論 文

大 韓 造 船 學 會 論 文 集
제 30 卷 第2號 1993年 5月
Transactions of the Society of
Naval Architects of Korea
Vol. 30, No. 2, May 1993

선박의 설계 및 생산 정보의 통합을 위한 Product Model 의 구축

유상봉*, 이재원*

Product Model for the Integration of Design and Manufacturing Information in Shipbuilding

by

S.B.Yoo* and J.W.Lee*

요 약

CAD, CAE, CAM, Database, Expert System등 다종의 응용프로그램으로 구성된 CIM 환경에서 Product Model의 역할은 Data 공유를 통한 System Integration이다. 이를 위하여 Product Model은 생산 활동에 포함되는 모든 정보를 관리한다. 이러한 정보의 종류에는 형상정보, 공정정보, 일정정보, 품질정보, 그리고 기존의 관리정보가 포함된다.

Product Model의 Architecture에는 Product Model Kernel, Object Schema, 모델조작언어, 그리고 사용자 인터페이스 등이 있다. Product Model 을 통하여 공유될 객체들은 모델조작언어를 통하여 정의되고 그 정의는 Object Schema에 저장된다. 본 논문에서는 Product Model의 설계와 CAPP(Computer Aided Process Planning)프로그램을 통한 프로토타입의 구현을 설명한다.

Abstract

The role of product model in CIM environemt (where such heterogenous application programs as CAD, CAE, CAM, Database, and Expert Systems are included) is system integration. Product model manages all the information related to manufacuring activities. This information includes shapes, operation, process, scheduling, quality, and mangement.

Product model architecture includes product model kernel, object schema, model manipulation language, and user interface. Objects to be shared are defined using the model manipulation luage and the defintions are saved in the object schema. In this paper, we pres-

발 표 : 1992년도 대한조선학회 추계연구발표회('92. 11. 14)
접수일자 : 1992년 12월 14일
*정회원, 인하대학교 자동화공학과

ent the design and implementation of a prototype. In this prototype, application programs for CAPP (Computer Aided Process Planning) are used.

1. 서 론

CAD, CAE, CAM, Database, Expert Systems 등의 많은 컴퓨터 시스템이 생산성 향상을 위하여 개발되어졌다. 각각의 기술들은 그동안 많은 시행착오를 거쳐 이제 어느 정도 성숙한 단계에 도달했고 생산현장에서 사용되어 그 효과를 인정 받고 있다. 이러한 컴퓨터 응용 시스템 사용자들의 제일 큰 요구사항은 시스템 통합이다. 예를들어 CAD로 제작된 설계를 컴퓨터 File형태로 CAE 시스템으로 전송하여 분석되고 그 결과가 CAD 시스템에 다시 피드백 될 수 있으며 완성된 설계를 갖고 CAM 시스템은 로보트나 NC 공작기계의 프로그램을 생성할 수 있어야 한다. 또한 이러한 시스템들이 Database Systems 또는 Expert Systems들과도 데이타를 교환할 수 있어야 하겠다. 물론 현재에도 동일 회사 제품이거나 몇몇 대표적인 제품들간에 제한된 데이타 호환 기능을 갖추고 있지만 전 생산 시스템의 통합은 아직 많은 연구가 필요하다.

Product Model은 생산 시스템의 통합을 위하여 생산 활동에 포함되는 모든 정보를 관리한다 [7, 9]. 이러한 정보의 종류에는 형상정보, 공정정보, 일정정보, 품질정보, 그리고 기존의 관리정보가 포함된다. 이러한 다양한 종류의 데이타 타입을 모델링 하기 위하여 객체지향 프로그래밍 기술을 도입하는 것이 산업의 추세이다. 객체지향 프로그래밍에 의하면 각각의 정보형태는 하나의 데이타 타입 또는 Object Class로 정의 되고 각 Object Class에 포함되는 객체들을 조작 또는 관리하기 위하여 필요한 Operations들을 Class Function 또는 Method로 정의한다. 또한 Class들은 계층구조를 가질 수 있어서 정보의 형태를 체계적으로 분류할 수 있다. 이러한 정보의 모델링 외에 사용자 인터페이스, 모델조작언어, Porduct Model Kernel, Object Schema 등이 Product Model에 포함된다.

본 논문에는 우선 Product Model에 관련된 세계적인 기술 동향을 요약한다. 대표적인 활동으로서 STEP과 PDES등의 표준화 활동과 NEUTRABAS 또는 일본 조선 CIMS Pilot Model 개발 Project와 같이 Prototype 개발 과제가 있다. 3장에서는 Product Model의 Architecture와 각 구성요소에 대하여 설명한다. 이어서 4장에서는 Product Model의 논리적 구조인 객체모델을 설명한다. 5장에서는 Product Model에서 Object 구조를 정의, 수정, 질의하는 모델조작언어를 기술한다. 6장에서는 현재 실험중인 Prototype을 설명하고 조선 공정계획 시스템을 예로 들어 전체 시스템의 운용을 설명한다. 마지막으로 7장은 결론과 향후 연구계획을 포함한다.

2. Product Model의 기술동향

Product Model은 기존의 CAD/CAM 제품들이 공통으로 이용할 수 있는 화일 포맷을 제정하면서 시작되었다. 현재 널리 쓰이고 있는 것은 1981년 ANSI Standard로 제정된 IGES (Initial Graphics Exchange Specification)이다. 하지만 IGES의 정의 가운데 모호한 부분이 발견되어 ISO는 새로운 스탠다드를 개발하기 위한 작업을 시작하여 1987년 그 초안인 STEP (Standard for the Exchange of Product model data)을 발표하였다. [1,6]. STEP은 또한 그래픽 데이타뿐만 아니라 제품의 전 수명주기를 모두 표현하기 위하여 개발 중이다.

이와 유사한 목표를 갖고 작업중인 다른 하나의 표준화 작업은 PDES (Product Data Exchange Standard)이다 [10]. PDES는 특히 CIM의 구현에 촛점을 두고 있으며 향후 STEP과의 통합을 염두에 두고 여러가지 아키텍처를 가지고 작업중이다.

또한 STEP에 관련된 프로젝트로서 조선 산업에 필요한 정보의 형태와 구조를 분석하여 표준화된 Product Model을 개발하기 위한 것이 ESPRIT (European Strategic Program for Research and Development in Information Technology)의 NEUTRABAS (Neutral Product Definition Database for Large Multi-functional System)이다 [13]. NEUTRABAS에서는 정보의 모델링뿐만 아니라 효과적인 저장과 관리를 위한 데이타베이스 인터페이스도 고려하여 프로토타입을 개발중이다.

