Journal of
Ship&Ocean
Technology
©JUNE 2001

SOTECH Vol. 5, No. 2, pp. 1~13, 2001

An Instance-Oriented Modeling Method for Shipbuilding Applications

Shinro Hamada[1] and Kiyoshi Konaka[1]

[1]The Ishikawajima-Harima Heavy Industries, Co., Ltd., Japan;
E-mail: hamada@soe.ty.ihi.co.jp

Abstract

Data in one Product Model for shipbuilding is inevitably referenced and manipulated during each phase of design or production activities, and data or manipulation status naturally varies from the original with the advance of each activities. For Object-Oriented approach, it is hard to identify classes dealing with those variations, and even if once a product model is developed, it might be getting much harder to modify it to cope with a new additional phase of activities. This paper proposes an Instance-Oriented Modeling Method, temporarily named "Concept-Relationship Modeling Approach", which handles Data structure and Behavior independently of each other in order to resolve the difficulties above.

Keywords: product model, instance-oriented modeling method, object oriented approach

1 Introduction

A major need felt by computerized design circles is a Product Model that will integrate and share in common all the different kinds of information concerning a product. In the domain of shipbuilding, various attempts have been made in the past for creating such a Product Model covering the various aspects of shipbuilding design and construction. The interest of shipbuilders and research institutions both in Japan and abroad have in particular centered around Object-Oriented shipbuilding Product Models. Shipbuilding Product Models have in common the characteristics of:

- The same information requiring to be referable and manipulable to serve different operations
- The object of information tending to change with progress of design or manufacture

The foregoing characteristics of the shipbuilding Product Model make it difficult to design classes that satisfy all needs, and even if realized, the structure would be even more difficult to modify for adapting it to suit additional operations.

With a view to solving the above difficulties, it is here proposed an Instance-Oriented Modeling Method, temporarily named "Concept-Relationship Modeling Approach", which handles Data structure and Behavior independently of each other. The present report describes this method, together with examples of its application.

2 Key points in creating a product model for shipbuilding

The creation of a conventional Object-Oriented Product Model proceeds along the follow line.

1. Clearly envisage the substances of all operations needing to be covered, together with the activities needed for performing the operations.
2. Pick out requisite information separately as Data structure, Attribute and Behavior.
3. Using the information thus picked out, design classes that will express the Product Model.

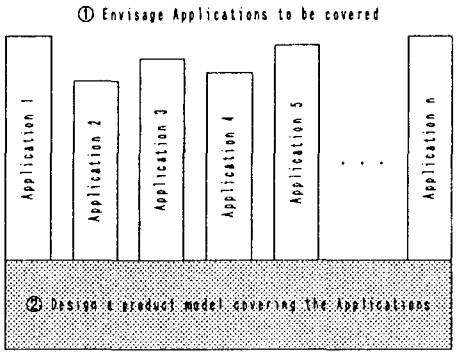

Figure 1: Creation of a conventional object-oriented model

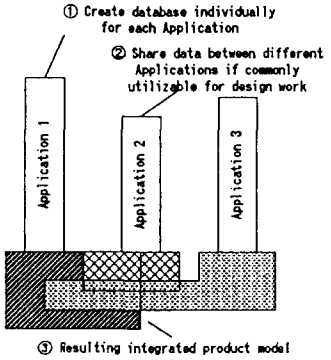

Figure 2: Creation of an instance-oriented model

Such a conventional Object-Oriented Product Model is illustrated in Figure 1. Existing shipbuilding Product Models can be said to take the form of Object-Oriented Product Models, which present the difficulties cited below.

2.1 Difficulty of analyzing/designing the Product Model

It is difficult to envisage and analyze beforehand the substance of all requisite operations, for which to pick out separately as Data structure, Attribute and Behavior the information that is necessary and sufficient to cover all the operations, based on which to design classes that express an adequate shipbuilding Product Model. The work of picking out requisite information and designing the Product Model would call for expenditure of unaffordable time, and the resulting class would tend to be impractically cumbersome to manipulate.

