論 文

大 韓 造 船 學 會 論 文 集
제 30 卷 第 3 號 1993年 8月
Transactions of the Society of
Naval Architects of Korea
Vol. 30, No. 3, August 1993

객체 지향 선박 구획 정의 표현 방법론

강원수*, 서승완*, 이규열*

The Representation Methodology for Object-Oriented Ship Compartmentation

by

W.S. Kang*, S.W. Suh* and K.Y. Lee*

요 약

최근 컴퓨터로 실세계를 모델링하기 위한 소프트웨어 개발 기술인 객체 지향 패러다임은 복잡한 모델과 시뮬레이션 시스템등을 처리하는데 신속하고 생산성 높은 프로그래밍 기법으로 알려져 있다. 본 논문에서는 선박의 구획 배치 설계과정을 모델링 대상의 범위로 설정하여 객체 지향 개념을 이용한 모델링 방법에 관해 살펴보았다. 아울러 제안된 객체지향 선박구획배치 모델링 방법을 토대로 객체지향 언어인 C++를 이용하여 개발한 선박구획배치 프로그램(OO-COMDEF) Version 1.0을 소개하고, 개발된 프로그램을 이용한 적용 예를 보였다.

Abstract

The object-oriented paradigm is being recognized as the most effective technique to develope the complex, large software system.

In this paper, we propose the modeling methodology using object-oriented technology for the ship compartmentation design(OO-COMDEF) based on the proposed modeling methodology.

Illustrative examples to show the application of the prototype system are given.

발 표 : 1992년도 대한조선학회 추계연구발표회('92. 11. 14.)
접수일자 : 1993년 2월 9일, 재접수일자 : 1993년 5월 3일
* 정회원, 한국기계연구원 선박·해양공학연구센터 CSDP 사업단

1. 서 론

지금까지 사용되어 오고 있는 전통적인 소프트웨어 개발 기술들은 실세계를 모델링 하기 위해서 문제 범위를 함수의 집합으로 대응시키고 각 함수에서 필요로 하는 자료 구조를 정의한 후, 입력 데이타를 출력 데이타로 변환시키기 위한 함수의 내부 기능을 기술하는 기법들이다. 즉, 데이타 흐름 방식이나 수학적 논리를 기초로 한 프로그래밍 방식이다. 그러나 이러한 전통적인 소프트웨어 개발 기술로서는 복잡도가 높은 대상물의 시스템 개발에 상당한 어려움이 뒤따르게 된다.

이에 반해, 최근 도입된 객체 지향 시스템 기술은 컴퓨터로 실세계를 모델링 하기에 가장 좋은 소프트웨어 개발 기술로서 받아들여 지고 있는데, 이것은 소프트웨어 개발의 새로운 패러다임으로서 복잡한 모델과 시뮬레이션 시스템 등을 처리하는데 신속하고 생산성 높은 프로그래밍 기법으로 알려져 있다. 즉, 함수들과 함수의 처리 대상이 되는 데이타를 객체라는 하나의 정보 단위로 사용하여 이 객체들의 집합으로서 해당 문제를 모델링 하고자 하는 새로운 프로그래밍 기법이다.

본 논문에서는 선박의 구획 배치 설계 과정을 모델링 대상 범위로 설정하여 객체지향 개념을 이용한 모델링 방법에 관해 살펴보았다. 아울러 제안된 객체 지향 선박 구획배치 모델링 방법을 토대로 객체 지향 언어인 C++를 이용하여 개발한 선박 구획 배치 프로그램(OO-COMDEF) Version 1.0을 소개하고, 개발된 프로그램을 이용한 적용 예를 보였다.

2. 선박 구획배치 모델의 객체 지향 표현 방법론

2.1 일 반

최근의 소프트웨어 개발 기술들은 새로운 방향으로 발전하고 있다. 기존의 전통적인 소프트웨어 기술은 함수적인 속성을 위주로 하여 입출력 데이타 구조와 이를 처리하는 함수들로서 주어진 문제 범위를 해결하고자 하는 방식이었다. 그런데 최근에 대두된 객체 지향 개념을 토대로한 각종 컴퓨터 기술들은 현실세계 및 문제범위에 대한 표현 방식 및 모델링 방식이 전혀 다른 새로운 기술이라고 할 수 있다. 이러한 기술들은 소프트웨어 개발 뿐만 아니라 데이타베이스 시스템, 프로그래밍 언어들을 포함한 전반적인 컴퓨터 분야와 응용 분야에 적용되고 있는 추세이다.

이러한 추세에 발맞추어 본 연구에서는 객체 지향 개념을 통한 프로그래밍 기법 및 현실 세계의 문제 범위를 객체 지향 개념을 통해 모델링 및 설계 구현하는 방법을 살펴 보았으며, 이 방법론을 선박 구획 배치 설계 문제에 적용하여 선박 구획 배치 모델의 객체 지향적 표현 방법론을 정립하였다.

2.2 객체 모델링 기법

객체 지향 패러다임을 이용한 시스템 개발을 위해 본 연구에서는 객체 모델링 기법을 이용하였다[10]. 객체 모델링 기법은 현실 세계의 문제 범위를 객체와 그들간의 관계로 나타내는데, 이때 세 가지 측면에서 해당 시스템을 각각 모델링한다. 객체 모델링 기법에서 다루는 세 가지 모델은 다음과 같다.

(1) 객체 모델(Object Model)

객체 모델은 한 시스템 내에 존재하는 객체(object)들과 그 객체들 간의 관계(relationship)들 그리고 그 객체들을 표현해 주는 클래스(class)들의 특성을 결정지워 주는 속성(attribute) 및 속성과 관련된 연산(method) 종류들을 보여주므로서 해당 시스템의 정적 구조를 알 수 있도록 해 주는 모델이다. 객체 모델에서는 해당 시스템의 함수적인 특성 보다는 그 시스템을 구성하고 있는 객체들에 관심을 갖고 모델링을 한다.

(2) 동적 모델(Dynamic Model)

시간의 흐름에 따른 객체들 및 그들 객체들 간의 관계 변화를 알 수 있도록 해주는 모델로서 프로세스 중심의 콘트롤 양상을 보여준다. 즉, 외부 명령(message)을 받았을 때 그 명령을 수행하기 위해 필요한 연산들의 순서를 기술한 모델이다. 이때 수행될 연산의 기능 및 구현 방법 등에 대해서는 고려하지 않는다.