이와 같은 시기에 일본 Ship & Ocean Foundation은 조선 CIMS Pilot Model의 개발 연구를 시작하였다[3]. 이 Pilot Model은 Oil Tanker의 중앙부 설계로부터 건조에 이르는 업무를 대상으로 Product

Model, Database System, 구조배치, 치수결정지원 System 등의 Subsystem으로 이루어졌고 현재 Prototype 개발과 검토를 마쳤다. 이 연구에서는 특히 객체지향 분석기법을 이용하여 Product Model을 설계하여 긍정적인 결과를 얻었다.

2.1 STEP

STEP은 현재 ISO (International Standard Organization) 산하에서 개발중인 스탠다드로서 다음과 같은 기존 스탠다드를 기초로 하여 작업이 진행되고 있다.

(1) ISO 8632: CGM (Computer Graphics Metafile)
(2) PHIGS Proposal: 일반화된 3차원 모델링에 대한 프로그램머의 인터페이스
(3) IGES: Initial Graphics Exchange Specification
(4) PDES: Product Data Exchange Specification
(5) SET: 프랑스의 데이타 교환 프로포절로서 IGES와 비슷하지만 더 간략한 구조
(6) VDAFS: 독일의 프로포절로서 표면을 표시하기 위한 표준

이러한 여러가지 기존의 표준을 기초로 작업되는 STEP은 특히 다음 몇가지가 중점적으로 강화된다.

(1) 엔코딩 방식을 최적화 하여 화일의 크기를 줄이고 기능을 강화한다.
(2) 엔티티 타입의 종류를 증가 시킨다. 예를 들면, 자유형태의 곡면을 첨가한다.
(3) 정보의 범위를 증가 시킨다. 예를 들면, 생명주기 데이타, 관리 데이타, 제어 데이타 등을 첨가한다.
(4) 각종 랭귀지 바인딩(Language Bindings)을 개발하여 응용프로그램 인터페이스를 표준화한다.

STEP은 그 범위가 방대하여 부분에 따라 연구 진척에 상당한 차이가 있으며 어떤 부분은 벌써 IGES에서 수행된 것도 있다. 따라서 STEP은 여러 Class로 분류하여 개발 되고 있으며, 현재 개발중인 STEP 1.0에는 Overview, EXPRESS언어, Physical File, Conformance-testing, Information Model, Application Protocol등 9개 부분이 포함되어 있다. 이와 같이 분류된 각 부분은 개발되어 국가투표에 부쳐지게 되고 이의 실용화는 90년대 중반으로 예상되고 있다.

2.2 PDES

PDES는 한 제품과 그것의 전 생산과정에 필요한 정보를 표현, 공유, 교환하기 위하여 개발중인 데이타 표현과 양식에 관한 표준이다. 1980년대 초반 IGES를 사용한 산업체 엔지니어들은 다른 정보들 역시 전산화될 필요성을 느꼈다. 예를들면, 제품 구상단계의 개념설계나 Form Features등이 그런 것이다. 이 시기에 미국 공군은 이러한 다양한 생산 또는 엔지니어링에 관련된 정보를 정의하기 위해 PDDI (Product Definition Data Interface)프로그램을 지원하였다. PDDI는 후에 PDES에 의해 계승되었다.

PDES는 현재 Boeing, General Dynamics, IBM 등 20여 회사들의 보조로 운영되는 독립된 회사에 의하여 개발 중이다. PDES의 목표는 제품과 그것을 생산하기 위한 모든 정보를 설계와 제조 단계간에 공유 또는 교환 함으로서 생산성을 극대화하고 최상의 품질을 달성하는 것이다. PDES의 주요 연구 분야는 다음과 같다.

(1) 제품과 그것의 생산 시스템을 정의한다.
(2) 제품정보의 공유와 교환을 최대한 자동화한다.
(3) 제품의 전 수명주기를 통하여 필요한 모든 응용프로그램을 지원하기 위하여 광범위한 정보들의 상호관계를 정의한다.

PDES는 제품정보의 교환과 공유를 실현하기 위하여 다음과 같은 4가지 아키텍처를 정의하였다. 이중 레벨 1과 2는 정보의 교환을 위한 것이고 레벨 3과 4는 정보의 공유를 위한 것이다.

(1) 레벨 1: Passive File Exchange
(2) 레벨 2: Active File Exchange
(3) 레벨 3: Shared Database (관계형 Database 이용)
(4) 레벨 4: Knowledgabase (객체지향형 Database 이용)

2.3 NEUTRABAS

NEUTRABAS의 목적은 조선산업을 위하여 제품 수명주기의 많은 부분을 커버하는 뉴트랄 (Neutral) 프러덕트정의 데이타베이스를 구체화하는 것이다. 이렇게 함으로써 제품의 전 수명주기에 걸친 다양한 응용프로그램들간의 제품관련 정보를 공유하기나 교환을 용이하게 하여 전 컴퓨터 시스템을 통합할 수 있다. 이러한 목적은 다음과 같이 4가지로 요약될 수 있다.

(1) 조선산업과 관계있는 정보가 표현되는 방법의 표준화
(2) 조선산업의 제품정의데이타의 교환 및 저장을 위한 표준 방법 개발
(3) 이러한 제품정의데이타의 교환 및 저장을 위한 데이타베이스 구조의 구체화 및 개발
(4) Prototyping 수행

이 연구에서 Product Model은 속성들의 집합으로 표현되는 많은 물리적 대상과 추상적 대상으로 구성된다. 이 모델의 사양은 제품의 완전한 표현을 형성하기 위하여 객체와 그들의 속성 그리고 객체간의 관계를 완전하고 명백히 묘사해야 한다. 객체와 속성의 표현 그리고 그들간의 관계선언은 다음 세가지 방법으로 이루어진다.

(1) 사용되는 문장과 그들의 취급에 영향을 주는 제한 및 규칙을 설명하는, 객체와 속성의 자연언어(영어) 표현이 주어진다.
(2) 객체와 속성 그리고 관계가 그래픽 표현방법인 NIAM(Nissen Information Analysis Method)이다. NIAM은 ISO/STEP에 의해 채택된 것으로 E-R 모델의 확장이다.
(3) Product Model은 또한 EXPRESS 정보모델링 언어에 의하여 공식적이고 완전하게 기술된다.

NEUTRABAS의 선체모델에서, 객체는 배의 구조 표현과 관련된 것과 공간 구성에 관련된 것으로 나누어진다. 예를 들어 공간구성 모델은 배의 내부적 구성의 기하학적 및 위상학적 특성을 표현하기위한 객체들의 수집이다. 이것은 내부 폐쇄공간 또는 구획등의 내부적 배치를 위해 결합되는 면의 정의와 3차원 공간에서 제품 부분의 방향과 위치를 지정하는 Reference 시스템을 포함한다. 이러한 방법으로 전체 정보모델을 구성해 나간다.