The above difficulties can be ascribed to the particular characteristics of shipbuilding operations such as:

- Multiformity of the operations involved, ranging from design to manufacture, and eventually even further.
- Many cases where the approach will vary from operation to operation.
- Complexity of the object of information - the ship - comprising a formidable number and variety of components.

2.2 Difficulty of extending the range of system application

A ship lasts through several decades of service life, and the Product Model system is required to be capable of following the changes that will inevitably occur during the long lifetime of the object - the ship. With the conventional Object-Oriented Product Model, it is difficult to newly add operations and applications that have not been envisaged at the time of original system design, and it may become necessary to re-create anew the Product Model itself. As a rule, Product Model systems created on Object-Oriented principle are said to be easy to modify, needing only to newly input the Attributes or Behavior that come to be additionally needed. In practice, it is not so difficult to extend the range of application of a shipbuilding Product Model created on Object-Oriented principle, so long as the extension does not overstep the originally designed framework of classes. On the other hand, if this framework has to be modified for adapting it to a new application, such as for incorporating an aspect not until then envisaged, this cannot be accomplished by simple addition of needed information in the case of a system based on Object-Oriented principle.

2.3 Problem of adapting Behavior coverage to progress of operation

A ship will also involve changes in Data structure, Attributes and Behavior with progress of design and manufacture. Taking as example stiffeners, in the initial stages of design, it should suffice to be able to manipulate information simply covering the lines along which to install them and the relevant attributes, but with progress of design, it would become necessary to manipulate more detailed information such as the stiffener cross section. Further progress would then call for additionally handling information on such detailed as extremity treatment and welding joints.

It is difficult with the conventional Object-Oriented system to modify the Behavior to match the progress of operations, since in the normal Object-Oriented programming language and database, the class to which the object belongs cannot be modified in substance once it is created.

3 Concept-relationship product modeling approach

We propose an Instance-Oriented Modeling Method - "Concept-Relationship Modeling Approach" - to resolve the difficulties cited above that are inherent in the conventional Object-Oriented Product Model.

3.1 Concept-relationship modeling

In Concept-Relationship Modeling, the information requiring to be stored is converted to the utmost into abstract form, and the modeling proceeds with the treated information classified into solely three Categories, which are the "Concept", the "Attribute", and thirdly the "Relationship" that provides the link between Concepts (see Figure 3).

(a) Concept is the generic term applied to information expressed in noun-phrase form concerning a conceivable object, such as "ship", "hull", "engine room" or "bow tank".

(b) Attribute is the generic term applied to information specifying the quality of the Concept in alphanumeric symbols, such as - in the case of the concept "ship" - "length", "breadth", "height".

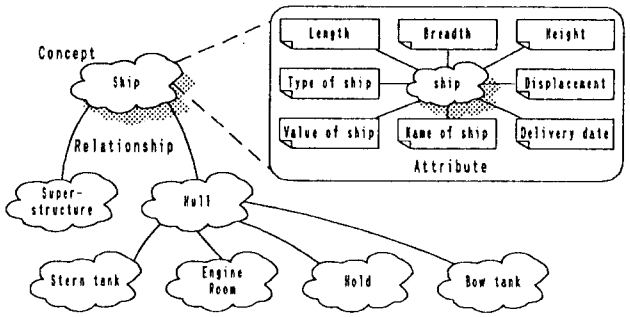

Figure 3: Elements of concept-relationship model

(c) Relationship is the generic term applied to information that links together different Concepts.

Concepts can be linked to each other by several different Relationships. A Relationship, in turn, can express different forms of linkage between Concepts, such as between the whole and part of an object, sequences in time or in space, or between relative positions in society or in an organization.

3.2 Object-oriented and instance-oriented

With conventional Object-Oriented Product Modeling, an information is fixedly bound, Object by Object, to the class definition. With Instance-Oriented Product Modeling, on the other hand, attention is focused on the changeable independent character of the object, to constitute a flexible model.

3.3 Characteristics of the Concept-Relationship Product Model

A Concept-Relationship Product Model presents the characteristics of: - Data classification structure comprising solely the three Categories of Concept, Attribute and Relationship, which constitute the framework of the Concept-Relationship Product Model.

