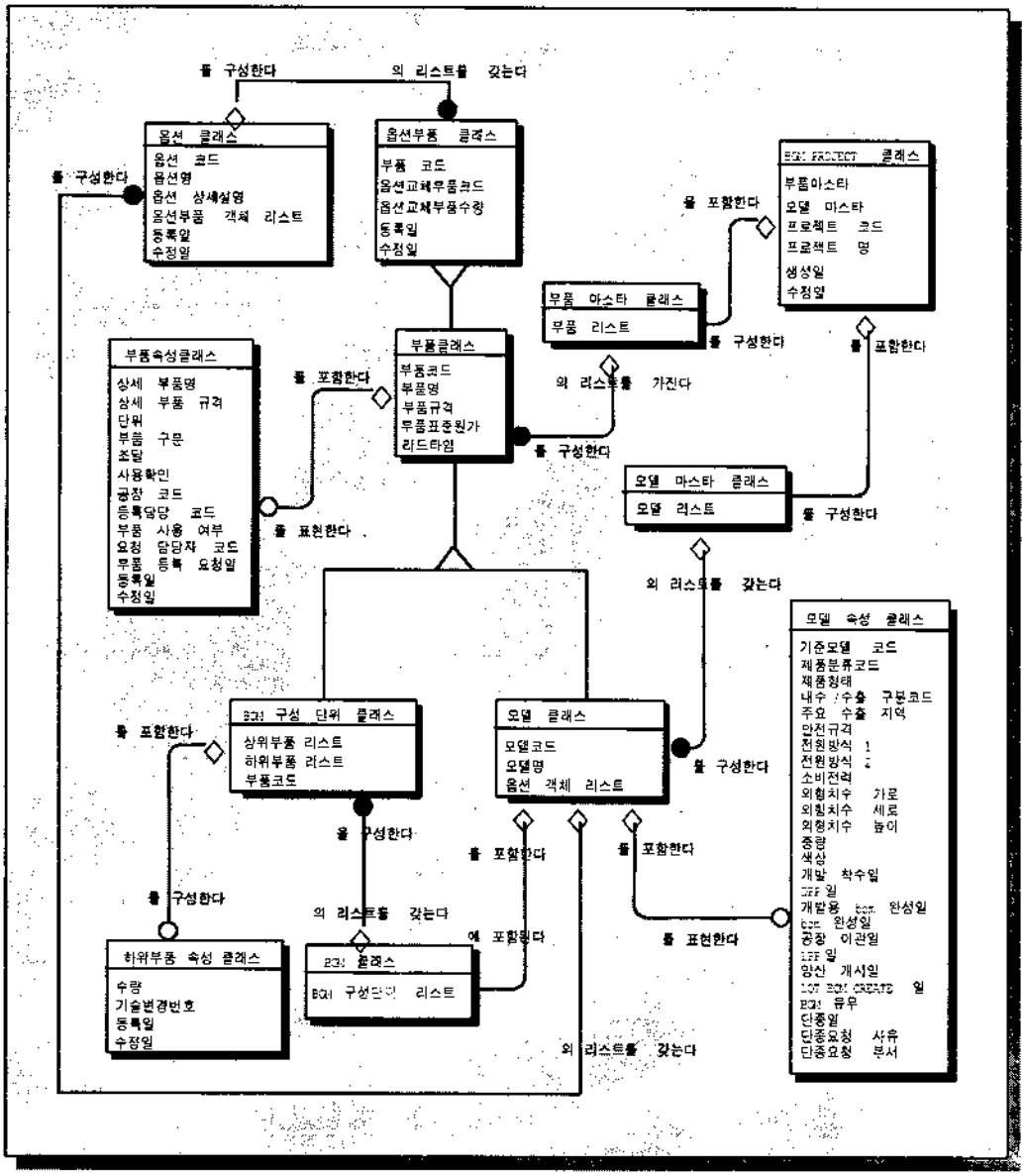

그림 16. 객체지향 BOM 관리 시스템 전체구조

표 1. 도출된 클래스

클래스명	상 세 설 명
BOM Project 클래스	프로젝트란 객체지향 BOM 관리 시스템의 관리 단위로서 유사한 생산 유형을 갖는 제품군을 나타낸다. 이 클래스는 모델, 부품, 옵션부품의 세 개의 마스터 클래스를 포함하며 마스터 클래스의 생성과 파괴, 변경을 제어하는 함수를 가진다. 일반적으로 하나의 프로젝트에는 여러개의 모델이 존재하며 각 모델들은 자신의 BOM 구조 정보를 가지고 있다. 부품은 프로젝트에 전반에 걸쳐 하나의 마스타로 존재하며 각 모델들은 이 부품 마스타를 기초로 BOM을 구성하고 옵션부품을 설정하게 된다.
모델 마스터 클래스	모델은 다량으로 생산되어지는 하나의 제품을 말하며 모델 클래스는 이러한 모델의 정보를 관리하는 기능을 가지고 있다. 모델마스터 클래스는 이러한 모델 클래스의 리스트를 가지고 있으며 이러한 리스트를 유지하고 관리하는 기능을 가지고 있다.
부품 마스터 클래스	부품 마스터 클래스는 BOM 관리 시스템의 기본이 되는 부품 클래스의 리스트를 가지고 있으며, 이 리스트를 유지 관리하는 기능을 가지고 있다. 부품 마스터 클래스는 하나의 프로젝트에 대하여 하나만이 존재하며 프로젝트 전반에 걸쳐 사용되어진다. 프로젝트에 존재하는 여러 모델의 BOM과 옵션부품들은 이 부품 마스터에 존재하는 부품을 기반으로 생성되며 부품마스터의 부품이 변경이나 삭제될 경우 관련되는 BOM과 옵션부품들도 수정을 해주어야 한다.
부품 클래스	부품 클래스는 BOM 관리 프로그램에 기본이 되는 클래스로서 부품에 대한 속성 클래스를 관리하는 기능과 부품의 등록 삭제, 변경에 관한 기능을 수행한다. 부품 클래스로부터 옵션부품 클래스와 BOM 구성단위 클래스, 모델 구성단위 클래스가 상속되며 이들 클래스에서는 부품 클래스의 기능과 정보를 사용할수 있게 된다.
부품속성 클래스	부품의 상세한 속성정보를 관리하는 클래스로서 시스템의 필요에 따라 다양한 데이터를 정의하여 사용할 수 있다. 본 논문에서는 부품의 일반적인 데이터만을 정의해 사용하였으나 사용목적에 따라 부품의 형상정보나 도면정보, 생산업체정보, 관리 담당자 정보 등을 클래스로 정의하여 부품 클래스에 연결시켜 사용할 수 있다.