NEUTRABAS는 단지 정보모델이나 화일 포맷만 정의하는 다른 프로젝트와는 달리, 조선산업에서 제품데이타의 효과적인 관리와 전달을 도모하는 통합 시스템을 구체화한다. 이러한 시스템을 구성할 때 선박의 수명이 20년을 넘고 이것은 컴퓨터 H/W 및 S/W의 수명을 능가하기 때문에 통합 시스템은 컴퓨터 H/W 및 S/W에 종속되지 않아야 한다. 또한 여러 응용프로그램이 공유하고 있는 데이타에 관한 정보모델의 변화 및 발전을 가능하게 하기 위하여 데이타베이스와 정보모델 그리고 응용프로그램을 독립된 Subsystem으로 구성하고 상호간의 접속방법(Interface)을 정의하여야 한다.

MEUTRABAS프로젝트에서는 이러한 시스템의 효용성을 입증하기 위하여 2개 이상의 응용 프로그램을 포함하는 Prototype 시스템을 구현하는 중이다. 이 원형 시스템에는 철구조 설계 시스템인 CADIS와 생산 계획 시스템인 CRESTA 그리고 선박 초기설계 시스템인 SPAN을 포함할 계획이고, 데이타베이스 시스템으로는 ORACLE 관계형 DBMS를 선정하였다. 따라서 이 Prototype은 PDES의 구분에 의하면 레벨 3인 아키텍처이다. 이 Prototype은 STEP의 개발 부분중 일반 정보모델과 응용 정보모델중 Part Number 102인 Ship Structure과 밀접한 관계가 있으며 STEP의 개발에 부분적으로 기여할 것이다.

2.4 일본 조선 OMS Pilot Model 개발 프로젝트

일본 Ship & Ocean Foundation에서 주관한 조선 CIM 프로젝트에서는 Product Model을 "형상이나 속성 뿐만 아니라 이것의 정의 과정, 부재 간의 관계나 공정 순서, 공사 일정과의 연결, 다른 설비나 인적자원을 모두 컴퓨터 내의 모델로서 취급하는 시스템"으로 그 기능을 정의한다. 다시 말하면 전체 Pilot Model의 연결과 통합을 의미하는 것이다. 특히 선각과 의장을 동일 모델로서 취급하고 설계 정보와 공정/생산관리 정보를 연계시키기 위하여 객체 지향적인 사고 방식을 응용한다. 이러한 요구를 실현하기 여 다음과 같은 기능을 필요로 한다.

(1) 모델 관리 기능
(2) 3차원 도형 처리 기능
(3) 구조 처리 기능
(4) 설계 데이타와 생산관리, 공정 데이타를 연결

시킬 수 있는 기능.
(5) 사람, 설비, 물량 모델의 통합화
(6) 실용적인 처리 속도 및 필요 Memory 산정
(7) 모델 조작 언어의 개발

일본의 프로젝트는 이와 같은 기능을 갖는 Product Model을 즉시 실현하기보다는 Prototype을 통해 실현 가능성, 효과, 문제점 등을 검토하는 것이 그 목적이다. 이 프로젝트는 1989년 부터 3년간 수행하였다. 현재 연구 결과나 평가등이 보고서나 학술 논문을 통하여 발표되고 있다. 이 Prototype은 객체지향 분석기법을 이용하며 PDES의 아키텍처 구분에 의하면 레벨 4에 해당한다. 이중 Product Model과 데이타 베이스에 관련된 부분을 종합하여 볼 때, 조선 CIMS 구현에 Product Model의 필요성은 확인되었고 객체 지향 프로그래밍 기법의 유용성도 입증되었다. 특히 CIMS화가 완료될 경우 선박의 건조공수가 30% 단축될 것으로 예측하였다. 몇가지 문제점으로서 다음 사항들이 지적되었다.

(1) 현재의 기술수준을 이용하여 전체 생산 과정을 Product Model을 통해 구현 하였을 때 그 방대한 데이타 양으로 인하여 처리 속도가 매우 늦어질 것으로 예상된다.
(2) 데이타베이스로서 객체지향 데이타베이스인 GemStone을 이용할 경우 기능은 향상 되었지만 처리속도가 현저히 떨어지는 문제가 있다.

이중 문제점 (1)은 향후 컴퓨터 프로세서의 성능 향상과 데이타 저장기술의 발전으로 인하여 개선될 수 있다. 문제점 (2)는 GemStone이 객체지향 언어인 Smalltalk으로 구현되었기 때문이다. Smalltalk은 Object Binding을 프로그램 실행시 행하기 때문에 기존의 프로그램 언어보다 처리속도가 늦어진다. 이러한 문제는 C++ 로 구현된 객체지향 데이타베이스 시스템을 사용함으로서 해결될 수 있다.

3. Product Model의 Architecture

Product Model의 표준화는 현재 연구가 진행 중이며 특히 객체지향 분석기법을 이용한 PDES의 레벨 4 아키텍처는 제한된 기능의 Prototype만이 존재한다. 이러한 국제 표준화 작업을 감안하고 다른 Prototype에서 제기된 문제점을 보완하여 다음과 같은 Product Model을 설계하였다. Product Model의 Architecture는 Fig 1과 같으며, 여기서 데이타베이스 시스템과 각종 응용프로그램은 분산된 컴퓨터 시스템에서 동작하며 Product Model과 컴퓨터 네트웍을 통하여 연결된다.

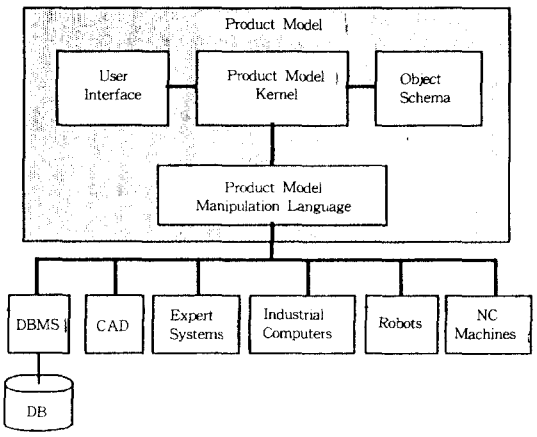

Fig. 1 The architecture of product model

Fig 1에서 Product Model은 Product Model Kernel, Object Schema, User Interface, Model Manipulation Language (모델조작언어) 등으로 구성된다. 이들 각각의 기능은 다음과 같다.

