論 文

大 韓 造 船 學 會 論 文 集
第 29 卷 第4號 1992 年 11 月
Transactions of the Society of
Naval Architects of Korea
Vol. 29, No.4, November 1992

제품모델을 기초로 한 선박모델의 표현방법론

강원수*, 서승완*, 신동우*, 이규옥*, 이규열*

The Representation Methodology for the Ship Model based on Product Model

by

W.S.Kang*, S.W.Suh*, D.W.Shin*, K.O.Lee* and K.Y.Lee*

요 약

선박과 같은 복잡한 대상물을 설계하거나 생산하는데 사용되는 소프트웨어 시스템들간의 상호 정보교환을 위한 표현방법을 구체화하는 제품모델(Product Model)의 표준화 동향을 살펴보았으며, 이를 근거로 제품모델을 기초로 한 선박모델을 정의하기 위한 새로운 표현방법을 정립하여 제안하였다. 제안된 선박모델의 표현방법론을 통해 응용측면에서 제품모델링 기술과 시스템 구현측면에서의 객체지향시스템 기술을 고찰하였다. 본 논문을 통해 제안한 표현방법론의 적합성을 검증하기 위해 선박구획배치 모델링을 프로토타입 모델링 대상으로 하여 객체지향 선박구획배치 프로그램("OO_COMDEF"라 함)을 작성하였고 이를 구획배치 모델링에 적용하여 보았다. 본 표현방법론은 조선 전용 CAD/CAM 시스템의 국내 개발을 위한 Framework의 첫단계로서 제시하고자 한다.

Abstract

In this paper, the trends for the standardization of the product model was surveyed, and the product model can be used as concrete means to realize the communication of information between the CAD/CAM softwares, which are used for the design and manufacturing of the complex products such as ships. We have proposed the newly developed methodology for the representation and definition of the ship model on a basis of the product model. And also we have studied the product modeling technology for the aspect of an application, and the object-oriented system technology have been surveyed for the system implementation issues.

발 표 : 1992년도 대한조선학회 춘계연구발표회('92. 4. 18.)
접수일자 : 1992년 5월 19일, 재접수일자 : 1992년 8월 21일
* 정회원, 해사기술연구소

We would like to verify the consistency and correctness of our proposed representaion methodology by using the prototype application model, which is applied to design work of ship compartmentation model. For this purpose we have developed the "OO_COMDEF" (which means the Object-Oriented System for the Compartmentation Definition of ship) program that is applied to the compartmentation model which can be considered as a submodel of the general ship models. The results of research work have been proved that the representation methodology for the ship model based on the product model is an efficient and appropriate scheme for the ship model definitions. Consequently, this methodology can be proposed as a fundamental framework for the development of the shipbuilding CAD/CAM system.

1. 서 론

조선 분야와 같은 복잡한 대상물을 다루는 CAD/CAM 시스템에서는 현재까지는 제품의 형상표현 중심이었으나, 효과적이고 실질적인 설계·생산을 위한 시스템 구축을 위해서는 제품의 설계 의도와 기능 및 형상 등의 설계정보와 생산전개를 위한 부재간의 공정순서, 생산일정, 설비, 인적자원등을 망라한 새로운 모델의 필요성이 요구되고, 이를 구체적으로 구현할 수 있는 기술로서 제품모델링 기술과 전산표현방법 기술이 개발되어야 한다. 또한 시스템 구현 측면에서는, 종래의 절차식 프로그래밍 방식에서는 프로시듀어 자체보다는 데이타 중심으로 구조화 되어졌었다. 예를들면 일반적인 언어를 이용할 때 서브루틴간에 데이타를 넘겨주고 받을 때 매개변수를 사용했었다.

이러한 종래의 소프트웨어 기술로는 복잡도가 높은 대상물의 설계·생산 전산 시스템 개발에는 적절치 못했다. 반면에 객체지향 기술은 실세계의 현상을 모델링하므로써 설계·생산업무에 내재한 복잡성을 다룰 수 있는 시스템 개발을 가능케 하므로 소프트웨어 생산성이 대폭 향상된 기술이다 [12,13].

이와같은 객체 지향적 시스템 기술에 의한 선박제품모델을 구축하려면 선박제품모델을 구성하는 서브모델의 정의와 이의 전산표현방법론이 개발되어야 한다.

본고에서는 제품모델의 표준화 동향과 이의 구체적인 사례인 ISO(International Standard Organization) 의 STEP(Standard for the Exchange of Product Data)에서 응용분야의 하나로 제시한 Ship Structural에 대한 표준화 동향을 고찰하였으며, 객체지향 패러다임을 기초로 하여 개발한 선박모델과 표현방법론을 제시함과 아울러, 선박의 구획배치모델링을 프로토타입 모델링 대상의 범위로 설정하여 제시된 방법론의 적합성을 검증해 보았다. 본고에서 제시된 표현방법론을 조선 전용 CAD/CAM 시스템의 국내 개발을 위한 Framework의 첫단계로서 제시하고자 한다.

2. 제품모델의 표준화 기술 동향

2.1 일반

오늘날 선박과 같은 공학적인 대상물을 설계하거나 생산하는데 사용되는 대부분의 소프트웨어 시스템들은 해당 대상물에 특정하게 국한되도록 개발되었다.

이러한 시스템들을 서로 상이한 조직에서 사용해온 경험에 비추어 볼 때, 이들 시스템들간의 상호 정보교환의 필요성이 명백하다. 필요한 정보교환을 달성하기 위해서 여러가지 개념들이 모색되고 있다. 예를 들면 조선분야의 상용 CAD/CAM 시스템들은 선박을 구성하는 상이한 양상(기본설계, 선체설계, 의장설계 등) 중에서 특정한 한 부분만을 다룰 수 있도록 개발되었으므로 이들 상이한 소프트웨어 시스템들간의 상호 정보교환이 잘 이루어지지 않고 있다. 이를 해결하기 위해 표준화에 관한 개발이 진행중이며, 상이한 소프트웨어 시스템들간에 사용이 가능한 형태의 제품정의 데이타의 전달을 위한 표준화에 초점이 맞춰지고 있으며, 이와 같은 표준화의 기초는 제품 모델이라고 할 수 있다.

즉, 제품의 설계·생산을 위한 소프트웨어 시스템들간의 상호 정보교환을 위한 정확한 기술 방법을 구체화한 것이 제품모델이라고 하며, 제품을 완전하

게 기술하려면 여러가지 종류의 정보가 필요하므로 이를 위한 서브모델 개념이 제시되고 있다.

제품모델을 기술하는 방법론으로 IGES(Initial Graphics Exchange Specification), STEP(Standard for the Exchange of Product Data)이 제시되었으며, 이중에서 ISO(International Standards Organization)에서 주관하는 STEP이 중요한 방법론으로 거론되고 있다. 1988년도에 STEP에서는 선박제품모델을 구성하는 하나의 서브모델로 간주할 수 있는 Ship Structural 분야를 제품모델의 응용분야 중의 하나로 채택하여 이에 대한 표현방법과 형식을 표준화하고자 시작하여 진행중에 있다[1,2,3].