(3) 기능 모델(Functional Model)

기능 모델은 외부 명령에 대해 수행될 해당 연산과 그 연산에 필요한 인자(argument) 및 계산 결과를 보여주는 모델로서 데이타 흐름 모형도(data flow diagram)로 구성된다. 여기서, 연산의 실행이 언제 발생되는지 혹은 어떤 방법으로 구동되는지는 고려하지 않는다.

이상에서 설명한 객체 모델링 기법에서 다루는 세 가지 모델 중 객체 모델은 한 시스템 내에 존재하는

객체들과 그 객체들 간의 관계들을 보여주는 것으로서 시스템을 모델링하는 측면에서 가장 핵심적인 부분이다. 따라서 본 연구에서는 모델링 대상인 선박 구획 배치를 위한 객체 모델을 제시한다.

2.3 객체 모델의 정의

객체 모델링을 위해서는 모델링 대상이 되는 문제 영역내에 포함되어 있는 객체들 및 그 객체들간의 관계를 도출해야 한다. 따라서 선박 구획 배치 설계 과정을 분석하여, 구획 배치를 위해 필요한 정보를 파악한 후[11, 12], 이를 토대로 객체를 정의하였다. 또한 이들 객체들간의 관계를 설정해 줌으로서 선박 구획 배치 문제에 대한 객체 모델링을 수행하였다.

선박의 구획 배치 설계는 넓은 의미에서, 정의된 선체 형상내에 선체의 강도를 유지하고 화물을 안전하게 수송할 수 있는 능력을 제공하는 구조 부재와 선박의 기능상 필요한 공간을 설정하기 위한 비구조 부재들의 기하학적 형상 및 위치를 결정하는 설계 과정이라 할 수 있다. 따라서, 구획 배치 모델은 구조 부재 및 비구조 부재들 그리고 이들 부재들에 의해 정의되는 공간 등의 특성 및 형상으로 표현됨을 알수가 있다.

이로부터, 구획배치 모델을 위해 필요한 객체로 생각할 수 있는 것으로서는, 구조부재 및 비구조부재들을 정의하는 부재요소 객체(이하 "Logical Element"라 함), 이들 객체를 경계면으로 하여 형성되는 공간 객체(이하 "Space"라 함) 및 이들 각 객체들의 기하학적 형상을 정의하는 형상 객체(이하 "Geometry"라 함)이다. 즉, 선박 구획배치는 이들 각 객체 요소들 및 객체 요소들 간의 상관 관계에 의해 표현된다고 생각할 수 있다. 이상으로 부터, 구획 배치 모델을 위한 기본 객체로서 Logical Element, Space 및 Geometry를 정의하였으며, 이들 각 객체에 대한 자세한 내용은 다음과 같다.

(1) Logical Element 객체

Logical Element 객체는 선체의 강도 유지에 필요한 구조 부재와 선박의 기능상 필요한 공간을 설정하기 위해 필요한 비구조부재를 정의하기 위한 기본 객체로서 Table 1에 나타나 있는 바와 같은 데이타 구조를 갖도록 정의하였다. 형상 관련 정보는 Geometry 객체를 통해 정의되어 있고, 다음과 같은 Logical Element 간의 상호 관련성(Topology Relationship)을 표현할 수 있도록 데이타 구조를 정의하였다.

- Boundary 관계:
  하나의 Logical Element가 다른 Logical Element의 경계를 제한하고 있다는 관계를 표현한다. 예를 들어, 임의의 횡격벽을 정의할 때, 횡격벽의 기본 형상을 평면상에 존재하는 무한평판으로 정의한 후, 이 무한평판을 상갑판면과 선체를 경계면으로 제한함으로서 횡격벽의 실제 형상을 구할 수 있다. 이때, 상갑판면과 선체는 횡격벽의 Boundary Logical Element라고 하며, 이들은 서로 Boundary 관계를 갖고 있다고 말한다.

- Intersection 관계:
  하나의 Logical Element가 다른 Logical El-

Table 1 Data structure of logical element object

Attribute 이름	의 미	비 고
lEName	해당 Logical Element의 이름	
lEGeometry	해당 Logical Element의 형상 정보가 정의되어 있는 Geometry객체의 이름	
thisLEBoundaryList	해당 Logical Element의 경계면을 결정짓는 Logical Element 객체들의 이름	-연결 리스트(Linked List) 형태로 구성
thisLEIntersection List	해당 Logical Element와 Intersection 관계에 있는 Logical Element 객체들의 이름	-연결 리스트 형태로 구성
ParentLE	해당 Logical Element를 생성시킨 Logical Element 이름	-예를들어, TBHD-001이라는Logical Element를 두 부분으로 나누었을 경우 그 각각의 Logical Element는 TBHD-001의 Logical Element를 parentLE로 갖게 된다

Table 2 Data structure of space object

Attribute 이름	의 미	비 고
spaceName	해당 Space의 이름	
spaceFactor	용적 계산을 위한 Factor	
density	해수 혹은 담수의 밀도	
volume	해당 Space의 용적	
lcb	해당 Space의 길이 방향 중심	
tcb	해당 Space의 폭 방향 중심	
kb	해당 Space의 높이 방향 중심	
thisSpaceFaceList	해당 Space의 경계면으로 사용되는 Logical Element 들의 이름	-연결 리스트로 구성

ement를 관통해서 지나가는 관계를 의미한다.

- Parent 관계:
  Logical Element 들 간의 종속관계를 표현한다.

(2) Space 객체

Space 객체는 선박의 기능상 필요한 선체내의 폐위된 공간을 표현하기 위한 객체로서, 정의된 Logical Element객체들을 경계면으로 하여 정의된다. Space 객체의 자료구조는 Table 2와 같다.

(3) Geometry 객체

Geometry 객체는 Logical Element 및 Space 객체들의 기하학적 형상을 정의하기 위한 객체이다. Logical Element는 경우에 따라 서로 다른 기하학적 형상을 갖는다. 따라서, Geometry 객체에 대해 Sculptured Surface와 Planar Surface라는 두 개의 파생 객체를 정의하였다. Sculptured Surface 객체는 선체 형상과 같은 3차원 형상을 표현하기 위한 객체로서, 본 연구에서는 2차원 곡선들의 집합으로 표현하였고, Sculptured Surface를 구성하는 각 2차원 곡선에 대한 정보를 저장하기 위해, Definition Curve라는 객체를 Sculptured Surface 객체의 파생 객체로서 정의하였다. 한편, Planar Surface 객체는 종격벽, 횡격벽 등과 같이 평면으로 이루어지는 2차원 형상을 표현하기 위한 객체이다. Geometry 객체의 자료구조는 Table 3에 나타나 있다.