(1) Product Model Kernel: Product Model의 가장 핵심 부분으로 다음과 같은 기능을 갖는다.

- 모델조작언어 Interpreter: 응용프로그램으로부터 전달된 모델조작언어를 해석하여, Object Schema에 새로운 정의를 저장하거나 기존의 정의를 수정하고 질의어인 경우에는 적절한 응답을 보낸다.
- 데이타베이스 Interface: 상용화된 데이타베이스 시스템은 관계형과 객체지향형의 두가지가 있다. 관계형 데이타베이스 시스템의 경우 Relation 테이블의 구조가 Object Schema에 정의된 클래스 구조와 다르기 때문에 객체의 Attribute를 검색할 때 Inheritance를 고려하여 해당 테이블을 찾아야 한다. 또한 Object Schema의 클래스 구조를 변경할 경우 전체 시스템의 Integrity를 위하여 관련 데이타베이스의 사용을 제한하여야 한다.
- Object Allocation: 일반적으로 Product Model은 한개 이상의 데이타베이스 시스템을 이용한다. 이러한 상황에서 새로운 객체가 (예를들어 CAD 설계도면) 생성되었고 응용프로그램에 의하여 저장 장소가 지정 되지 않을 경우 이 객체의 저장장소를 지정한다. 이때 고려할 사항은 Object Size,

Storage Systems, 관련 Applications, Computer & Network Configuration 등이다.

– Administration: Product Model 전체의 원활한 운영을 지원하다. 새로운 사용자 또는 응용프로그램을 등록하고 이때 보안코드, 사용제한시간 등도 부여한다.

(2) 모델조작언어: 모든 응용프로그램은 Product Model과 모델조작언어를 이용하여 대화한다. 응용프로그램은 Product Model에 새로운 클래스 정의를 첨가할 수 있고 기존의 정의를 수정할 수 있다. 다른 응용프로그램에 의하여 생성된 객체를 이용할 경우 Product Model에 정의된 그 객체의 Attributes, Methods, 또는 현재의 저장위치를 확인한 후 사용한다.

(3) Object Schema: 모델조작언어에 의하여 정의된 클래스 정보를 저장한다. 여기에는 Class Hierarchy, Attributes, Methods등이 포함된다. 또한 분산 컴퓨터 환경 하에서 각 객체를 관리하는 데이타베이스 시스템을 표시한다. Object Schema는 Product Model에 의하여 관리되는 모든 데이타에 관한 정보를 갖고있는 Metadata이며 각각의 응용 프로그램은 Object Schema를 통하여 필요한 데이타를 찾는다.

(4) User Interface: Product Model에 연결된 모든 객체와 응용프로그램을 네트웍상 어디서나 사용할 수 있는 User Interface를 제공한다. Menu형태의 Graphical User Interface를 개발하여 사용자가 복잡한 명령어나 응용프로그램 시작방법을 모르더라도 화면에서 메뉴를 선택하여 이용할 수 있게 지원한다.

4. Object Model

Object Model은 두가지 Submodel을 포함하고 있다. 즉, Structural Model과 Behavioral Model이다. 종래의 데이타 모델링은 주로 데이타의 구조적인 (Structural) 측면에 촛점을 두었다. 데이타의 다른 측면인 행동 (Behavior)은 데이타베이스 Transaction에 의하여 구현되지만 잘 표현되지 않았다. 객체지향 프로그래밍은 이러한 데이타의 두가지 측면을 체계적으로 표현할 수 있다 [5].

4.1 Structural Model

객체지향 프로그램의 가장 하위 레벨에는 객체 (Object)가 있다. 이러한 객체를 종류별로 모아논 것이 Class이다. 같은 Class에 속하는 객체들은 같은 종류의 속성정보(Attributes)를 갖는다. 이러한 클래스는 프로그래밍에서의 데이타타입 (Data Type)에 해당한다. 모든 객체는 그 객체가 속한 클래스를 갖는다.

이러한 클래스들간에는 2가지의 대표적인 관계가 있다. 이것은 일반화 (Generalization)와 집합 (Aggregation) 관계이다. 일반화는 공통되는 특성을 가진 여러 클래스가 모여서 하나의 큰 클래스를 이루는 것이다. 예를들어 블럭은 크게 곡블럭과 평블럭으로 나눌 수 있다. 여기서 블럭, 곡블럭, 그리고 평블럭을 모두 클래스라 할때 블럭은 곡블럭과 평블럭의 일반화이다. 동시에 곡블럭이나 평블럭은 블럭의 특별화 (Specialization)라 할 수 있다. 이러한 특별화 관계는 "–의 종류"(Kind-of)를 나타낸다. 위 예에서 곡블럭이나 평블럭은 블럭의 한 종류이다.

이러한 클래스간의 관계를 객체지향 프로그램에서는 Superclass와 Subclass로서 표현한다. 즉 클래스 곡블럭과 평블럭은 클래스 블럭의 Subclass이다. 반대로 클래스 블럭은 클래스 곡블럭 또는 평블럭의 Superclass이다. 이때 Subclass는 Superclass 속성 (Attributes)을 물려받는다 (Inherit). 예를들어 곡블럭과 평블럭의 공통되는 속성인 블럭번호, 길이, 너비 등은 클래스 블럭에 정의하면 Subclass들에서 다시 정의할 필요가 없다. 하지만 곡블럭의 곡면을 표시하는 B-Spline 등은 클래스 곡블럭에서 정의한다. 일반적으로 한 클래스는 여러개의 Superclass를 가질 수 있다. 이러한 경우 같은 이름을 가진 속성을 두개 이상의 Superclass로 부터 물려받을 수 있다 (Multiple Inheritance). 이때에는 사용자가 정해준 순서나 시스템이 갖고 있는 방식에 따라 하나를 선택한다.

일반화-특별화 관계와 더불어 중요한 관계가 집합 (Aggregation) 관계이다. 집합 관계는 구성요소를 나타낸다. 예를들어 Hull은 Bottom, Main-Deck, Side-Shell 등으로 구성된다. 이때 Bottom은 Hull의 한 구성요소로서 Hull의 "일부분"(Part-of) 임을 나타낸다. 이러한 집합관계는 객체지향 프로그래밍에서는 하나의 속성정보로서 표시된다. 즉 클래스 Hull은 속성으로서 Bottom, Main-Deck, Side-Shell을 갖고 클래스 Bottom, Main-Deck, Side-Shell은 속성 Hull 을 갖음으로서 서로의 포함하고 포함되는 관계를 나타낸다. 일반화와 집합 관계 외에 응용분야에 따라 객체간에 여러가지의 관계가 있을 수 있다.