2.2 ISO의 STEP에서의 표준화 동향

2.2.1 제품모델의 표현방법론 필요성

서로 상이한 CAD 시스템 간에 설계의도를 교환하는 문제는 1970년대 후반부터 폭넓게 이해되고 연구되었다. 설계의도의 교환에 관한 이해는 그래픽스 차원 이상의 것이지만 당시에 널리 사용된 IGES라는 것 때문에 상당히 혼돈되었다. IGES는 기계엔지니어링 데이타의 교환을 위한 Neutral 형식으로 개발되었다. 즉, 전통적으로 사용되던 기계설계 도면을 표현하는 데이타의 교환을 위한 것이다. 설계도면은 관련분야의 폭넓은 지식과 작도상의 표준에 의해서 이해되고 해석되어 설계단계로 이행되어지므로 단순히 컴퓨터에 의해 전달된 디지탈 정보만으로는 설계도면이 본래 내포한 설계의도등을 상실한 채 정보가 전달된다고 할 수 있다. 그러므로 도면은 엔지니어링을 위해서는 결정적이고 또한 도면을 시스템의 그래픽스 기능에 의해서 작성하므로써 그래픽스 표준에 대한 혼돈이 발생하였다.

IGES가 출현하고 이에따라 다른 형태의 Neutral 표준이 개발되어 SET(Industrial Automation-Data Exchange and Transfer Standard Specification : France), VDA-FS(독일의 DIN 표준) 등이 제시되었다. 또한 CAD 시스템 개발 회사들은 각자 고유의 방식을 제시하였는데, 이러한 종류로는 Autodesk 사의 AUTOCAD용 DXF 화일 형식, Intergraph사의 EDIF 등이 있다. 이들은 Specification의 애매성을 줄이기 위한 형식언어로 사용되어 데이타 포맷의 상호교환에 활용되었다.

이로부터 얻은 경험은 STEP 개발에도 많은 기여를 하고 있다. STEP은 1984년도부터 ISO의 주관하에 제품모델의 표현에 관한 표준으로 개발되고 있다. STEP 개발에 많은 기여를 한 EC의 CAD*I 프로젝트와 미국의 PDES(Product Data Exchange using STEP) 프로젝트가 STEP에 관련된 주요한 프로젝트이다.

STEP에서 목표로 하는 것은 어떠한 제품의 전 수명주기 동안에 해당제품을 완전하게 표현하고자 하는 것이며, 이는 제품모델의 구체적인 표현방법이 된다고 할 수 있다[4,5,6].

2.2.2 STEP의 구성

STEP의 개발을 위한 기본 방법은 다음과 같은 세가지 개념으로 요약할 수 있다.

- 제품모델을 참조모델로 사용
- 형식기술언어인 EXPRESS 언어 사용
- Physical Layer, Logical Layer, Application Layer 등의 3 Layer 구조

대상 산업분야에 대한 STEP에서의 표준개발은 Fig. 1과 같이 4개의 논리적인 구분을 하여 이를 STEP class라고 하며 이에 대한 설명은 다음과 같다.

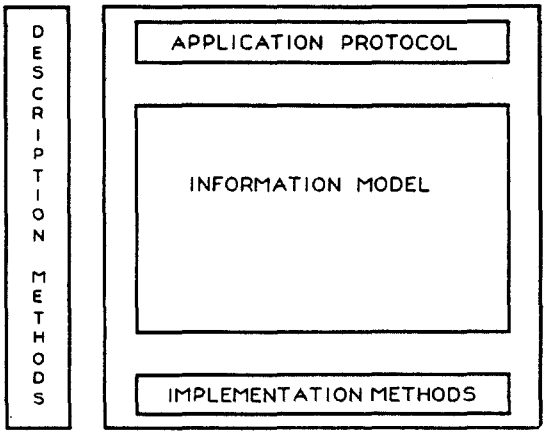

Fig.1 Configration of STEP classes

① Description Methods : STEP의 Entity를 기술하는 방식의 표준에 관한 부분으로 형식기술언어인 EXPRESS 언어를 제시한다. 이 언어는 제품모델의 Logical Layer를 기술하는데 사용되는 컴퓨터 언어로서, Schema, Type, Entity, Rule, Algorithm 등을 구성요소로 갖는다.

② Information Model : 제품모델을 이루는 핵심부분으로서, 일반적인 응용분야를 위한 모델과

특정분야의 응용에 관련된 Entity를 포함하는 모델로 구성된다. 이들 제품데이타는 Application Protocol을 통해서만 구현된다.

③ Application Protocol : Information Model이 갖고 있는 정보를 사용자가 응용분야별로 사용할 수 있는 기능을 제공하는 일종의 응용 프로세스로서, 대상분야의 응용에서 필요로 하는 정보를 명시적으로 기술한다.

④ Implementation Methods : STEP에 의해서 정의된 Information Model을 논리적으로 완전하게 지원할 수 있는 구현방식을 말한다.

이와같은 방법에 의해서 개발되고 있는 STEP의 표준과 STEP의 후보 Application Protocol은 Table 1, 2와 같다.

Table 1 The parts of STEP

Part number	(Preliminary) Title
Introductory class :	
1	Overview and fundamental principles
Description methods class :	
11	EXPRESS(including style rules)
Implementation forms class :	
21	Physical file
Conformance-testing methodology class :	Conformance-testing methodology and framework :
31	General concepts
32	Requirements on testing laboratories and clients for the conformance-assessment process
Information model : general resources :	
41	Generic Product Data Model(including miscellaneous resources)
42	Shape : (geometry, topology, design shape, solids)
43	Shape interface
44	Product structure and Configuration Management
45	Material
46	Presentation
47	Tolerances
48	Features
Information models : application resources :	
101	Draughting
102	Ship structures
103	Electrical applications
104	Finite-element analysis
105	Kinematics
Application Protocol Class :	
201	Draughting-related application protocol

Table 2 The candidate application protocol of STEP

Number	Title
1	Electrical distribution system, detailed design and production engineering
2	Exchange of 2D and 3D geometry and dimensioning
3	Exchange of 2D geometrically explicit CAD drawings with explicit annotation
4	Exchange of CAD drawings with reference to 3D geometry model and with explicit annotaion
5	Exchange of configuration-controlled 3D product-definition data
6	Exchange of sculptured-surface models
7	Factory workstation and machine-tool controller
8	HVAC, detailed design and production engineering
9	Libraries for distribution system
10	NC material-removal systems
11	Piping systems, detailed design and production engineering
12	Raceways, detailed design and production engineering
13	Road design
14	Sheet-metal parts
15	Ship structural systems, detailed design production engineering
16	Generative draughting exchange
17	Exchange of product-model data as a basis for inspection planning
18	Exchange of B-rep models

2.2.3 STEP의 Ship Structural Model

STEP Application Model중, Part 102인 Ship Structures 에 관한 표준화 작업을 1988년도에 미국의 NIDDESC(Navy Industry Digital Data Exchange Standards Committee)에서 1차로 초안을 작성하여 STEP에 제출하였다. 그러나 현재까지도 ISO 국제표준으로 확정되지는 않았고 진행중에 있다.