모델 클래스	모델은 다량으로 생산되어지는 하나의 제품을 말하며, 모델 클래스는 이러한 모델의 기본 정보를 관리하며, 모델의 생성과 삭제, 갱신의 기능을 가지고 있다. 모델 구성단위 클래스는 모델의 속성정보 클래스와 모델의 BOM 클래스를 포함하여 하나의 모델에 대한 모든 정보를 생성하고 처리하는 역할을 한다.
모델 속성 클래스	모델에 대한 상세한 속성정보를 관리하는 클래스이다. 모델에 대한 속성은 구성목적에 따라 약간의 차이가 있으나 일반적으로 모델에 대한 일반적 규격/형상정보, 제품의 일반적 사용옵션/사양 정보, BOM 구성에 관한 관리 정보 등을 포함하고 있다.
BOM 클래스	BOM 클래스는 BOM의 부품구조에 관한 정보를 생성하고 관리하는 기능으로 구성되어 있다. 이 클래스는 부품의 구조를 저장하기 위해 BOM 구성단위 클래스의 리스트를 가지고 있으며 이 리스트를 생성하고 관리하는 함수를 가지고 있다. 또한 BOM 구성단위 리스트를 사용하여 BOM의 전개를 하기위한 기능을 가진 함수들을 가지고 있다.
BOM 구성단위 클래스	BOM 구성단위 클래스는 부품구조상에 존재하는 하나의 부품정보를 기반으로 그 부품에 연결된 상위 부품과 하위부품들의 정보를 가지고 있다. 모델 또한 부품과 마찬가지로 BOM 구성단위로서 생성하여 관리한다. 이 클래스는 BOM상에 존재하는 중복부품을 제외한 부품의 수만큼 생성하게 되며 이 단위개체를 이용함으로서 BOM의 전개나 소요량 집계시에 부품 구조상의 임의의 위치로 부터 전개나 집계를 할수 있다.
하위부품 속성 클래스	BOM 구성단위의 하위 부품을 등록할 때 생성하여 하위 부품과 종속되어 BOM 구성단위 클래스로 등록되는 클래스이다. 자품목에 대한 데이터를 가지고 있으며 소요량 집계시에 주로 사용하게 된다.
옵션 클래스	각 모델마다 여러개의 옵션을 가질수 있는데 이러한 모델별 옵션에 대한 데이터의 생성과 삭제, 변경을 담당하는 클래스이다. 이 클래스는 옵션부품 객체의 리스트를 포함하고 있으며 옵션선택시 해당하는 옵션의 교체부품들을 검색하여 해당부품들을 교체시키는 기능을 수행한다.
옵션부품 클래스	옵션 교체부품의 정보를 가지고 있는 클래스이다. 이 클래스는 옵션 클래스에 리스트로 등록되어 사용자의 옵션 선택시 교체부품의 정보를 가지고 부품 교체를 실행하게 된다.