객체간의 관계는 대상객체의 갯수에 따라 일대일, 일대다, 그리고 다대다의 3가지로 나눌 수 있다. 이중 일대일과 일대다는 속성정보로서 나타낼 수 있다. 예를들어 객체와 고유번호는 일대일관계이고 하나의 블럭과 그 블럭에 속하는 부재들은 일대다 관계이다. 다대다 관계로는 블럭과 작업자의 관계가 있다. 한 블럭은 여러명에 의하여 조립되며 한 작업자는 여러개의 블럭작업에 속할 수 있기 때문이다. 이러한 다대다 관계는 관계를 나타내는 새로운 클래스를 만들고 이 클래스는 관계가 발생한 모든 짝 (객체와 객체의 pair)을 포함한다. 작업자와 블럭간의 관계를 표현하기 위하여 클래스 "작업배정"을 만들고 속성으로서 작업자, 블럭, 시작시간, 끝시간 등을 정할 수 있다. 객체간의 여러가지 관계는 이상과 같은 방법으로 객체지향 프로그래밍으로 표현되고 이것은 클래스 정의 (Class Definition)에 포함된다. 이러한 클래스 정의는 프로그램내에 포함되어 있기 때문에 일반 사용자는 전체 객체의 종류나 구성을 알기가 어렵다. 이점을 고려하여 사용자가 객체의 전체구성을 이해하고 필요한 객체를 쉽게 찾게하기 위하여 Frame 표시방법을 사용한다. 이 Frame 표시방법은 상위레벨에서 하위레벨까지 계층적으로 구성되어 있어서 화일 시스템의 Directory 구조와 유사하다. 또한 Frame은 Slot으로 객체의 속성정보를 나타내며 한 Frame은 하나의 클래스에 해당한다. Fig 2는 Product Model의 최상의 레벨에서 객체를 Frame 표시방법으로 나타낸 한 예이다.

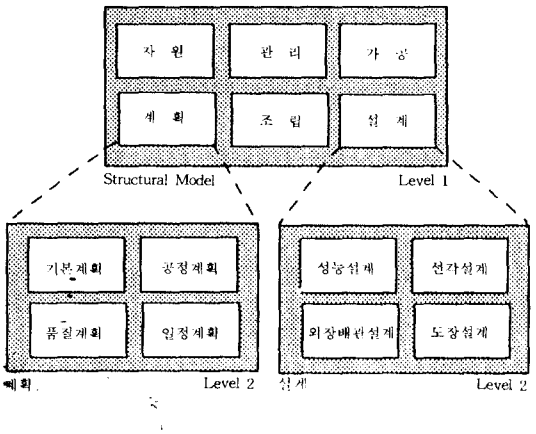

Fig. 2 Representation of object structure in frames

4.2 Behavioral Model

객체지향 프로그램에서 행동모델은 각 클래스에 정의된 Method로서 구현된다. Method는 해당되는 클래스에 속하는 객체에 대해서만 동작하는 Procedure 또는 Function이다. 같은 이름을 갖는 Method를 여러 클래스에 정의할 수 있다. 이러한 경우 대상 객체가 속하는 클래스에 정의된 Method가 찾아져서 실행된다. 예를들어 연산자 "+"는 클래스 정수에서는 덧셈을 실행한다. 같은 연산자 "+"를 클래스 부재에서는 부재용접을 실행하는 Method로 정의할 수 있다. 이때 두개의 부재인 부재$_1$과 부재$_2$를 용접 시키려면 프로그램 안에서 "부재$_1$+부재$_2$"를 실행시킨다.

Method도 클래스의 속성과 같은 방법으로 상위 클래스로 부터 Inherit 된다. 앞 절에서 예로 들었던 클래스 곡블럭과 평블럭이 같은 프로그램에 의하여 Block Division을 실행한다면, Method인 Block Division을 두 클래스의 Superclass인 클래스 블럭에서 한번 정의하면 된다. 상위 클래스에서 정의된 Method를 바꿀 필요가 있을 때는 하위 클래스에서 같은 이름의 Method를 다시 정의하면 된다. 객체지향 프로그램은 이러한 Method의 결합이라 할 수 있다.

객체의 Behavioral Model은 이러한 Method에 의하여 정의된다. 사용자가 필요한 기능을 프로그램하여 구현하면 되지만 대부분의 응용프로그램에서는

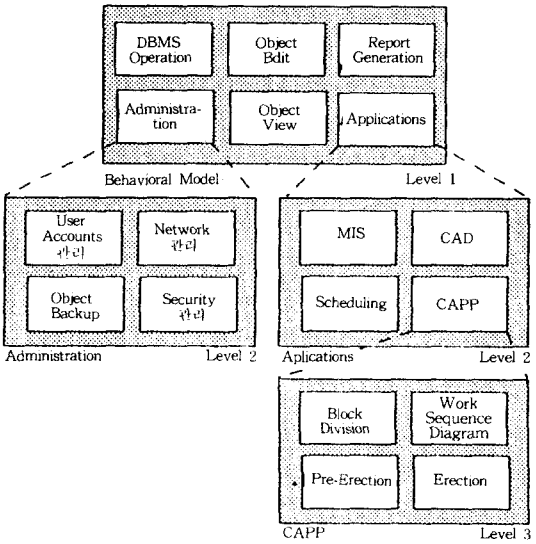

Fig. 3 Representation of transactions in frames

기본적인 Transaction이 Command로서 제공된다. Product Model에서는 제공되는 Command를 Frame을 이용하여 계층적으로 표시한다. 이러한 계층적 표시는 시스템 구현시 메뉴로 나타낼 수 있기 때문에 편리한 사용자 인터페이스가 될 수 있다. Fig. 3은 Product Model에서 제공되는 Transaction을 Frame 표시방법으로 나타낸 것이다.

5. 모델 조작 언어(Model Manipulation Language)

5.1 모델 조작 언어의 역할

모델조작언어는 응용프로그램이 Product Model과 의사소통을 하기 위하여 쓰이는 언어이다. 응용프로그램과 Product Model간의 의사소통에는 크게 세 가지 종류가 있다. 첫째, 새로운 객체 클래스의 정의이다. 어떤 응용 프로그램에 의하여 새로운 종류의 객체가 생성될 경우, 먼저 그 객체를 포함하는 클래스가 Product Model에서 새롭게 정의 되어야 한다. 이때 새로 정의되는 클래스에는 객체의 정의에 필요한 정보 (즉, Class-name, Superclass, Attributes, Methods) 외에 발생되는 객체들이 실제 저장되는 컴퓨터와 데이타베이스가 지정되어야 한다. Product Model에는 클래스의 정의만 기록되고 객체들은 네트웍상의 어느 컴퓨터에 저장되는 것이다. 이러한 클래스의 정의는 Product Model내의 Object Schema 내에 기록된다.