STEP Part 102의 목적은 선박의 Structures에 관한 CAD/CAM 시스템간의 데이타 교환을 위한 표준을 제정하기 위한 Information Model을 정의하는데 있다. 그 적용범위는 선박의 상세설계에서 Lofting 이 완료된 결과로 생성되는 Structural Product Model을 정의하는 레벨이라고 할 수 있다. 여기서는 다음과 같은 항목에 대한 Geometry, Topology, Property Data을 정의하고 있다[7].

- Lines
- Stiffened surfaces(shell, bulkheads, decks, web frames 등)
- Cutouts, lightening holes, penetrations
- Weld data, bevels
- Stiffener data(scantlings/traces/orientations/end cuts)
- Material definition(thickness, types, material 등)
- Brackets, collar plates

- Foundations
- Rudder

Ship Structural Model은 Application Information Model이므로 이의 실제적인 구현을 위해서는 STEP의 General Resource Information Model인 다음과 연계되어야 한다.

- Generic Product Model
- Shape(Geometry, Topology, Design Shape, Solids)
- Shape Interface
- Product structure and configuration management
- Material
- Presentations
- Tolerances
- Features

3. 선박모델의 표현방법론

3.1 제품모델을 기초로 한 선박 모델링

STEP의 Information Model은 시스템간의 데이타 교환을 위한 표준으로서의 역할을 하고 있지만, 우리는 이 Information Model을 시스템간의 데이타교환 이전에 CAD/CAM 시스템을 구성하는 핵심요소인 제품모델의 정의와 시스템간의 통합을 위한 가장 적절한 접근방법이라고 착안하여 이에따라 제품모델을 기초로 한 선박모델의 표현방법론을 개발하게 되었다.

STEP Part 102인 Ship Structure Model은 일반적인 선박제품모델의 하위 서브모델중의 하나로써 이해되어야 하며, 실제적인 시스템 적용을 하기 위해서는 선박 설계단계에서의 설계개념까지를 포함하는 선박제품모델이 개발되어야 한다. 이를 위해서는 우선 선박제품모델을 구성하는 서브모델을 정의하고, 이를 토대로 한 선박제품모델의 표현방법론이 정립되어야 한다[8].

선박제품모델을 구성하는 서브모델들의 정의를 위해서, 본고에서는 Bronsart[9]가 개념적으로만 제시한 선박제품모델의 정의를 참조하였다. 또한 STEP Part 102인 Ship Structural Model 및 STEP의 General Resource에 관한 Information Model중 Part 41-48과의 연계성을 모색하였다.

Fig.2는 Bronsart가 제시한 선박제품모델의 최상위 레벨을 보여주고 있으며, Table 3은 이를 STEP의

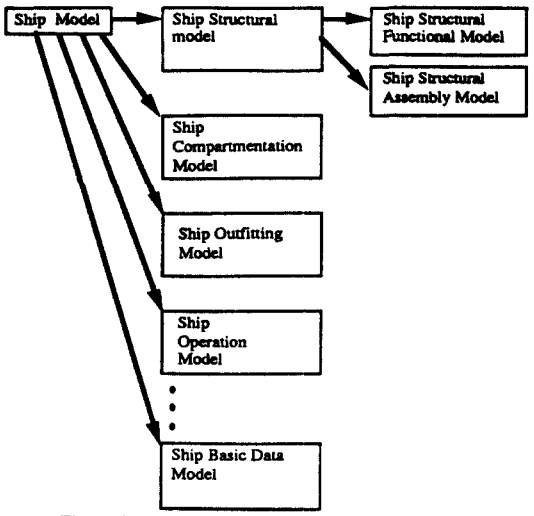

Fig.2 The ships model which is composed of its submodel

Table 3 The top-level ship model which is represented by EXPRESS language

```
SCHEMA      ipim_ship_model_schema;
ASSUME      (ipim_resource_schema, ipim_geometry_schema,
            ipim_topology_schema, ipim_material_schema)
ASSUME      (ship_basic_data_schema, ship_structure_schema,
            ship_compartmentation_schema, ship_assembly_schema,
            ship_outfitting_schema, ship_operation_schema)
ENTITY      ShipModel;
            BasicData;    ShipBasicDataModel;
            MadeUpOfSubSystem:    SETOFShipSubModel;
END_ENTITY
ENTITY      ShipBasicDataModel;
            ShipIdentification:    Name;
            PrincipalValues:       PrincipalValuesTable;
            GlobalReferenceSystem: ShipReferenceSystem;
Unique      ShipIdentification;
ENTITY      ShipSubModel;
            SUPERTYPE OF(ShipStructuralModel
                  XOR ShipCompartmentationModel
                  XOR ShipOutfittingModel
                  XOR ShipOperationModel
                  ("XOR…"));
END_ENTITY
END_SCHEMA
```

EXPRESS 언어로 표현한 것을 보여주고 있다.

본고에서는 그림(2)에 나와 있는 바와 같이 선박제품모델 전체를 취급하기에는 그 범위가 광범위하여, 일차적으로 선박제품모델을 구성하는 서브모델인 구획배치모델에 대하여 국한시켜, 선박모델의 표현방

법을 정립하였다.

표현방법으로서는 객체지향 패러다임을 적용하였다. 객체지향 패러다임이란 표현하고자 하는 모델을 구성하는 기본구성요소들을 도출하고, 이를 정의하기 위한 자료구조와 이들 기본구성요소가 수행할 메소드를 결정한 후, 이를 객체 클래스라고 하는 하나의 정보단위로 묶어 사용하는 것(encapsulation)을 말한다. 이러한 표현방법론이 정립되면, 곧바로 시스템 구현을 가능케 하는데, 이는 객체지향 시스템 개발 기술을 기초로 하고 있기 때문이다.

이상과 같이 객체지향 패러다임을 적용하여 정립된 구획배치모델의 표현방법론을 검증하기 위해 프로토타입으로 객체지향 구획배치 프로그램(이하 "OO_COMDEF"라 함)을 작성하여, 본 연구를 통해 제안된 표현방법론을 검증하였다[10].

3.2 OO_COMDEF의 객체모델 정의

3.2.1 객체모델링 기법

객체모델링 기법은 대상응용 분야를 모델링하기 위한 방법으로 현실세계의 문제범위를 객체(Object)와 그들간의 관계(Relation-ship)로 나타내며 세가지 각도에서 해당 시스템을 모델링하고 각각의 모델에서 중요한 양상들을 포착한다. 객체모델링 기법에서 다루는 세가지 모델에는 시스템의 정적이고 구조적인 데이타의 양상을 나타내는 객체모델(Object Model) 과 프로세스 중심의 콘트롤 양상을 보여주는 동적모델(Dynamic Model), 그리고 변화시켜줄 함수의 양상을 나타내는 함수모델(Functional Model)이 있다[12].