이상에서 언급한 객체들을 근간으로 작성된 객체 모형도는 Fig. 1에 나타나 있다.

Table 3 Data structure of geometry object

Attribute 이름	의 미	비 고
GeometryID	해당 Geometry의 이름	
GeometryType	기하학적 형상의 종류를 나타내는 것으로서 다음에 열거한 것 중의 하나가 명시된다. - Sculptured Surface - Planar Surface - Definition Curve	- 각 Geometry Type은 Geometry 객체의 파생 객체로서 선언되어 있으며, 각 파생 객체는 해당 Geometry Type을 정의하기 위해 필요한 데이타 구조를 갖는다.

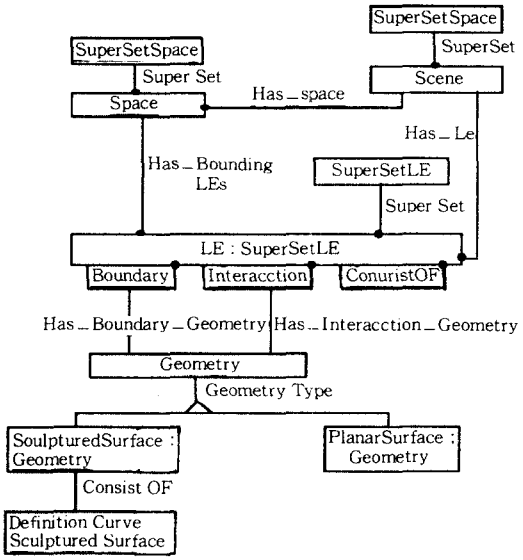


Fig. 1 Object model diagram for ship compartmentation

3. 선박 구획 배치 정의 프로그램의 구현

3.1 일 반

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 택함으로써 시스템을 구현할 수 있다.

앞서 정의된 Logical Element, Space 및 Geometry 등의 기본적인 객체들에 메소드를 추가하여 시스템 구현 요소인 클래스들을 정의한 후 OO_COMDEF 프로그램을 개발하였는데, 이의 실제적인 구현을 위해서 객체지향 언어인 C++를 사용하였다. 또한, 프로그램에서 생성된 객체들의 보존 및 검색을 위해서, 객체들의 데이타를 데이타베이스 시스템에 저장하여, 필요시 저장된 객체들을 검색하여 활용할 수 있도록 하였다. 그리고, 프로그램의 활용 측면에서 사용자가 사용하기에 편리하도록 그래픽 사용자 인터페이스 도구인 OSF/Motif를 이용하여 menu driven 방식으로 프로그램을 구현하였다.

3.2 클래스의 정의

클래스라 함은 임의의 객체를 표현하는데 필요한

데이타(멤버데이타)와 그 객체를 조작하기 위해 필요한 메소드(멤버함수)를 하나의 정보 단위로 정의해서 사용할 수 있는 사용자 정의 자료형(User-defined Type)이다. 전통적인 프로그램밍 언어중 하나인 FORTRAN에서 임의의 변수를 정수형으로 사용하고자 할 경우 Integer라는 자료 정의형으로 해당 변수를 선언해서 사용하게 된다 이때 Integer라는 자료 정의형은 FORTRAN 언어가 제공하는 내장 자료형이다. 이에 비해 객체 지향 프로그래밍 언어에서는 임의의 객체가 갖고 있는 특성(멤버 데이타와 멤버 함수)을 클래스로 정의하고, 주프로그램 작성시에 원하는 객체를 해당 클래스로 선언해서 사용한다. 그러므로, 앞서 언급한 객체들에 대한 클래스를 정의함으로써, 주프로그램(main program)에서는 이들 클래스 자료형으로 여러가지의 해당 객체 인스턴스(instance)를 선언하여 사용할 수가 있다. 객체 인스턴스라 함은 사용자가 정의한 클래스 자료형으로 선언되는 객체를 말한다. 예를 들면, 상갑판면은 Logical Element 객체로 취급되는데 주프로그램에서 상갑판면을 정의하기 위해서는 Logical Element 클래스로 선언해 주면 된다. 이때, 상갑판면을 Logical Element 객체의 인스턴스라고 한다.

OO_COMDEF 프로그램의 구현을 위한 클래스를 정의하는 과정에서, 앞서 언급한 바와 같이 구획 배치를 위해 실질적으로 필요한 Logical Element, Geometry, Sculptured Surface, Planar Surface, Definition Curve 클래스 외에도, 객체 인스턴스들의 효율적인 관리 등 프로그램 구현을 위해서 다음과 같은 클래스들을 정의하였다.

- Scene 클래스:
  사용자 측면을 고려한 개념적인 클래스로서, 임의의 작업 시간 동안 작업한 정보를 grouping해서 관리할 수 있도록 정의한 클래스이다. 또한, 한 척의 배를 여러 부분으로 나누어 별도로 작업하고자 할 때 그 각각에 대해 작업한 정보를 분할 관리할 수 있도록 한다.

- SuperSetScene 클래스:
  정의된 Scene 객체 인스턴스들을 관리하는 클래스이다. 앞서 Scene 클래스에서 설명한 바와 같이, 한 척의 배를 여러 부분으로 나누어 별도로 작업했을 경우, 그 각각의 Scene 객체 인스턴스들은 SuperSetScene 클래스로 선언된 객체 인스턴스에 자동적으로 저장된다.

- SuperSetLE 클래스:
  정의된 Logical Element 객체 인스턴스들을 관리하는 클래스이다. 이 클래스로 선언된 객체 인스턴스는 구획 배치 작업 과정에서 정의한 모든 Logical Element 객체 인스턴스를 보관한다.

- SuperSetSpace 클래스:
  정의된 Space 객체 인스턴스들을 관리하는 클래스로서 구획 배치 작업 과정에서 정의한 모든 Space 객체 인스턴스를 보관한다.

이상에서 설명한 바와 같이 OO_COMDEF 프로그램의 구현을 위해서 정의한 클래스의 구성도는 Fig. 2와 같다.