- Information for expressing different Activities and applications formulatable independently of each other within a flexible framework.

- Ready utilization possible of any existing information available in other Applications.

Information needed for expressing different Activities and Applications formulated as above constitutes an integrated Product Model (see Figure 2).

4 Solution with concept-relationship product modeling of difficulties inherent in the conventional product model

The manner in which the difficulties inherent in the conventional Product Model are overcome with Concept-Relationship Product Modeling is describes in what follows, together with examples of practical application.

4.1 Analysis and design of product model

It was pointed out as a difficulty of the conventional Object-Oriented Product Model that it required picking out and analyzing beforehand all the information related to all relevant operations and applications, upon which to design classes for expressing the Product Model.

With Concept-Relationship Product Modeling, the requisite information analysis and class designing can be performed individually and independently for each operations or applications. This diminishes the range of information needing to be covered at each stage, and facilitates the work of analysis and design required for creating the Product Model.

As an example, an application will be taken up covering the fairing operation of full contour. This application treats offset points and operation lines connecting the points for constituting a lattice representing the full contour (see Figure 4).

The first step in the procedure is to pick out requisite information individually for each relevant operation or Application, following conventional Object-Oriented practice. The result of analysis is a class such as shown in Figure 5, represented in UML (Unified Modeling Language). It indicates that an Operation line object is constituted of two or more Offset point objects, and that an Offset point object can constitute classes of up to two Operation line objects. At this stage,

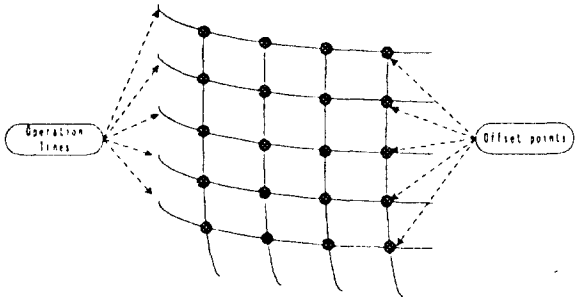

Figure 4: Example of lattice representing full contour

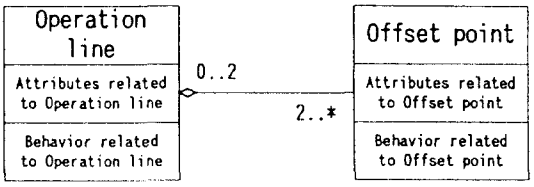

Figure 5: Classes for hull form fairing application

it suffices to pick out Attributes and Relationships related solely to the relevant hull form fairing application. The next step is to divide the class into parts of "Data object" - solely covering Attributes - and "Control object" - solely covering Behavior. In doing this, the Control object part can be made to cover also basic operations such as settings and references related to Attributes and Relationships (see Figure 6).

Storage in database is limited to the Data object. The other Control object is accommodated in the application side. When utilizing an application, the corresponding Data object and Control object are combined into a single object. The Attributes and Relationships of the Data object

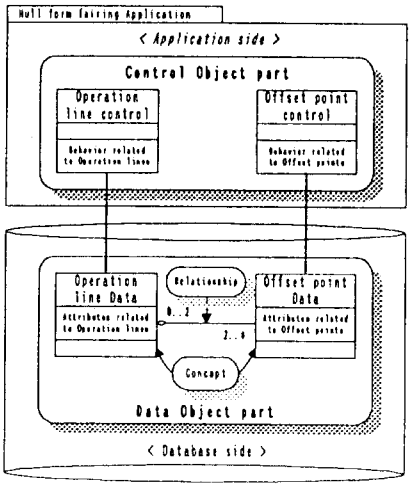

Figure 6: Classes, divided between data and control object parts, for full form fairing application

stored in the database are then manipulated through the Behavior covered by the Control object.