객체모델은 한 시스템 내의 객체들과 그 객체들간의 관계들 그리고 그 객체들을 표현해주는 클래스(Class)들의 특성을 결정지어주는 속성(Attribute)과 메소드(Method)를 포함한다. 여기서 메소드란 어떠한 클래스의 자료구조 부분을 다루는 일종의 프로세스라고 할 수 있다. 따라서 객체모델링 기법에서 객체모델은 세가지 모델중에서 가장 핵심적인 역할을 하므로 본고에서는 모델링 대상인 선박구획배치를 위한 객체모델을 제시하고자 한다.

객체모형도(Object Diagram)를 통해서 객체모델을 가시화 하며, 이 모형도는 객체, 클래스 그리고 이들간의 관계를 모델링하기 위해 필요한 형식화된 그래픽 기법으로 표시되며 이들 구성요소는 Fig.3 과 같다.

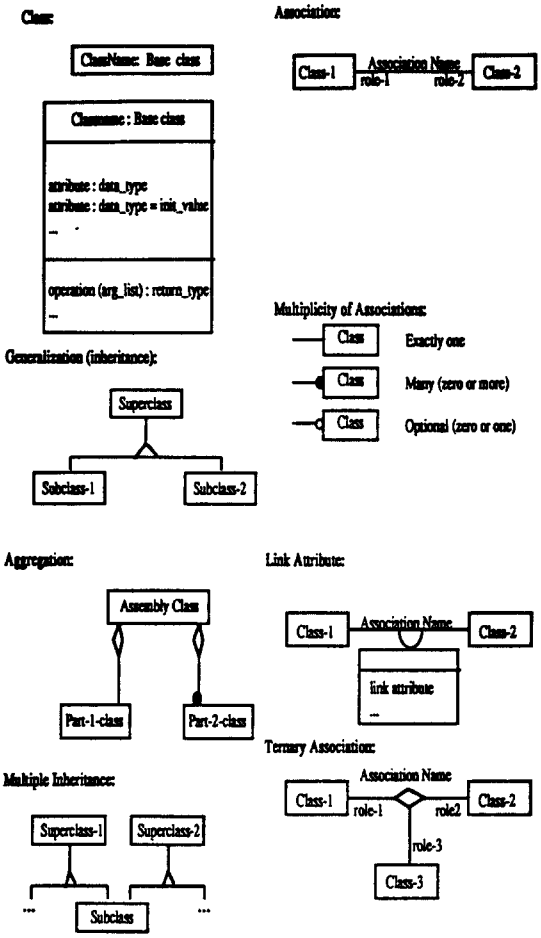

Fig.3 The components of object model diagram

3.2.2 객체모델의 정의

선박의 구획배치설계는 넓은 의미에서, 정의된 선체 형상내에 선체의 강도를 유지하고 화물을 안전하게 수송할 수 있는 능력을 제공하는 구조부재와 선박의 기능상 필요한 공간을 설정하기 위한 비구조부재들의 기하학적 형상 및 위치를 결정하는 설계과정이라 할 수 있다.

따라서, 구획배치 모델은 구조부재 및 비구조 부재들을 정의하는 부재요소(이하"Logical Element"라 함) 객체와 이들 객체들에 의해 형성되는 공간(이하 "Space"라 함) 객체 및 이들 각 객체들의 기하학적 형상을 정의하는 형상(이하 'Geometry'라 함) 객체들로 구성할 수가 있다. 즉, 선박은 이들 각 객체 요소들과 객체 요소들간의 상관 관계에 의해 표현된다고 볼 수 있다. 이로부터, 본 연구에서는 구획 배치

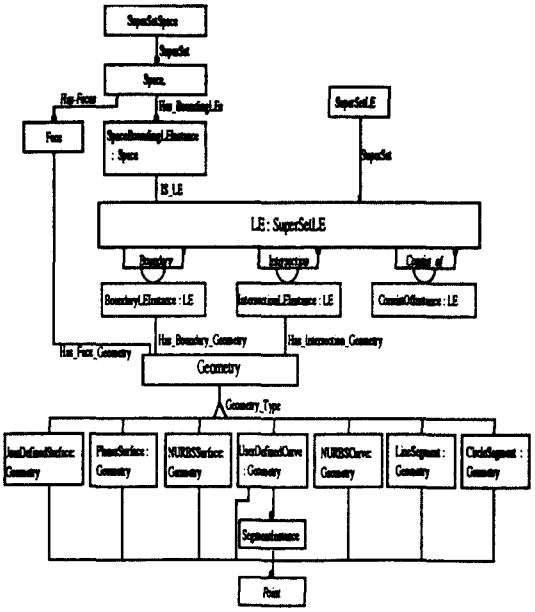

Fig.4 Object model

모델을 위한 기본 객체로서 Logical Element, Space 및 Geometry를 선정하였고, 이들 객체에 대한 객체 모형도는 Fig.4와 같다.

(1) Logical Element 객체

Logical Element는 선체의 강도 유지에 필요한 구조부재인 Ship Structure를 정의하기 위한 객체일 뿐만 아니라, 선박의 기능상 필요한 공간을 설정하기 위해 필요한 비구조부재, 즉 설계 개념이 포함된 Ship Compartmentation Model를 구축하기 위한 기본 객체이다.

Logical Element는 Table 4에 나와 있는 바와 같은 데이타 구조를 갖는데, 다음과 같이 서로 다른 Logical Element간의 상호 관련성(Topology Relationship) 을 표현할 수 있도록 설계하였다.

－Boundary, Bounded by 관계 :

하나의 Logical Element가 다른 Logical Element의 boundary를 제한하고 있음을 표현해 준다.

－Intersection 관계 :

하나의 Logical Element가 다른 Logical Element를 관통해서 지나가게 됨을 표현해 준다.

－Belongs To, Consist of 관계 :

Logical Element들간의 계승 관계(Inheritance)를 표현해 준다.

Table 4 Data structure of logical element object

Attribute 이름	의 미	비 고
LEID	해당 Logical Element의 이름	
LEGeometry	해당 Logical Element의 형상 정보가 정의되어있는 Geometry 객체를 가리키는 Pointer	
BelongsTo	Parent Logical Element 객체를 가리키는 Pointer	
Boundary	해당 Logical Element를 경계짓는 Logical Element 객체를 가리키는 Pointer	－연결리스트(linked list) 형태로 구성 －각 Boundary Logical Element에 의해 생성되는 경계면 형상은 Geometry 객체에 저장
BoundedLE	해당 Logical Element가 Boundary로 사용되는 Logical Element 객체를 가리키는 Pointer	
Intersections	해당 Logical Element와 Intersection 관계에 있는 Logical Element 객체를 가리키는 Pointer	－연결리스트 형태로 구성 －Intersection 관계에 의해 형성되는 교차면에 대한 형상은 Geometry 객체에 저장
ConsistOf	Child Logical Element 객체를 가리키는 Pointer	－연결리스트 형태로 구성
ResultGeometry	Boundary 관계에 의해 정의되는 형상이 표현되어 있는 Geometry 객체를 가리키는 Pointer	－연결리스트 형태로 구성

Table 5 Data structure of space object

Attribute 이름	의 미	비 고
SpaceID	해당 Space의 이름	
SpaceBoundingLEs	해당 Space를 구성하는 경계면으로 사용할 Logical Element를 가리키는 Pointer	－연결리스트로 구성
SpaceBoundingFaces	해당 Space를 구성하는 각 경계면의 형상이 정의되어 있는 Face 객체를 가리키는 Pointer	－연결리스트로 구성

(2) Space 객체

Space는 선박의 기능상 필요한 선체내의 폐위된 공간을 표현하기 위한 객체로서, 정의된 Logical Element 객체들을 이용하여 정의한다. Space는 Face라는 파생객체를 갖는데, 이 Face 객체는 Space를 구성하는 각 Face에 대한 형상이 정의되어 있는 Geometry 객체를 가리킨다. Space 객체의 자료구조는 Table 5와 같다.