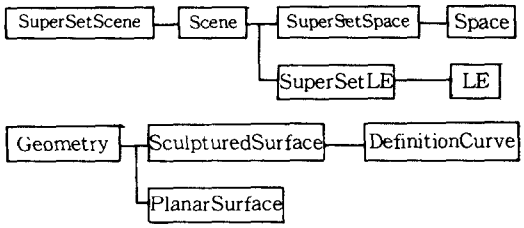


Fig. 2 Class structure of OO_COMDEF program

Fig. 2에 나타나 있는 클래스들을 정의하기 위해서 해당 클래스가 가져야 할 멤버 데이타와 멤버 함수를 결정하였는데, 이중에서 Logical Element 클래스에 대한 멤버 데이타와 멤버 함수의 기능을 살펴보면 다음과 같다(Table 4 참조).

(1) Logical Element 클래스의 멤버 데이타
  Logical Element 클래스는 구획의 경계면, 즉 구조부재 혹은 비구조 부재 요소(이하 '부재요소'라 함)를 정의하기 위한 자료형으로서, "2.3 객체 모델의 정의"중 "(1) Logical Element 객체"에서 설명한 바와 같은 멤버 데이타를 갖는다.

(2) Logical Element 클래스의 멤버 함수
  Logical Element 클래스의 멤버 데이타를 조작하기 위해 필요한 기능을 갖는 멤버 함수를 Table 4에 나와 있는 바와 같이 정의하였는데, 이중 주요한 멤버 함수들에 대한 기능을 살펴보면 다음과 같다.

① setLe(char* surfaceName)
정의하고자 하는 Logical Element 객체의 멤버 데이타 값을 초기화시킨다. 함수 인자인 surfaceName은 IEName 멤버 데이타에 부여(assign)되고, 이 surfacName에 "G_"를 붙인 이름을 IEGeometry 멤버 데이타에 부여한다. 그리고, 나머지 멤버 데이타인 this-LEBoundaryList, this LE IntersectionList 및 parentLE를 초기화 한다.

② getLe( )
임의의 Logical Element 객체의 멤버 데이타인 IEName이 갖고 있는 값을 반환하는 기능을 수행한다.

③ getLEGeometry( )
임의의 Logical Element 객체의 형상이 정의되어 있는 IEGeometry 멤버 데이타 값을 반환하는 기능을 수행한다.

④ findBoundaryLE(char* surfaceName)
사용자가 지정한 surfaceName이라는 Logical Element 객체와 Boundary 관계에 있는 객체들이 있는지의 여부를 검색하는 기능을 수행한다. 반환되는 값으로서는 Boundary 관계에 있는 Logical Element 객체의 갯수이다.

⑤ boundedBy(char* surfaceName)
임의의 Logical Element 객체와의 Boundary 관계를 설정하는 기능을 수행한다. 함수인자인 surfaceName은 Boundary관계를 갖게 되는 Logical Element의 객체 이름으로서, 사용자가 입력한 값이다. 이 값은 thisLEBoundaryList 멤버 데이타에 저장된다.

⑥ removeBoundaryLE(char* surfaceName)
임의의 Logical Element 객체가 가지고 있는 Boundary Logical Element 객체들 중에서 surfaceName을 갖는 것을 삭제하는 기능을 수행한다. 즉, surfaceName이라는 Logical Element와의 Boundary 관계를 삭제시키는 기능을 수행한다.

⑦ removeAllBoundaryLEs( )
임의의 Logical Element 객체와 Boundary 관계에 있는 모든 Logical Element들을 삭제하는 기능을 수행한다.

⑧ listBoundaryLEs( )
임의의 Logical Element 객체와 Boundary 관계에 있는 모든 Logical Element들의 이름을 반환하는 기능을 수행한다.

⑨ drawBoundaryCurve( )
임의의 Logical Element 객체와 Boundary 관계에 있는 모든 Logical Element 객체들과의 경계선을 계산한 후, 그 형상을 화면상에 그리는 기능을 수행한다.

⑩ deleteGeometry( )
임의의 Logical Element 객체의 형상을 표현하고 있는 해당 Geometry 객체에 관한 모든 정보를 제거시키는 기능을 수행한다.

⑪ copyFromDB( )
데이타베이스 내에 저장되어 있는 임의의 Logical Element 객체에 대한 모든 정보를 검색하여 in-memory상에 저장하는 기능을 수행한다.

⑫ copyToDB( )
in-memory상에서 정의된 임의의 Logical Element 객체에 대한 모든 정보를 데이타베이스 내에 저장하는 기능을 수행한다.

Table 4 Member data and functions for logical element class

LE
IEName : char
IEGeometry : char
thisLEBoundaryList : BoundaryList
thisLEIntersectionList : IntersectionList
parentLE : char
setLE(char* surfaceName)
getLE( ) : char*
getLEGeometry( ) : char*
findBoundaryLE(char* surfaceName) : int
boundedBy(char* surfaceName)
removeBoundaryLE(char* surfaceName) : int
removeAllBoundaryLEs( )
listBoundaryLEs( )
findIntersectionLE(char* surfaceName) : int
intersectedBy(char* surfaceName)
removeIntersectionLE(char* surfaceName) : int
removeAllIntersectionLEs( )
listIntersectionLEs( )
drawBoundaryCurve( )
drawDefCuve( )
deleteGeometry( )
copyFromDB( )
copyToDB( )

3.3 데이타베이스 설계

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 저장하고 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 저장하고 검색할 수 있는 방법을 모색하였다.

OO-COMDEF 프로그램에서 데이타베이스 시스템에 요구하는 기능은 크게 두가지가 있다. 첫번째는, *OO-COMDEF*의 수행시에 참조되는 객체들의 멤버 데이타 초기값을 데이타베이스로 부터 검색하여 멤버데이타를 초기화시키는 기능이다. 두번째는, 수정된 객체들의 멤버 데이타를 데이타베이스에 Persistent Object로서 저장하는 기능이다. 이 두가지 기능의 구현을 위해서, 객체의 멤버 데이타를 초기화시키는 함수와 정의된 객체들의 멤버 데이타를 데이타베이스에 저장하는 함수를 각 클래스의 멤버 함수로 정의하였다. 이 두가지 함수를 이용하여 OO-COMDEF 프로그램의 수행 중에 참조되는 객체가 in-memory상에 존재하지 않으면 데이타 베이스로 부터 해당 객체의 멤버 데이타를 검색하여 객체를 초기화시키고, 해당 객체에 대한 작업이 종료된 후에, 멤버 데이타를 데이타베이스에 저장되도록 하였다. 이와 같이 in-memory상의 객체와 데이타베이스에 저장된 Persistent Object를 효율적으로 관리하게 함으로서, 매번 데이타베이스를 access할 경우에 발생되는 성능 저하를 방지하였다.