4. 결론 및 추후 연구 과제

본 연구에서는 BOM 구조를 기존의 관계형의 구조와는 다르게 객체지향 방법론을 적용하여 설계하였다. 여기서 제안된 방법에서는 BOM 구성단위 클래스(BOM Item Class) 의 객체를 생성함으로서 BOM 구조를 보다 효율적으로 관리하는 방법을 제시하였다. BOM은 구조가 tree 형태이므로 객체지향 방식으로 표현하기가 더 용이하며 전개시에도 더 빠른 것으로 알려져 있다. 이 설계방식에서 얻을 수 있는 이점들은 다음과 같다.

- 기존의 BOM을 기반으로 하는 시스템의 요소들을 객체화 시킴으로서 이러한 시스템의 요소들을 보다 쉽게 이해할 수 있다.
- 구현된 객체들을 새로운 시스템에 적용하는 것이 가능하다. 이러한 재사용성의 향상으로 인해 추후개발을 위한 노력과 시간을 절약할 수 있다.
- 객체지향형 모델로 구현된 BOM은 확장이나 적용이 쉽기 때문에 업무의 확장이나 변화시 쉽게 적용할 수 있다.
- 기존의 BOM에 비하여 보다 신속한 조회가 가능하며, 임의의 부품에 대한 모든 정보는 해당하는 부품 객체 자체에서 모두 접근이 가능하므로 보다 명시적인 관리가 가능하다.

추후 연구 과제로는 이 구조를 바탕으로 생성, 전개, 복사, 삭제, 소요량 집계 등의 기능을 mamber function으로 클래스에 포함하여 구현하고자 한다. 그리고, 관계형 구조에 따른 기능들과 성능을 비교 분석하고자 한다.

참고문헌

[1] Bruegge, B., Blythe, J., Jackson, J., Shuleft J., "Object-Oriented System Modelling with OMT", OOPSLA, pp359-376, 1992.

[2] Chung, Y., and Fischer, G.W., "A Conceptual Structure and Issue for an Object-Oriented Bill of Materials(BOM) Data Model", Computers and Industrial Engineering: An International Journal, V26, N2, pp321-339, 1994.