(3) Geometry 객체

Geometry는 Logical Element 및 Space 객체들의 기하학적 형상을 정의하는 객체이다. Logical Ele-

Table 6 Data structure of geometry object

Attribute 이름	의 미	비 고
GeometryID	해당 Geometry의 이름	
GeometryType	기하학적 형상의 종류를 나타내는 것으로서 다음에 열거한 것중의 하나가 명시된다. -User Defined Surface -NURBS Surface -Planar Surface -User Defined Curve -NURBS Curve -Line Segment -Circle Segment -Point	-각 Geometry type은 Geometry 객체의 파생객체로서 선언되어 있으며, 각 파생 객체는 해당 Geometry Type을 정의하기 위해 필요한 데이타 구조를 갖는다.

ment 및 Space를 구성하는 각 Face는 경우에 따라, 여러가지 형태의 서로 다른 기하학적 형상으로 표현된다. 이를 감안하여,본 연구에서는 Geometry 객체에 대해 기하학적 형상의 종류에 따른 파생 객체를 정의하였고, 이들 각 경우에 해당하는 형상정보를 저장할 수 있도록 하였다. Geometry 객체의 데이타 구조는 Table 6에 나와 있는 바와 같다.

3.3 OO_COMDEF의 구현

일반적으로 객체지향 시스템의 개발은 해당 응용분야의 문제영역 전체를 분석하여 문제영역의 요소와 시스템 구현을 위한 프로그램 요소를 대응시키는 작업으로 볼 수 있으며, 추상화된 문제영역을 개념적인 객체모델로 표현하고 이를 객체지향 언어를 사용하여 구체적인 프로그램으로 변환하는 방법을 취하므로써 자연스러운 시스템 개발을 달성할 수 있다.

OO_COMDEF 프로그램의 실제적인 구현을 위해서는 객체모델에서 도출된 Logical Element, Space, Geometry 등의 객체들에 메소드를 추가하여 시스템 구현 요소인 클래스들을 구성하여야 한다. 또한 프로그램에서 생성된 객체들의 보관 및 검색을 하기 위해서는 데이타베이스 시스템을 통해 객체들의 데이타 및 메소드를 데이타베이스에 보관하고 필요시 저장된 객체들을 검색하여 활용하는 것이 필요하다.

3.3.1 클래스의 정의

OO_COMDEF 프로그램을 구현하기 위해서 본 연구에서는 객체지향언어인 C++을 선정하여 프로그램을 개발하였다. C++ 언어에서 클래스라고 부르는 사용자 정의자료형(User-defined Type)의 정의는 그 형의 객체를 표현하는데 필요한 데이타(멤버데이타 라고 함)와 그 객체를 조작하기 위한 메소드(멤버함수라고 함)를 포함하고 있다. 클래스들은 계층구조를 형성하면서 정의되므로 이 계층구조에 따라서 클래스들간의 계승(Inheritance)과 함수다형성(Polymorphism) 이 이루어져서 응용분야의 요구사항을 반영한다. OO_COMDEF 프로그램에서 사용하는 클래스의 계층구조는 · Fig.5와 같고, 이들 각각의 클래스가 갖는 멤버데이타와 멤버함수의 대표적인 예는 다음과 같다.

3.3.2 데이타베이스 설계 구현

선박구획배치 모델 구현시, 각 객체에서 정의된 데이타를 보관하고, OO_COMDEF 프로그램 구동시 필요에 따라 데이타베이스로부터 읽어들인 데이타를 in-memory 상의 해당 객체 데이타부분으로 초기화할 수 있도록 데이타베이스를 설계하고 이를 관계형 DBMS인 ORACLE로 구현하였다.

C++ 언어로 작성된 OO_COMDEF 프로그램을 통해서 선박구획배치 모델을 구현할 때, 생성된 객체들의 보존 및 검색이 문제점으로 인식되었다. 이는 객체지향 프로그램 언어가 갖고 있는 Persistent Object('Object Which is stored on disk') 개념의 결여 문제로서 객체지향 프로그램 언어의 공통적인 문제점이다. 이를 해결하기 위해서는 객체지향 데이타베이스 시스템을 통해 객체들의 데이타 및 메소드를 데이타베이스에 보관하고 필요시 저장된 객체들을 검색하여 활용하는 것이 바람직하다고 알려져 있으나, 현재까지 상용화된 객체지향 데이타베이스 시스템이 보편화되지 않은 상황이므로 본 연구에서는 ORACLE 관계형 DBMS를 이용해서 객체들의 데이타를 보관하고 검색할 수 있는 방법을 모색하였다.

이 방법은 객체지향 데이타베이스 시스템을 이용하는 것보다는 몇가지 제약이 있지만 기존에 저장된 관계형 데이타베이스의 데이타와의 연계가 용이하다는 잇점이 있다.

OO_COMDEF 프로그램에서 데이타베이스 시스템에 요구하는 기능은 크게 두가지가 있다. 첫번째는, OO_COMDEF 프로그램 구동시 클래스에 기반을 둔 객체들을 생성할 때에 이들의 초기치를 데이타베이스로 부터 검색하여 in-memory 상의 객체의 데이타부분을 초기화하는 것이고, 두번째로는 in-memory 상에서 정의된 객체들의 데이타 내용과

SuperSetSpace
AllSpace : SpaceList
FindSpace(SpaceID) Delete(SpaceID) DBStore(SpaceID)

Space : SuperSetSpace
SpaceID : String SpaceBoundingLEs : SpaceBoundaryList SpaceBoundingFaces : FaceList
NewSpace(SpaceID) SpaceBoundedBy(SpaceBoundingLE) : SpaceBoundaryList MakeSpace(SpaceID) SpaceBoundedBy(SpaceBoundingFace) : FaceList ChangeSpaceBoundingLE(SpaceBoundingLE, SpaceBoundingLE) : SpaceBoundaryList Draw(SpaceID)

SuperSetLE
AllLE : LEList
FindBoundary(LEID) FindBounded(LEID) FindIntersection(LEID) FindIntersected(LEID) FineConsistOf(LEID) FindBelongsTo(LEID) Delete(LEID) DBStore(LEID)