본 연구에서는 CSDP-데이타베이스 관리시스템 [13]에서 제안한 확장형 ER(Entity-Relationship) 모형을 이용하여 관계형 데이타베이스의 논리적 설계 방법론에 따라 상향식과 하향식을 혼합한 방법으로 관계형 데이타베이스의 논리적 구조인 릴레이션들의 스키마를 도출하여 선박구획배치 데이타베이스를 설계하였다.

선박구획배치 모델에서 정의된 각 클래스들의 멤버데이타를 엔티티와의 관계로 표현하여 Fig.3과 같은 확장형 ER 모형도를 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 5와 같다.

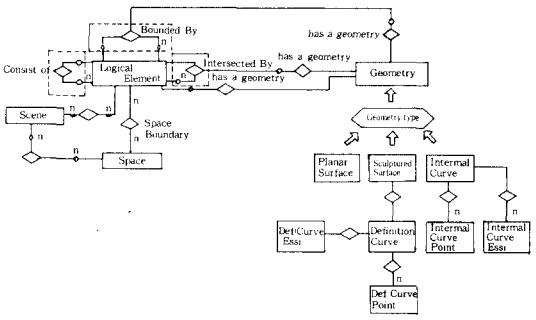


Fig. 3 Extended entity relationship diagram for ship compartmentation

Table 5 Relational schema for OO_COMDEF program

Table Name	Attibule Name
T_Scene	SceneName, WXL, WYL, WXH, WYH, VXL, VYL, VXH, VYH, XE, YE, ZE, XF, YF, ZF, TM11, TM13, TM14, TN21, TM22, TN23, TN24, TN31, TN32, TN33, TM34, TM41, TM43, TM44
T_Scene LE	SceneName, LEName
T_LE	LEName, LEGeometry, ParentLE
T_LE_BLE	LEName, BoundaryLename, BoundaryDirection
T_LE_ILE	LEName, IntersectionLEName
T_InternalCurve	InternalCurveName, InterpolationType
T_InternalCurvePoint	InternalCurveName, PointNo, U, V
T_InternalCurveEssi	CurveName, SegmentNo, Spointn, Spointv, EPointu, Epoinv, A, B, C, Mod
T_Geometry	GeometryID, GeometryType
T_PlanarSurface	GeometryID, XO, YO, ZO, XU, YU, ZU, XV, YV, ZV UL, VL, HU, VH, A, B, C, D, PlaneType
T_SculpturedSurface	GeometryID, XO, YO, ZO, XU, YU, ZU, XV, YV, ZV, XL, YL, XH, YH, ZH
T_SculpturedSurface DefCurve	GeometryID, Defimition CurveName
T_DefCurve	DefinitionCurveName, XO, YO, ZO, XU, YU, ZU, XV, YV, ZV, PlaneType, InterpolationType
T_DefCurve Essi	Curvename, SegmentNo, Spointu, Spointv, Epointu, Epointv, A, B, C, Mod
T_DefCurve Point	DefinitionCurveName, PointNo, Pointname, U, V, W
T_Scene_Space	SceneName, SpaceName
T_Space	Spacename, SpaceFactor, Density, Volume, LCB, TCB, KB
T_Space_Face	Spacename, FaceName, SideIndicator

3.4 사용자 인터페이스 설계

OO-COMDEF프로그램은 선박구획배치 및 정의 시 설계자가 대화식에 의한 설계 대안을 반복적으로 수행하여야 하는데 종래의 명령어 입력 방식으로는 명령어의 Syntax 오류와 작업 순서상의 오류 등이 자주 발생할 수 있다. 따라서, 본 프로그램에서 채용한 사용자 인터페이스는 이러한 사용자의 명령어 입력의 오류를 방지할 수 있고 또한 사용하기에 편리한 개념을 채용하였다.

본 프로그램에서 사용한 사용자 인터페이스는 X-윈도우시스템 기반의 OSF/Motif를 근간으로 하였다. 그리고, 기하학적 형상의 가시화를 위해서는 X-윈도우 환경에서 수행될 수 있도록 자체적으로 개발한 3차원 그래픽 라이브러리인 GLBAX(Graphic Library Based on X)를 이용하였다.

Motif에서 제공하는 버튼(button), 팝업 메뉴(pop-up menu), 풀 다운 메뉴(pull-down menu) 등과 같은 다양한 형태의 그래픽 사용자 인터페이스 요소 등을 이용하여, 정의하고자 하는 객체들의 멤버 데이타 및 멤버 함수들을 menu-driven 방식으로 처리하게 함으로서 사용자가 편리하게 사용할 수 있도록 하였다.

4. 적용 예

본 연구를 통해 개발한 OO-COMDEF 프로그램을 이용하여 구획 배치 모델링을 직접 수행하면서 객체지향 선박구획 표현 방법론의 정합성을 검증하였다.

본 장에서는 적용 예로 Logical Element 객체의 정의 및 그들간의 관계 설정, Space 객체를 정의하는 절차 등을 프로그램 실행 순서와 함께 프로그램 내부처리 과정 측면에서 설명하고, 그 결과에 따른 실행예를 보였다.

4.1 상갑판면의 정의

일반적으로 구획 배치 모델링 작업은 선체 형상 내의 구획을 결정하기 위해, 그 경계면들을 우선적으로 정의한다. 구획의 경계면은 부재 요소로서 볼 수 있는데, 본 프로그램에서는 Logical Element 객체로 정의된다. 실제적인 예를 위해 상갑판면의 정의에 대해 설명한다. 상갑판면은 Logical Element 객체로 취급되는 부재요소로서 실질적으로는 곡면 형상을 갖게 되는데, 본 예에서는 편리상 평면 형상을 갖는 것으로 하였다. 그러므로, 상갑판면의 Geometry Type은 Planar Surface가 되고, 이의 정의를 위해서는 프로그램 실행시 나타나는 Menu Bar에서 "Begin Planar Surface" 메뉴를 선택함으로서 수행된다. "Begin Planar Surface" 메뉴를 선택하면 상갑판면 정의를 위한 Input Area pad가 생성되는데 사용자는 다음과 같이 각 데이타 항목에 대한 값을 입력한다(Fig. 4 참조).