[3] ISO 10303-44, Industrial Automation Systems and Integration - Product Data Representation and Exchange, Part 44: Integrated Generic Resources: Product Structure Configuration, 1994.

[4] Katz, R.H., "Toward a Unified Framework for Version Modeling in Engineering Databases", ACM Computing Surveys, V22, N4, pp 375-408, 1990.

[5] Ladd, S. R. , C++ Techniques & Applications, M&T Books, 1990.

[6] Martin, J., Odell, J., Object-Oriented Analysis & Design, Prentice-Hall, 1992.

[7] McHenry, S., "RDBMSs vs. ODBMSs for Product Information Management Systems", Proceedings of AUTOFACT '93 Conference, pp28/13-30, 1993.

[8] Mullin, M., Object-Oriented Program Design with Examples in C++, Addison Wesley, 1989.

[9] Nandakumar, G., "Bill of Material Processing with a SQL Database", Computers and Industrial Engineering, V18, N4, pp471-483, 1990.

[10] Rumbaugh, J., Blaha, M., Premeriani W., Eddy, Lorensen F., Object-Oriented Modeling and Design , Prentice-Hall International, Inc. 1991.

[11] Taylor, D.A., Object-Oriented Technology: A Manager's Guide, Addison-Wesley, 1990.

[12] Tello, E.R., Object-Oriented Programming for Artificial Intelligence, Addison-Wesley Publishing Company, Inc., 1994.

[13] Trappey, A.J.C., Pen, T.K, and Lin, H.D. "An Object-Oriented Bill-Of-Materials System for Dynamic Product Management", Proceedings of the 3rd International Conference in Computer Integrated Manufacturing, V1, pp629-636, 1995.

[14] Wong, S., "Next Generation Enterprise PDM-PDM II", Proceedings of CALS Pacific Korea '96, Seoul, September 1996.

[15] 강영신, 이춘열, "BOM(Bill of Material)의 효과성 분석모형 연구", 석사학위 논문, 국민대학교, 1994.

[16] 곽서경, 객체지향기법을 이용한 BOM과 기술자료의 설계 및 통합에 관한 연구, 금오공과대학교 석사학

위 논문, 1994.

[17] 김선호, 윤희철, "도면정보관리시스템 개발", 대한산업공학회 '94 춘계학술대회 논문집, 1994.

[18] 김선호, 윤희철, "Technical Document Management System을 위한 도면정보관리 시스템 개발", IE Interface 산업공학, V7, N3, 1994.

[19] 김선호, 문희석, "Group Technology를 이용한 설계정보관리시스템 개발", 한국정밀공학회지, 14권, 1호, pp58-68, 1996.

[20] 김선호, 문희석, "Family BOM을 통한 효율적인 설계정보 관리에 대한 연구", 대한산업공학회/경영과학회 '96 춘계학술대회 논문집, pp346-351, 1996.

[21] 김정재, "RDB와 OODB의 한계점과 미래", 경영과 컴퓨터, 4월호, pp242-247, 1996.

[22] 삼성휴렛팩커드,. "CIM 실천전략", (주)컴퓨터엔지니어링, 1990.

[23] 박춘건, 유길호, 이순재, 정영재,. "기업의 정보관리와 활용기법", 아세아문학사, 1995.

[24] 최영근, 허계범,. "객체지향 소프트웨어 공학", 도서출판 한국실리콘, 1995.

[25] 이경우, 정기원, "반복조인(Join)을 이용한 관계형 논리부품구성표(BOM) 데이터베이스설계와 그 효용성 분석", 경영정보화연구, 제2권 1호, pp57-75, 1992.

[26] 이한표, 이춘열, 이국철, "Family BOM데이타베이스 구조에 대한 대안: 목적별 BOM 연결구조의 간접표현 방안", 대한산업공학회 95 추계학술대회 논문집, pp195-202, 1995.

97년 12월 최초 접수, 98년 8월 최종 수정