LE : SuperSetLE
LEID : String LEGeometry : Geometry ResultGeometry : Geometry BoundingLEs : BoundaryList IntersectingLEs : IntersectionList ConsistOfLEs : ConsistOfList
NewLE(LEID) : LEGeometry ChangeLEGeometry(LEID) : LEGeometry DrawMe(LEID) DrawBoundary(LEID) BoundedBy(LEID) : BoundaryList IntersectedBy(LEID) : IntersectionList ConsistOf(LEID) : ConsistOfList GenerateRG( ) : Geometry RemoveBoundaryLE(LEID) : Boundary List RemoveIntersectionLE(LEID) : IntersectionList RemoveConsistOfLE(LEID) : ConsistOfList

Geometry
GeometryID : String GeometryType : Enumeration(PlanarSurface, NURBSSurface, UserdefinedSurface, NURBSCurve, UserdefinedCurve, LineSegment, CircleSegment)
NewGeometry(GeometryID, GeometryType)

UserdefinedCurve : Geometry
MNBox : Box StartPoint : Point EndPoint : Point Coef : UserdefinedCurveCoef InterpolationMethod : String DefinitionPoint : PointList ConsistofSegment : SegmentList
NewGeometry(Coef, InterpolationMethod, DefinitionPoint) : SegmentList ChangeGeometry( )

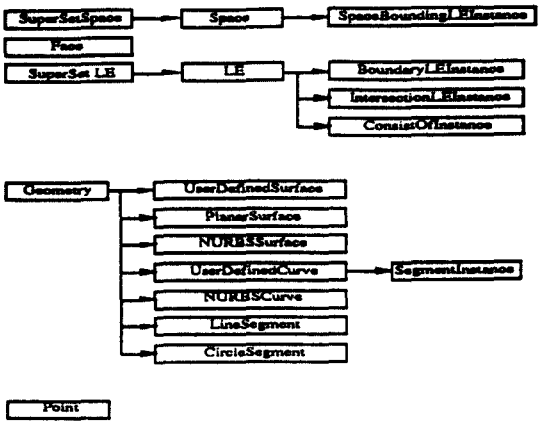

Fig.5 Class hierarchy of OO_COMDEF program

구조를 데이타베이스에 보존하는 기능이다.

이 두가지 기능을 구현하기 위해서는 데이타베이스 액세스만을 담당하는 별도의 클래스를 정의하고 이 클래스의 멤버함수로서 앞에서 요구된 두가지 기능을 실행토록 한다. 즉 데이타베이스로부터 객체들의 데이타 구조를 검색하여 OO_COMDEF 프로그램에서 다루는 in-memory상의 객체의 데이타 부분을 초기화시키는 함수와 정의된 객체들의 데이타 구조를 데이타베이스에 삽입하는 함수를 정의하여야 한다.

이 두가지 함수를 이용하여 데이타베이스 내의 데이타구조를 OO_COMDEF 프로그램의 in-memory 객체 구조로 사상(mapping)시킨후에 OO_COMDEF 프로그램이 구동되어 선박구획배치 모델링이 수행되며, 작업 완료후에는 정의된 in-memory 객체 구조를 데이타베이스 내의 데이타 구조로 사상시켜 보관하므로써 객체들의 데이타구조를 데이타베이스내에 보관할 수 있다.

본 연구에서는 설계코자 하는 선박구획배치 데이타베이스를 ORACLE 관계형 DBMS에서 사용 가능

한 구조로 설계하기 위해서, 관계형 데이타베이스의 논리적 구조인 릴레이션들의 스키마를 설계하였다. 관계형 데이타베이스를 설계하기 위한 상향식 접근 방법은 복잡도가 증가되면 실제로 데이타베이스 설계에 적용하기에는 어려움이 있다. 따라서 본 연구에서는 CSDP－데이타베이스 관리시스템 [14]에서 제안된 확장형 ER(Entity-Relationship) 모형을 이용하여 관계형 데이타베이스의 논리적 설계방법론에 따라 상향식과 하향식을 혼합한 방법으로 선박구획배치 데이타베이스를 설계하였다.

선박구획배치 모델에서 정의된 각 클래스들의 멤버데이타를 엔티티와의 관계로 표현하여 Fig.6과 같은 확장형 ER모형도를 작성하였으며, 이 모형도로 표현된 개념적인 데이타베이스 구조로부터 논리적 구조인 후보 릴레이션 스키마를 도출하였다.

후보 릴레이션들의 스키마로부터 각 데이타 항목들간의 함수적 종속관계를 고려하여 최종적인 릴레이션 스키마를 도출하였다. 설계된 데이타베이스의 최종적인 릴레이션 스키마는 Table 7과 같다.

Table 7 Relation Schema

Table Name	Attribute Name
SpaceConsistOf	SpaceID, FaceID
SpaceBoundary	SpaceID, BoundaryLE
Face	FaceID, BasisGeometry, XU, XL, YU, YL, ZU, ZL
FaceBoundary	FaceID, BasisBoundary & IntersectionCurveGeometry,
Geometry	Order, StartPoint, EndPoint
LogicalElement	LEID, LEGeometry, ResultGeometry
LEBoundary	BoundedLE, BoundingLE, BoundaryCurveGeometry
LEIntersection	IntersectedLE, IntersectingLE, IntersectionCurveGeometry
LEConsistOf	ConsistOfLE, BelongsToLE
BoundedSurface	ResultGeometryID, BasisBoundaryCurveGeometry,
	StartPoint, EndPoint
Geometry	GeometryID, GeometryType, XU, XL, YU, YL, ZU, ZL
PlanarSurface	PlanarSurfaceID, CoefA, CoefB, CoefC, CoefD
UserdefinedSurface	UserdefinedSurfaceID, UserdefinedCurveID, Order
UserdefinedCurve	UserdefinedCurveID, InterpolationMethod, StartPoint,
	EndPoint
UCDefinitionPoint	UserdefinedCurveID, Point, Order
Segment	SegementID, SegmentType, CoefA, CoefB, CoefC, Start Point,
	EndPoint
UCConsistOf	UserdefinedCurveID, SegmentID, Order
LineSegment	LineSegmentID, StartPoint, EndPoint, CoefA, CoefB, CoefC
CircleSegment	CircleSegmentID, StartPoint, EndPoint,
	CoefA, CoefB, CoefC