둘째, 기존 클래스 정의의 수정이다. 이미 존재하는 클래스의 정의중 일부가 바뀌었을 때, 수정할 수 있는 권한을 가진 응용프로그램은 모델조작언어를 이용하여 Object Schema내의 클래스 정의를 수정할 수 있다. 예를들어, 새로운 Attribute가 첨가되거나 기존 Attribute의 Type이 바뀔경우 모델조작언어를 이용하여 클래스의 정의를 수정한다.

세째, 기존 클래스 정의에 대한 질의이다. 한 응용프로그램이 어떤 클래스에 속한 객체가 필요할 때, Product Model에 현재 이 객체가 저장된 컴퓨터와 데이타베이스 이름을 질의할 수 있다. 이렇게 하여 저장된 데이타베이스를 알고난 후에, 이 응용프로그램은 해당 데이타베이스에 적절한 질의어를 보냄으로서 원하는 객체를 받을 수 있다. 이외에도, 어떤 클래스의 Attribute 이름이나 Superclass등을 질의하여 알 수 있다.

이상에서 열거한 모델조작언어는 응용프로그램뿐만 아니라 Product Model이 제공하는 User Interface를 통하여 사용자가 직접 이용할 수 있다. 이러한 모델조작언어의 기능을 제공하기 위하여 Product Model에는 Object Schema외에 Lexical Analyzer와 Interpreter가 있다. Lexical Analyzer는 입력되는 모델조작언어를 문법적으로 해독하고 이를 내부 Token으로 바꾸어 준다. Interpreter는 Lexical Analyzer로부터 전달받은 Token의 의미를 해석하여 필요한 작업을 수행한다. Product Model중 모델조작언어와 관련된 기능을 나타낸 것이 Fig 4이다.

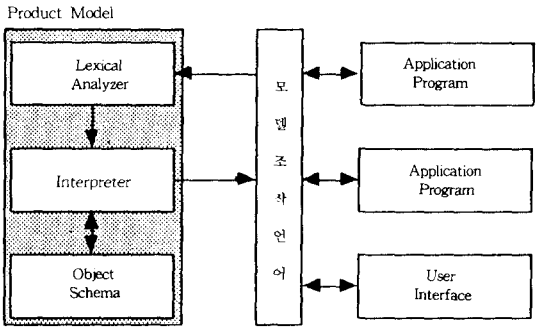

Fig. 4 Related modules to object manipulation language

5.2 모델조작언어의 문법

본 절에서는 모델조작언어의 문법을 BNF (Backus-Naur Form)로 표시한다. BNF는 주어진 프로그램을 해석하는 첫 단계에서 Lexical Analyzer에 의하여 이용된다 [4]. BNF는 Symbol과 화살표로 구성되며, Symbol은 다시 Terminal Symbol (또는 Token)과 Nonterminal로 구분된다. 여기서 화살표 (→)는 "은 다음과 같다"라는 의미를 갖는다. 즉 화살표 왼쪽에 나오는 Symbol은 오른쪽에 나오는 Symbol들과 같다는 의미이다. 이때, 화살표 왼쪽에 한번도 나타나지 않는 Symbol을 Terminal Symbol 또는 Token이라 하고, 그렇지 않은 Symbol들을 Nonterminal이라 한다.

BNF에서 화살표 오른쪽에는 Symbol과 부호 (즉, 괄호, Comma, Colon, Semicolon 등)외에 세로막대 (|), 대괄호 ([]), 그리고 Empty (ε)가 있다. 세로막대는 선택 (or)의 의미가 있고, 대괄호는 반드시 필요하지는 않은 항목 (optional)을 나타내며, Empty

는 아무 Symbol도 포함하지 않는 상태이다. 이상과 같은 정의하에 모델조작언어의 문법을 표시하면 다음과 같다.

```
program→
        statement_list
statment_list→
        statement_list ; statement
        | statement
statment→
        define_statement
        | modify_statement
        | query_statememt
define_statement→
        DEFINE class_name : parent_class
                (attribute_list)
        METHOD method_list
        WHERE node_name ; db_identifier
attribute_list→
        attribute_type attribute_name
        | attribute_list,
         arttribute_type attribute_name
method_list→
        method_list ; method_declaration
        | ε
method_declaration→
        [method_type] node_name :
        method_name (parameter_list)
parameter_list→
        parameter_list, parameter_type
        parameter_name
        | ε
modity_statement→
        MODIFY class_name
        [ : parent_class]
        [(attribute_list)]
        [METHOD method_list]
        [WHEE node_name :
        db_identifier]
query_statement→
        QUERY class_name_or_question
        [ : parent_class_or_question]
        [(attribute_list_or_question)]
        [METHOD
         method_list_or_question]
        [WHERE
         node_name_or_question]
class_name_or_question→
        class_name
        | ?
parent_class_or_question→
        parent_class
        | ?
attribute_list_or_question→
        attribute_list
        | ?
method_list_or_question→
        method_list
        | ?
node_name_or_question→
        node_name : db_identifier
        | ?
```

위에서 BNF로 정의된 모델조작언어를 간단히 설명한다. 모델조작언어에서 프로그램은 일련의 Statement로 구성된다. Statement의 기능은 정의, 수정, 질의 등 세가지가 있는데 각각의 statement는 그 첫 머리에 DEFINE, MODIFY, 또는 QUERY라고 써서 구별한다.

새로운 클래스를 정의하는 Statement는 클래스 이름, Superclass의 이름, Attribute 이름과 Type의 List, Method List, 그리고 이 클래스의 객체가 저장된 컴퓨터와 데이타베이스 이름을 포함한다. 여기서 Method는 클래스 객체에 한정되어 사용되는 Procedure로서 Product Model에는 그 Declaration만 정의되고 컴파일된 Executable은 네트웍상의 한 컴퓨터에 존재한다. 응용프로그램과 해당 Method가 서로다른 Computer에 존재할 경우 UNIX의 RPC (Remote Procedure Call)을 이용하여 Method을 수행시킨다. 키워드인 WHERE 다음에 기록된 컴퓨터와 데이타베이스 이름은 Network Programming에 사용되어 정보교환을 수행한다.