This means that, in Concept-Relationship Product Modeling, the object part - i.e. application side - controls all information that, in Object-Oriented Product Modeling, would carry a tag indicating the category to which it belongs. With Concept-Relationship Product Modeling, the database will solely contain Concepts devoid of indication about its assignment to any Category, but which are instead each assigned with Attribute or Attributes, and are linked to other Concepts through various Relationships. The separation of information into Data object and Operation object, which are treated indiscriminately in the conventional Object-Oriented system, is a prominent feature of Concept-Relationship Product Modeling. The elimination from database of information carrying a Category tag dispenses with the need of analyzing and designing the entire Product Model beforehand, and permits proceeding with the analysis and design individually on each application.

4.2 Extensibility of the system

Another problem that was cited regarding the conventional Object-Oriented Product Model was the difficulty of adding operations and applications that had not been envisaged at the time of original system design. With Concept-Oriented Product Modeling, such operations and applications that come to be needed with progress of operations can be simply added by supplementing the information contained in an existing Product Model.

The example taken up this time is an additional application to supplement the Product Model obtained in the preceding section, where the application that was formulated covered the operation of full contour fairing.

The additional application is to generate the curved hull surface that had been derived by fair-

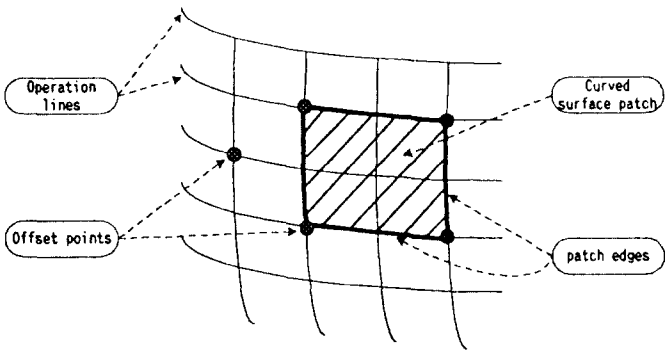

Figure 7: Fitting in curved surface patches to constitute hull contour

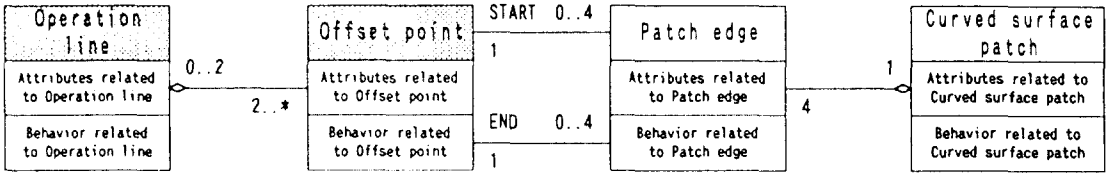

Figure 8: Classes for curved surface patch defining application

ing: In this application, multiple patches to constitute the hull contour are defined from the curves smoothed by fairing (see Figure 7). Following the procedure described in the preceding section, the first step is to pick out requisite information individually for each relevant operation or application, using conventional Object-Oriented procedure. Analysis of this application for defining multiple patches on the curved surface has proved the application to need covering, similarly to the preceding case, the objects of Operation line and Offset point. The analysis has further indicated as requisite additional objects a Curved surface patch object controlling the curved patch, together with its constituent element of Patch edge object.

The class derived from the analysis is shown in Figure 8, which indicates that a Curved surface patch object is constituted of four Patch edge objects, and that a Patch edge object is related to two Offset point objects respectively indicating the initial and terminal points of the patch edge. The ensuing step is, again following the preceding procedure, division of the class into Data object and Control object parts for assigning Attributes and Behavior, respectively. In doing this, the Data objects already generated in the preceding procedure are directly utilized for the objects of Offset point data and Operation curve data. Objects additionally found to be requisite, such as the two cited above, can be accommodated at this stage in the Data object part (see Figure 9).

It is to be recalled that, in Concept-Oriented Product Modeling, Concepts, Attributes and Relationships are modeled in such manner as to constitute separate Data objects, so that supplementation with additional objects or with additional Attributes or Relationships concerning a Concept can be easily performed at the time of execution without having to modify the database structure.