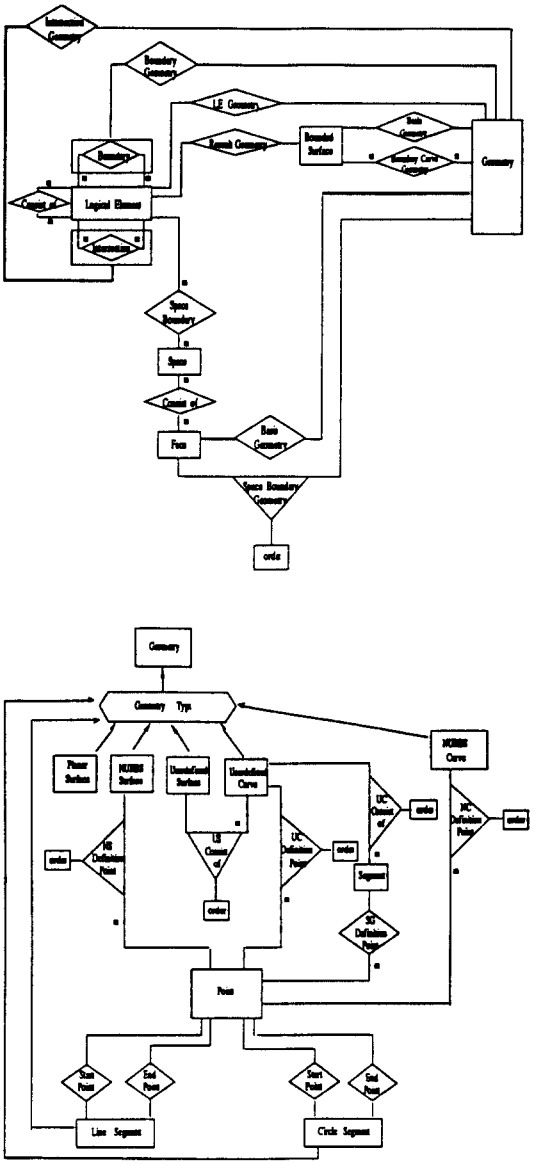

Fig.6 Extended entity relationship diagram

3.3.3 표현방법론의 적용 예

선박구획배치 모델링을 하기 위해서 본고에서 제시한 객체의 표현방법론에 따라 C++언어로 프로그램을 작성하였고, 주요한 클래스인 Logical Element 클래스를 정의한 C++ Header file의 예는 다음과 같다.

```
class LE;
class Boundary{
        LE*                     BoundLe;
}
class BoundaryList {
        Boundary*               BoundLe;
        BoundaryList*           prev;
        BoundaryList*           next;
};
class Intersection {
        LE*                     Intle;
};

class IntersectionList {
        Intersection*           IntersectionLe;
        IntersectionList*       prev;
        IntersectionList*       next;
};
class ConsistOf {
        LE*                     ConsistOfLe;
};
class ConsistOfList {
        ConsistOf*              ConsistOfLe;
        ConsistOfList*          prev;
        ConsistOfList*          next;
};
class LE      :public SuperLe {
        char*                   LEID;
        Geometry*               LEGeometry;
        Geometry*               ResultGeometry;
        BoundaryList*           BoundingLEs;
        IntersectionList*       IntersectionLEs;
        ConsistOfList*          ConsistOfLEs;
        public:
        void                    NewLE(char* LEName, char* LEType);
        void                    ChangeLEGeometry (LE* LEID);
        void                    DrawMe (LE* LEID);
        void                    DrawBoundary (LE* LEID);
        BoundaryList*           BoundedBy (LE* LEID);
        IntersectionList*       IntersectedBy (LE* LEID);
        ConsistOfList*          ConsistOf (LE* LEID);
        BoundaryList*           RemoveBoundaryLE (LE* LEID);
        IntersectionList*       RemoveIntersectionLE (LE* LEID);
        ConsistOfList*          RemoveConsistOfLE (LE* LEID);
};
LE: :NewLE(char* LEname, char* LEtype) {
        BoundaryList*           b;
        char*                   G;
        strcpy (G = malloc(3), "G_");
        switch(*LEtype) {
        case 'p' :      LEGeometry = new (PlanarSurface);
                        LEGeometry -> geometryID   = strcat(G, LEname);
                        LEGeometry -> geometryType = LEtype;
                        break;
        case 'l':       LEGeometry = new (LineSegment);
                        LEGeometry -> geometryID   = strcat(G, LEname);
                        LEGeometry -> geometryType = LEtype;
                        break;
        default:        cout << " Unknown geometry type !!!\n";
                        break;
};
        LEID = LEname;
        b = new (BoundaryList);
        b -> BoundLe= NULL;
        b -> prev     = NULL;
        b -> next     = NULL;
        BoundingLEs = b;
};
LE: :BoundedBy (LE* c) {
        Boundary* f;
        BoundaryList* g;
        f = new (Boundary);
        g = new (BoundaryList);
        f ->BoundLe = c;
        BoundingLEs -> BoundLe = f;
        g -> next = NULL;
        g -> BoundLe = NULL;
        g -> prev = BoundingLEs;
        BoundingLEs -> next = g;
        BoundingLEs = g;
        return BoundingLEs;
};
LE: : RemoveBoundaryLE (LE* rmle) {
      char* name;
      name = rmle -> LEID;
while (BoundingLEs ->prev ->BoundLe ->BoundLe ->LEID ! = name)
      BoundingLEs = BoundingLEs ->prev;
BoundingLEs ->prev ->BoundLe = BoundingLEs ->BoundLe;
BoundingLEs ->prev ->next     = BoundingLEs -> next;
if (BoundingLEs ->next ! = NULL) {
      BoundingLEs ->next ->prev     = BoundingLEs -> prev;
      delete BoundingLEs;
      }
else          BoundingLEs = BoundingLEs ->prev;
while (BoundingLEs ->next ! = NULL)
          BoundingLEs = BoundingLEs ->next;
return BoundingLEs;
};
```

Fig.7 과 같은 선박구획배치 모델중 일부분에 대해서 사용자가 설계하고자 하는 Logical Element 객체와 Space 객체를 개념적으로 기술하면 Table 8과 같다.

Table 8에서 Object(LE, Space) 항목은 설계대상이 되는 객체이고 이 객체의 Basis Geometry와 다른 Logical Element들과의 관계는 Geometry 항목과 Operator 항목에 나타나며, 이들 관계를 지워주는 Operator에 관여하는 객체는 Operand Object 항목에 기술된다.

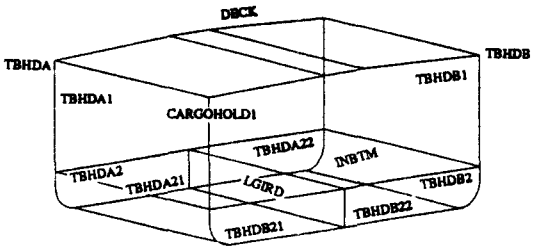

Fig.7 Example of ship compartmen tation model

Table 8 Conceptual definition of object

Object (LE,Space)	Geometry	Operator	Operand Object
SHELL	UserDefinedSurface	-	-
DECK	PlanarSurface	BoundedBy	SHELL
TBHDA	PlanarSurface	BoundedBy	SHELL, DECK
TBHDB	PlanarSurface	BoundedBy	SHELL, DECK
INBTM	PlanarSurface	BoundedBy	SHELL
		Intersect	TBHDA, TBHDB
LGIRD	PlanarSurface	BoundedBy	SHELL, INBTM
		Intersect	TBHDA, TBHDB
TBHDA1	PlanarSurface	BelongsTo	TBHDA
		BoundedBy	INBTM
TBHDA21	PlanarSurface	BelongsTo	TBHDA2
		BoundedBy	LGIRD
CARGO HOLD1	BoundedSpace	BoundedBy	SHELL, DECK, TBHDA1, TBHDB1, INBTM