여기서 " "로 폐위되어 있는 부분은 사용자가 직접 입력한 값임을 의미한다.

- SurfaceName = "Deck"
- Plane Type = "xy"
- Local Origin(XO, YO, ZO) = "0.0 0.0 17.8"
  (XU, YU, ZU) = "0.0 0.0 0.0"
  (XV, YV, ZV) = "0.0 0.0 0.0"
- Box2D(Umin, Vmin, Umax, Vmax)
  = "−5.0, −16.0, 250.0, 16.0"

이후의 작업 과정과 이에 따른 프로그램 내부처리 절차는 Table 6과 같고, 정의 결과는 Fig. 4에 나타나 있다.

4.2 Logical Element 객체간의 Boundary 관계 정의

본 논문에서 제안하고 있는 선박 구획 배치를 위한 표현 방법에서는, 부재 요소들 간의 상관관계를 설정함으로서 해당 부재 요소의 정확한 형상이 구해지도록 하였다. 즉, 부재 요소의 형상은 자기 자신의 기본이 되는 기하학적 형상을 토대로 하여, 다른 부재 요소들에 의해 그 범위가 제한되도록 Boundary 관계를 설정하여 정확한 형상이 정의되도록 하였다.

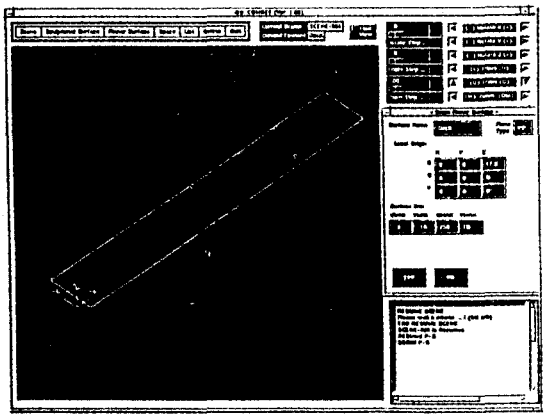


Fig. 4 The definition of planar surface for upper deck("Deck")

Table 6 Operating Functions and Data Structure for Upper Deck("Deck") Definition

수행 내용	구동 함수	멤버 데이타 자료 구조
Logical Element 객체의 정의를 위한 기억장소 할당	currentLEP - 1EAllocate( )	
Logical Element 객체의 멤버데이타인 1EName 및 1EGeometry에 이름 부여	currentLEP.setLE("Deck")	LE 1Ename: "Deck" 1EGeometry: "G_Deck" thisLEBoundaryList: – thisLEIntersectionList: – parentLE: –
Geometry 객체의 정의를 위한 기억장소 할당	currentGeometryP = geometryAllocate( )	
Geometry 객체의 멤버데이타인 geometryID 및 geometryType 이름 부여	currentGeometryP. setGeometry("G_Deck") currentGeometryP. setGeometryType("Planar")	Geometry geometryID: "G_Deck" geometryType: "Planar"
Planar Surface 객체의 정의를 위한 기억장소할당	currentPlanarSurfaceP = planarSurfaceAllocate( )	
Planar Surface 객체의 멤버데이타인 geometryID, planeType, localOrigin 및 box2D에 값 부여 planar Surface의 평면 방정식계수의 계산 및 멤버데이타인 pCoef의 값 부여	curentPlanarSurfaceP. setPlanarSurface("G_Deck") curentPlanarSurfaceP. setPlanarSurfaceType("xy") curentPlanarSurfaceP. setPlanarOrigin(LocalOrigin) curentPlanarSurfaceP. setBox(Box2D) curentPlanarSurfaceP. calculateCoef( )	PlanarSurface geometryID: "G_Deck" localOrigin: (0, 0, 17.8) (0, 0, 0) (0, 0, 0) box2D: (-5.0, -16.0, 250.0, 16.0) pCoef: (a1, a2, a3, a4) planeType: "xy"
SuperSetLE 객체의 등록	surperSetLE. addLE("Deck")	SuperSetLE allLEList: {"Deck"} 여기서, { }은 liked list 형태로 정의된다는 것을 의미한다.

앞서 설명한 상갑판면의 정의는 자기 자신의 기본이 되는 기하학적 형상만을 정의한 예이고 상갑판면과 Boundary 관계를 정의(Fig. 5 참조)함으로서 상갑판면의 형상을 결정(Fig. 6 참조)하게 되는데 그 과정은 다음과 같다.

여기서, "TBHD1"과 "TBHD2"는 횡격벽을 나타내고, "BULK1"은 선체 형상을 나타내는 Logical Element(Fig. 7 참조)로서 이미 정의된 것으로 가정한다.

(1) "Planar Surface" 메뉴의 서브 메뉴인 "Resume Planar Surface" 메뉴를 선택한다. 그러면, 현재까지 정의되어 있는 Logical Element 객체들, 즉 "Deck", "TBHD1", "TBHD2", "BULK1" 등이 팝업 메뉴상에 리스팅되는데, 이때 resume 하고자 하는 Logical Element 객체를 선택한다. 여기서 resume이란 해당 객체를 작업 대상이 되는 객체로 선언함을 의미한다. 본 예에서는 "Deck"를 선택하였다.

(2) 선택된 "Deck"가 SuperSetLE 객체의 멤버 데이타인 allLEList에 등록되어 있는지의 여부를 확인한 후, 작업 대상이 되는 객체로 선언한다.