기존 클래스를 수정하는 Statement는 앞에서 설명한 클래스 정의 Statement와 유사한 양식을 가지며 키워드 MODIFY가 앞에 온다. 이러한 수정 Statement를 이용하여 기존 클래스 정의를 수정하는 방법은 Pattern Match 방법에 의한다. 클래스 이름은 유일하기 때문에 클래스 이름과 수정하고자 하는 부분을 수정 Statement에 포함함으로서 클래스 정의중 일부를 고칠 수 있다. 클래스 이름을 바꾸고

자 할 경우에는 클래스 이름을 제외한 나머지 부분을 현재의 정의와 일치시키고 클래스 이름만 바꿔줌으로서 정확하게 원하는 클래스의 이름을 수정한다.

기존 클래스 정의를 질의하는 Statement도 수정 Statement와 같이 Pattern Match에 의하여 동작하며 키워드 QUERY가 앞에 온다. 알고 있는 클래스에 대하여 Parent Class, Attributes, Methods, 저장장소 등을 알고자 할 경우 클래스 이름을 쓰고 해당하는 부분에 물음표 (?)를 쓴다. 또한 특정한 Parent Class, Attributes, Methods, 또는 저장장소를 갖는 클래스를 찾을 수 있다. 이때 해당되는 클래스가 하나 이상일 경우 모든 클래스 이름이 List로서 주어진다.

6. Prototype 개발

6.1 개발 범위

Product Model은 CIMS 환경하에서 소속 Application의 정보 공유 또는 교환을 목적으로 한다. 이러한 목적을 실현하기 위하여는 Product Model 뿐만 아니라 조선 산업에 관계하는 모든 업무의 전산화가 동시에 추진되어야 한다. 이러한 거대한 규모의 작업을 실시하기전에 필요한 요소기술의 확보는 필수적이다. 본 Prototype에서는 조선 CIMS 구현시 필요한 Product Model의 요소기술로서 객체모델, 모델 조작언어, Object Schema, Network Programming, User Interface 등을 검증한다.

현재 본 Prototype은 조선 산업의 CAPP (Computer Aided Process Planning)를 대상으로 개발중이며 네트웍으로 연결된 두대의 SUN Workstation에서 구현되고 있다. Prototype에 포함되는 Software Module은 다음과 같다.

(1) Product Model:
- -Product Model Kernel
- -Object Schema
- -모델 조작언어 (Lexical Analyzer, Interpreter)
- -User Interface

(2) CAPP System: 조선 공정계획 시스템
- -Block Division System
- -Working Sequence Diagram System

(3) Nexpert: Expert System Development Shell

(4) Pro/Engineer: Parametric Feature Based CAD System

위에 열거한 Software Module중 Nexpert와 Pro/Engineer는 상용 Software로서 CAPP System에서 이용되고 있다. 현재의 Prototype의 구현에서는 데이타베이스 시스템은 포함하지 않았고 대신 UNIX File System과 Nexpert와 Pro/Engineer의 내부 Database를 이용한다. Prototype Hardware 및 Software의 구성은 Fig. 5와 같다.

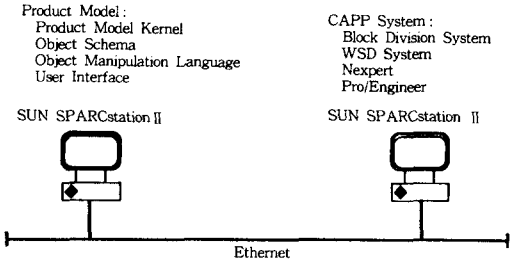

Fig. 5 H/W and S/W configuration of the prototype

본 Prototype을 두대의 Workstation에서 구현한 것은 Network Programming을 검증하기 위해서다. 많은 Subsystem을 포함한 조선 CIMS는 각 Subsystem의 크기, 계산량, 지역적 분산 등을 고려할때 하나의 Mainframe보다는 네트웍으로 연결된 다수의 Workstation상에 구현하는 것이 바람직하다. 네트웍상에 분산된 다양한 응용프로그램간의 Interface를 위하여 UNIX에서 제공되는 Socket을 이용한다[11]. Socket은 응용프로그램들이 같은 Computer에 있을때나 네트웍상의 다른 Computer에 있을때에 모두 사용할 수 있으며 Communication Protocol로서 동기 (Synchronous) 및 비동기 (Asynchronous) 통신을 제공한다. 이러한 Network Programming을 위하여 각 Computer의 Internet Address 지정이 필요하다.

6.2 Product Model 사용 예

본 절에서는 조선 CAPP 시스템에 포함된 두개의 응용프로그램이 Product Model을 이용하여 데이타를 교환하는 경우를 예로 들어 설명한다. 불럭분할 시스템은 설계된 선박에 대하여 선체를 주어진 제약조건 하에서 블럭형상으로 분할하는 시스템이다. 블럭분할시의 제약조건은 블럭크기, 블럭형상, 그리고 작업 용이성등으로 주어진다. 블럭분할 시스템으로부터 출력되는 정보는 분할선, 용접선 길이, 그리고

각 블럭의 물리량 정보이다.

블럭분할 시스템에 의하여 생성된 각 블럭은 WSD 시스템의 입력자료가 된다. WSD 시스템은 주어진 블럭에 대하여 소조립(Part Assembly), 중조립(Subassembly), 그리고 대조립(Block Assembly)로 나뉘어지는 각 조립단계의 순서를 정한다. 이를 위하여 블럭은 WSD 시스템에서 부재나 부분 조립 단위로 나뉘어진다[2]. Fig.6은 CAD System, 블럭분할 System, 그리고 WSD System간의 정보교환을 나타내는 데이타 흐름도이다.

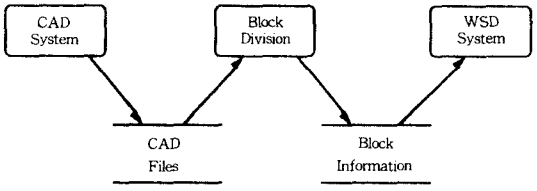

Fig. 6 Data sharing through the product model

위에서 설명한 블럭분할 시스템과 WSD 시스템은 객체 블럭을 공유한다. 즉, 블럭은 블럭분할 시스템의 출력이고 WSD 시스템의 입력이다. 이렇게 하나의 객체를 여러 응용시스템에서 공유할 수 있게 하기 위하여 이러한 객체의 정의를 Product Model에 포함시킨다. 블럭분할 시스템은 Product Model에 정의되어 있는 모든 Attribute를 생성하여 Product Model에 지정된 장소에 저장한다. WSD 시스템은 이렇게 저장된 블럭을 지정된 장소로부터 검색하여 사용한다. 다음은 Product Model에서 모델조작언어를 이용하여 블럭을 정의한 예이다.

```
DEFINE Block : TableObject(int block _code,
                           real weight,
                           SizeType size,
                           ListOfParts part _list,
                           ListOfTopology block _topology,
                           DrawingType drawing)
  METHOD / /Create a new Block Object
                           void sun2: /bin2 /Block(
                           int n _blk _code,
                           real n _weight,
                           SizeType n _size
                           ListOfParts n _part _list,
                           ListOfTopoloty n _block _topology,
                           DrawingType n _drawing):
               / /Destroy a Block Object
                           void sun2: /bin2 /~Block ():
               / /Method for deriveing joint
               / /length of a Block Object
                           real sun2: /bin2 /joint _length(
                           Block input _block):
  WHERE sun2: /cad /proengineer
```