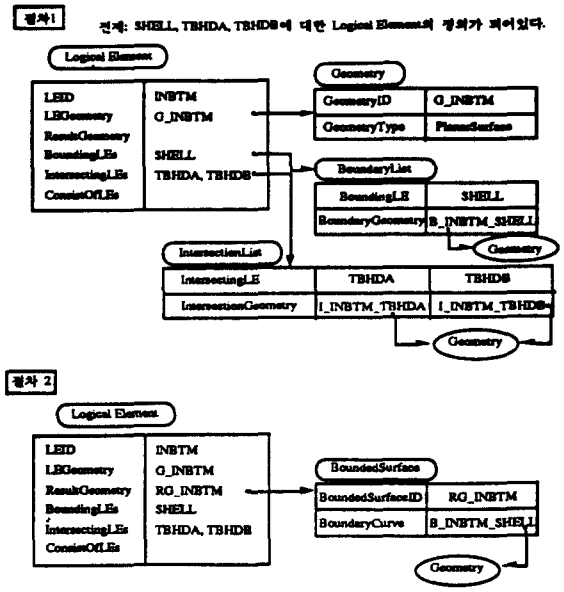

Fig.8 Data structure of logical element class

Table 9 Internal procedure of member-functions for logical element object definition

(1) NewLE("INBTM", "PlanarSurface", P1, P2, P3)
(2) INBTM. BoundedBy("SHELL")
(3) INBTM. IntersectedBy("TBHDA")
(4) INBTM. IntersectedBy("TBHDB")
(5) INBTM. GenerateRG( )
(6) INBTM. DBStore( )

이와같은 모델링 절차를 OO_COMDEF 프로그램에서 구현하는 내부적인 처리방법을 살펴보면, 우선 해당 클래스(예, Logical Element)의 클래스 인스턴스(예, Deck, Shell 등)가 in-memory 상에 자동적으로 생성되며, 다음으로는 해당 클래스의 멤버함수가 작동함에 따라 멤버 데이타들의 자료구조가 형성된다.

본 적용에서 Logical Element인 INBTM("Inner Bottom") 객체에 대한 프로그램의 내부적인 멤버데이타의 자료구조는 Fig.8과 같이 정의되며, 이들 멤버 데이타를 생성하고 처리하는 멤버함수의 작동절차는 Table 9와 같다.

4. 결 론

본고에서 제시한 선박 모델의 표현방법론은 응용측면에서의 제품 모델링 기술과 시스템 구현 측면에서의 객체지향 시스템 기술을 토대로 개발되었고, 따라서 이들에 대한 충분한 이해가 전제되어야 한다.

선박의 제품 모델링이란 선박 설계 생산 전단계에서 요구되는 데이타와 해당 응용 프로세스를 하나의 제품 모델이라는 틀 내에서 단일 모델로 성숙시키는 접근방식을 말한다[11].

즉, 선박의 제품모델을 구성하는 데이타 요소중의 하나를 살펴보면, 초기설계 단계에서 설계의도가 반영된 Logical Element(LE)라는 객체를 정의하고 설계 단계가 진행됨에 따라 상세 설계 정보인 부재 특성등이 추가된 Structural Element(SE)라는 객체가 생성되는데, 이 객체는 앞단계에서 정의된 LE로부터 속성을 계승받는다. 생산일정과 생산전개를 위한 정보등은 생산 설계단계에서 Manufacturing Element (ME)라는 객체에 포함된다. 즉 ME는 SE와 LE의 속성 및 형상정보를 자연스럽게 계승받는다. 이와같이 제품 모델링이란 단일 모델내에서 LE-〉SE-〉ME와 같은 공통의 정보를 갖는 객체가 성숙화되는 과정이라 할 수 있다.

응용분야의 개념적이고 추상화된 요구사항을 충분히 반영시킨 제품모델링을 실제적으로 전산기 내에서 구현하려면 데이타와 응용 프로세스가 하나의 정보단위인 객체로 취급되는 객체지향 패러다임을 적용하므로써 시스템 구현이 가능하다.

본 논문에서 제안한 선박모델의 표현방법론을 선박구획배치 모델링에 적용하여 검증한 결과, 설계자의 선박설계 개념과 복잡성을 내포한 자료구조를 전산화된 코드로 표현할 수 있는 효율적인 접근방식임을 알 수 있었다.

본 연구를 통해 제안된 표현방법론을 CSDP 연구사업의 선박제품 모델 관련 기술개발을 위한 Framework 의 첫단계로서 제시하고자 하며, 보다 실질적인 시스템 구축을 위해 다음과 같은 내용들을 계속적으로 연구 개발하여 시스템을 확장하고자 한다.

- 선박설계 단계별로 관련된 객체 클래스의 추가 및 확장
- STEP Part 102(Ship Structural Model), Part 41~48(표1 참조)과 연결된 Application Protocol 개발
- GUI(Graphic User Interface)를 이용한 사용자 환경 개발
- OODB(Object Oriented Database)의 적용

－기존 조선용 CAD/CAM 시스템과의 연계방안 모색

참 고 문 헌

[1] Wilson P R, "A short history of CAD data transfer standards", IEEE Comp. Graphics & Application Vol.7, No.6, pp.64－67, 1987.

[2] Wilson P R, "PDES STEP forward", IEEE Comp. Graphics & Application Vol.9, No.2, pp.79－80, 1989.

[3] Bloor M S, Owen J, "CAD/CAM product data exchange : the next step", CAD, vol.23, No. 4, pp.237－243, 1991.

[4] STEP, "Part 1 : Overview and fundamental principles", ISO TC184/SC4/WG PMAG Document N 43, 1991.

[5] STEP, "EXPRESS language reference manual", ISO TC184/SC4/WG5 Document N 14, 1991.

[6] STEP, "Part 42 : Integrated resources : geometric and topological representation", ISO TC 184 /SC4/WG3 Document N 45, 1991.

[7] STEP, "Part 102 : Ship structures", ISO TC 184/SC4/WG1 Document N 411, 1989.

[8] Bronsart R, Lehmann E, "A data model for ship steel structure", ICCAS, 1988.

[9] Bronsart R, "Design and management of a product model", Schiffstechnik, 1990.

[10] 이규열, 김용철, 강원수, "A Computer-based Compartment Layout Design System Using Entity-Relationship Data Model", The 4th International Marine Systems Design Conference, Kobe, Japan, 1991.

[11] 이규열, 서승완, 강원수, "CSDP(Ⅲ)－종합시스템 개발", 해사기술연구소 연구보고서, 1991.

[12] Rumbaugh James, Blaha Michel, "Object-oriented modeling and design", Prentice hall inc., 1991.

[13] Stroustrup, "The C++ programming language", Addison-Wesly, 1986.

[14] 신동우, 이규옥, 박노상, "CSDP(Ⅲ)－데이타베이스 시스템 개발", 해사기술연구소 연구보고서, 1991.