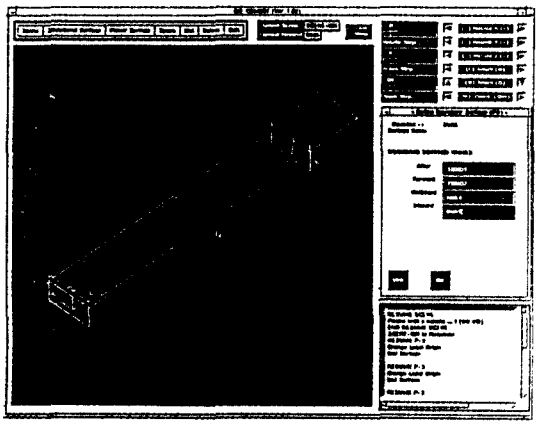


Fig. 5 The definition of boundary relation for upper deck("Deck")

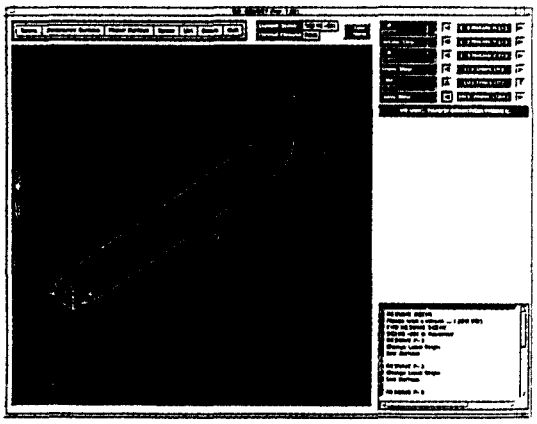


Fig. 6 The shape of defined upper deck("Deck")

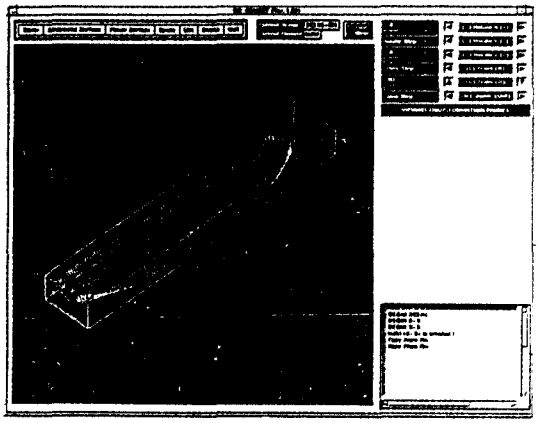


Fig. 7 The definition result f hull form

(3) "Planar Surface" 메뉴의 서브 메뉴인 "Boundary Relation"의 서브메뉴인 "Define Boundary Surface" 메뉴를 선택한다. 그러면, 선택된 메뉴에 의해 나타난 팝업 메뉴상에 현재 작업 대상인 Logical Element의 형상을 제한하는 Logical Element들 즉 "TBHD1", "BHD2", "BULK1"을 입력한다.

(4) 입력한 "TBHD1", "TBHD2", "BULK1"이 SuperSetLE 객체의 멤버 데이타인 allLEList에 등록되어 있는지의 여부를 확인한다.

- 구동함수 : surperSetLE. findLE("TBHD1")
  surperSetLE. findLE("TBHD2")
  surperSetLE. findLE("BULK1")

(5) 현재 작업 대상인 "Deck"와의 Boundary 관계를 설정하기 위해 Logical Element 객체 멤버 데이타인 thisLEBoundaryList에 "TBHD1", "TBHD2", "BULK1"을 등록시킨다.

- 구동함수 : currentLEP. boundedBy ("TBHD1")
  currentLEP. boundedBy ("TBHD2")
  currentLEP. boundedBy ("BULK1")

여기서, currentLEP는 (2)항에서 언급한 바와 같이 현재 작업대상인 "Deck" Logical Element를 가리키고 있는 변수명이다.

(6) Boundary 관계에 의해 생성될 Boundary Curve에 대한 이름을 생성하고, 그에 관한 정보를 저장하기 위한 기억장소를 할당한다.

- 구동함수 : generateInternalCurveName ("b"_, "Deck", "TBHD1")
  boundaryCurveGeometryP = geometryAllocate( )

여기서, generateInternalCurveName 함수는 "b_Deck_TBHD1"라는 Boundary Curve 이름을 생성하는 기능을 수행한다. 이하 "TBHD2"와 "BULK1"에 대해서도 동일하다.

(7) Boundary Curve인 "b_Deck_TBHD1"에 대한 Geometry 객체의 이름과 형상 type을 Geometry 객체의 멤버 데이타인 geometryID와 geometry Type에 부여한다. geometryID는 "b_Deck_TBHD1"의 값을 갖고, geometryType은 "internal"이라는 값으로 정의된다. "internal"의 의미는 geometry 형상이 Bound-

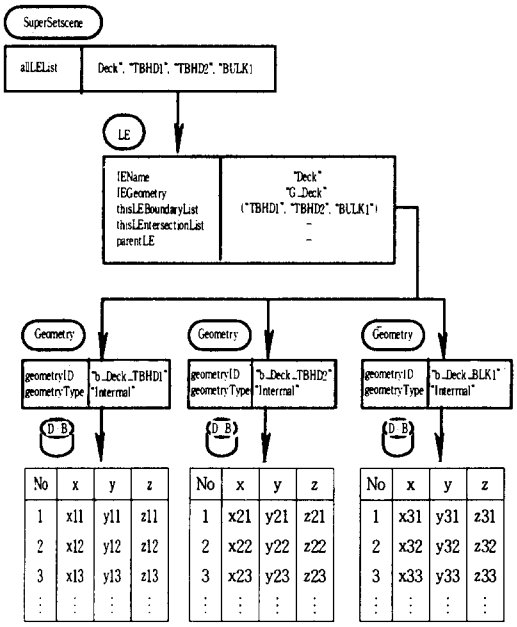
ary 혹은 Intersection 관계로 부터 생겨난 형상임을 의미한다.

- 구동함수 : boundaryCurveGeometryP.
  setGeometry
  ("b_Deck_TBHD1")
  boundaryCurveGeometryP.
  setGeometry Type("internal")

(8) Boundary Curve에 대한 형상 정보, 즉 좌표값들을 데이타베이스에 저장한다.

- 구동함수 : boundaryCurveGeometryP.
  copyToDB("b_Deck_TBHD1")

(9) 이상의 과정을 통해 "Deck"의 형상은 "TBHD1", "TBHD2", "BULK1"에 의해 제한되어져 정의되는데, 이때"Deck"와 관련되는 객체들의 멤버 데이타는 다음과 같이 저장된다.