위 예는 Product Model에서 정의된 객체 클래스인 Block이다. 클래스 Block의 Superclass인 TableObject는 클래스에 포함된 객체들이 데이타베이스의 한 테이블을 형성하는 클래스이다. 클래스 TableObject의 Method로는 데이타베이스의 기본 Operation인 Union, Set Difference, Cartesian Product, Projection, 그리고 Selection이 제공된다. 클래스 Block의 Attribute로는 Block의 ID, 무게, 크기, 부재 리스트, 부재간의 위상관계, 그리고 CAD로부터의 드로잉 화일이 있다. 이 클래스의 Method로는 새로운 객체의 Constructor와 기존 객체의 Destructor외에 블럭 전체에 필요한 용접길이를 계산하는 Function이 있다. 클래스 Block에 속하는 객체들은 컴퓨터 sun2의 디렉토리 /CAD내의 Proengineer에 저장되어 있다.

7. 결 론

본 논문은 조선 CIMS 구현에 필요한 요소기술인 Product Model의 구현에 대하여 기술하였다. Product Model에는 두가지 측면이 있다. 첫째는 STEP을 중심으로 연구되고 있는 데이타 타입의 정의이다. 데이타 타입에는 CAD/CAM 시스템의 구현에 필요한 화일 포맷도 포함되어 있다. 두번째는 공통의 데이타 타입을 갖는 정보를 다종의 응용프로그램이 공유하는 방법론이다. 후자는 일본 조선 CIM Pilot Model 등과 같은 Prototype 개발 프로젝트에서 연구 중이며 본 논문의 촛점도 여기에 있다.

Product Model에 있어서 객체의 계층구조는 객체지향분석기법에 따라 특수화와 일반화 관계에 의하여 클래스와 서브클래스의 관계를 가진다. 이러한 객체의 구조를 정의하기 위하여 모델조작언어를 사용한다. 모델조작언어에 의하여 정의된 객체모델은

Object Schema에 저장되어 다수의 응용프로그램에 의하여 공유된다. Product Model의 구현에 꼭 필요한 다른 요소기술은 Network Programming이다. 이는 현재 컴퓨터 환경의 추세가 네트웍으로 연결된 분산 시스템으로 가고 있기 때문이다. 이러한 Product Model의 Prototype 구현과 동작을 조선 공정계획 시스템을 예로 들어 설명하였다.

현재 Prototype에서 빠져있는 응용프로그램중의 하나가 데이타베이스 시스템이다. 독립적인 데이타베이스 시스템을 이용하는 대신 UNIX 화일 시스템과 각 응용프로그램에서 제공되는 데이타베이스 기능을 이용하여 데이타를 저장/검색한다. 이러한 방법은 소수의 응용프로그램이 포함되어 있는 경우에는 사용할 수 있지만, 현실적인 CIMS의 구현시에는 데이타베이스 시스템의 기능들이 (즉 Concurrency Control, Crash Recovery, Version Control, Integrity Constraints, Security 등) 꼭 필요하다.

대표적인 데이타베이스 시스템은 크게 관계형 시스템과 객체지향형 시스템으로 나눌 수 있다 [12]. 관계형 시스템은 현재 사무 관리 시스템을 중심으로 널리 사용되고 있으며 객체지향 시스템은 다양한 데이타 타입과 강력한 모델링 파워를 바탕으로 멀티미디어어와 공학 응용분야에 이용될 전망이다 [8]. Product Model에 관계형 데이타베이스 시스템을 사용할 경우 Object Model과 같은 클래스 구조를 가질 수 있지만 현재 상용화 초기 단계로서 질의어의 산업·표준화가 이루어지지 않았다. 본 연구는 이러한 데이타베이스 시스템과의 연계방안을 구체적으로 연구하여 현재의 Prototype을 확장하고자 한다.

후기

본 연구를 지원하여 주신 대우조선(주) 관계 제위께 감사드립니다.

참 고 문 헌

[1] 강원수, 서승완, 신동우, 이규옥, 이규열, "제품모델을 기초로 한 선박모델의 표현 방법론", 대한조선학회 1992년도 춘계연구발표회지, pp 14-23, 1992

[2] 권창완, "조립 공정계획 전문가 시스템의 개발에 관한 연구 (선체 선각 블럭에의 응용)", 인하대학교 석사학위논문, 1992.

[3] 일본조선진흥재단, "1991년도 조선 CIMS PILOT MODEL 개발 연구 보고서", 1992

[4] Alfred Aho, Ravi Sethi, and Jeffrey Ullman, "Compilers-Principles, Techniques, and Tools", *Addison Wesley, 1986*

[5] Prabhat K. Andleigh and Michael Gretzinger, "Distributed Object-Oriented Data Systems Design", *Prentice-hall*, 1992

[6] Dimitris Chorafas, "The Engineering Database", *Butterworths*,1988

[7] Fumihiko Kimura, "Factory Automation and CAD-Product Modelling for Advanced Manufacturing Automation", 한국정밀공학회지 제8권 제2호 pp 7-26, 1991

[8] Charles Lamb, Gordon Landis, Jack Orenstein, and Dan Weinreb, "The ObjectStore Datalase System", *Communications of the ACM*, Vol 34, No 10, 1991

[9] Kai Mertins and Wolfram Sussenguth, "Integrated Information Modelling for CIM", *Computer-Integrated Manufacturing Systems*, Vol 4, No 3 pp 123-131, 1991

[10] Society of Manufacturing Engineers, "PDES: The Enterprise Data Standard," 1989

[11] W.Stevens, "UNIX Network Programming", *Prentice-Hall*, 1991

[12] Jeffrey D. Ullman, "Database and Knowledge-Base Systems Volume 1," *Computer Science Press*, 1988.

[13] Welsh, M., Lynch, J., Burn, P., "A Data Model for the Integration of the Pre-Commisioning Life-cycle Stages of the Shipbuilding Product", *Proc. of SNAME* 1991 Ship Production Symposium, San Diego, Califonia, Sep. 1991