4.3 Space 객체의 정의

Space 객체는 이미 정의된 Logical Element 객체들을 경계면으로 사용하여 정의하게 된다. 이의 실제적인 예를 위해 "HOLD1"이라는 Space의 정의에 대해 설명한다. 이때 "HOLD1" Space를 구성하는 경계면으로서 상갑판면 "Deck", 횡격벽 "TBHD4", "TBHD5", 선저판 "INBTM" 및 선체 형상을 나타내는 "BULK1" Logical Element들은 이미 정의된 것으로 가정한다.

(1) 프로그램 실행시 나타나는 "begin space" 메뉴를 선택하면 Space 객체 정의를 위한 Input Area Pad가 생성되는데 사용자는 다음과 같이 각 데이타 항목에 대한 값을 입력한다.

- SpaceName = "HOLD1"
- Bounding Faces
  After = "TBHD4"
  Forward = "TBHD5"
  Lower = "INBTM"
  Upper = "Deck"
  Starboard = "BULK1"
  Port = "BULK1"
- Space Factor = "1.0"
- Density = "1.025"

(2) "HOLD1"을 Space 객체로 정의하기 위해 필요한 기억장소를 할당한다. 기억장소를 할당하는 함수가 수행되고 나면, 할당된 기억 장소의 번지수(currentSpaceP)를 반환하는데, 이 번지수가 가리키는 기억 장소에 해당 정보를 저장한다.

- 구동 함수 : currentSpaceP
  = spaceAllocate( )

(3) "HOLD1" Space를 정의하기 위해 입력한 데이타를 각 멤버 데이타에 부여한다.

- 구동함수 : currentSpaceP. setSpace
  ("HOLD1")
  currentSpaceP. setSpaceFactor
  ("1.0")
  currentSpaceP. setDensity
  ("1.025")
  currentSpaceP. boundedBy
  ("TBHD4", "A")
  currentSpaceP. boundedBy
  ("TBHD5", "F")
  currentSpaceP. boundedBy
  ("INBTM", "L")
  currentSpaceP. boundedBy
  ("Deck", "U")
  currentSpaceP. boundedBy
  ("BULK1", "S")
  currentSpaceP. boundedBy
  ("BULK1", "P")

(4) "HOLD1" Space의 용적 및 용적 중심을 계산하여, 그 결과를 멤버데이타인 volume, lcb, tcb, kb에 부여한다.

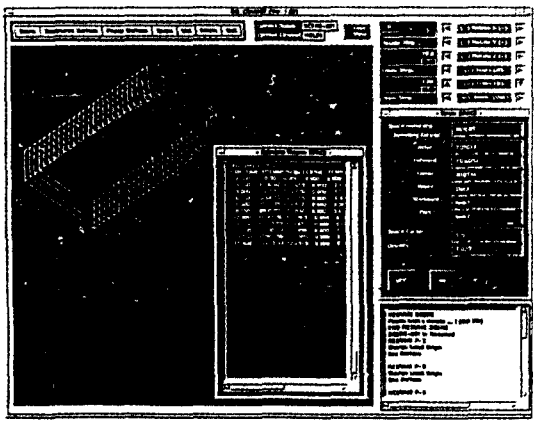


Fig. 8 The definition result of space("HOLD1")

- 구동함수 : currentSpaceP.
  calculateVolume( )

(5) 이상에서 언급한 절차에 의해 "HOLD1"에 관한 정의가 끝나면 Space 객체를 관리하는 SuperSetSpace 객체에 "HOLD1"을 등록시킨다.

- 구동함수 : superSetSpace
  addSpace("HOLD1")

(6) 이상의 과정을 통해 "HOLD1"을 정의한 결과는 Fig. 8에 나타나 있다.

5. 결 론

본 연구를 통해 시스템 구현 측면에서 객체 지향 시스템 기술을 토대로 한 선박구획 배치 모델의 표현 방법론을 정립하였다. 객체지향 선박구획 배치의 모델링 과정을 통해 응용 분야의 개념적이고 추상화된 요구사항을 충분히 반영시킨 제품 모델링을 실제적으로 전산기 내에서 구현하려면 데이타와 응용 프로세스가 하나의 정보단위인 객체로 취급되는 객체지향 패러다임을 적용하므로서 전통적인 방식에 비해 비교적 쉽게 구현될 수 있음을 알 수 있었다.

본 논문에서 제안한 객체 지향 선박구획배치 모델의 표현방법론을 Unix 엔지니어링 워크스테이션에서 구현시켜 검증한 결과, 설계자의 선박설계 개념과 복잡성을 내포한 자료구조를 전산화 된 코드로 표현할 수 있는 효율적인 접근방법임을 알 수 있었다.

본 연구를 통해 제안된 표현방법론을 CSDP 연구사업의 선박제품모델 관련기술 및 선체 CAD 시스템 개발을 위한 Framework의 첫단계로서 제시하고자 하며, 보다 실질적인 시스템 구축을 위해 다음과 같은 내용들을 계속적으로 연구 개발하여 시스템을 확장하고자 한다.

- 선박 기본 계산(유체정역학적 재 계산, 비손상시 및 손상시 복원성 계산, 종강도 계산 등) 기능 추가
- 선박 제품 모델 구축을 위해 각 설계 단계별 주요 정보와 관련된 객체 클래스의 도출 및 확장
- OODB(Object Oriented Database)의 적용
- 기존 조선용 CAD/CAM 시스템 과의 연계 방안 모색

6. 후 기

본 연구는 CSDP 사업과 관련하여 수행된 연구 결과의 일부임을 밝혀 둔다.

참 고 문 헌

[1] P.R. Wilson, "A short history of CAD data transfer Standards," IEEE Comp. Graphics & Application, Vol 7, No 6, pp64~67, 1987.

[2] P.R. Wilson, "PDES STEP forward," IEEE Comp. Graphics & Application Vol 9, No 2, pp79~80, 1989.

[3] M.S. Bloor and J. Owen, "CAD/CAM product data exchange : the next step," CAD, Vol 2, No 4, pp237~243, 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 TC184/SC4/WG3 Document N 45, 1991.

[7] STEP, Part 102 : Ship structures, ISO TC184/SC4/WG1 Document N411, 1989.

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

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

[10] James Rumbaugh and Michel Blaha, "Ob-

ject-oriented modeling and design," Prentice hall inc., 1991.

[11] 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.

[12] 이규열, 서승완, 강원수, "CSDP(Ⅲ)-종합시스템개발", 해사기술연구소 연구보고서, 1991.

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

[14] Stroustrup, "The C++ programming language," AddisonWesly, 1986.