In the foregoing manner, existing available Data objects of similar conceptual signification can be directly utilized, and which can be easily supplemented by additional applications that may come to be found requisite.

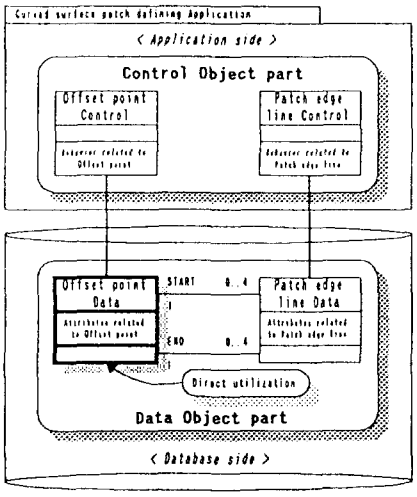

Figure 9: Classes, divided between Data and control object parts, for curved surface patch defining application

The procedure for packaging the Concept-Relationship Product Model in database will be described in the ensuing Chapter 5.

4.3 Modification of behavior for adapting to progress of information

A further problem mentioned earlier concerning the conventional Product Model was the difficulty of modifying a Behavior related to a given object that should come to require alteration with progress of operation. With Concept-Relationship Product Modeling, a Behavior can be flexibly changed for each application.

A single Data object being susceptible to assignment of Multiple Control objects, it is the application side that determines which of the Control objects to assign to a Data object information that is retrieved from database. This means that a Behavior would differ between an application X which Control object capable of modifying the Attribute and an application Y solely provided with read-only capability for the Attribute.

The Two applications treated in the preceding section will be taken up as example for describing the above activity modification.

As the first step, for the purpose of fairing the hull contour, the hull form fairing application side will accommodate the Activity A for modifying the curve configuration, covering:

- Offset point Control object: "To displace the Offset points by indicated amounts, and to inform the Operation curve upon accomplishment of displacement"

- Operation curve Control object: "Upon receiving the information on Offset point displacement, to redefine the cure configuration".

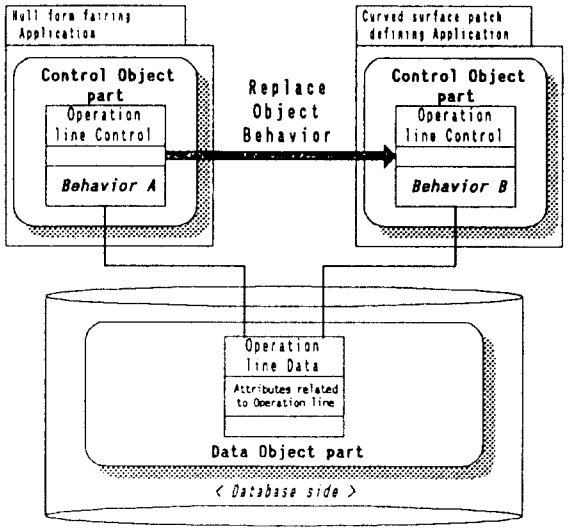

Figure 10: Replacing the behavior of object

On the application side, the Behavior of curve configuration smoothing has already been accomplished in the preceding stage, and needs not to be repeated: To this effect, the Behavior A must be eliminated from the Control object, and instead, this side is to accommodate a Behavior B for defining a curved surface configuration that will determine the patch edges "to redefine the curve configuration in accordance with the displaced Offset points within a specified region."
In this manner, the Control object is so modified as to adapt the Behavior of object to progress of information (see Figure 10).

5 Implementation for the database part of a concept-relationship product model

Implementation for the database part of Concept-Relationship Product Model does not require following any special procedure. Practical examples will be described below of cases where Object-Oriented programming language and database are used for implementation for the database part of a Concept-Relationship Product Model.

Other cases of implementation such as with use of relational database are given in Reference(Hata et al 1998).

5.1 Implementation with use of object-oriented language

The Concept-Oriented system having its origin in the Object-Oriented system, use of the same language and data will facilitate implementation in database of Concept-Relationship Product Models.

The result of implementation for the database part of a Concept-Oriented Product Model in an Object-Oriented database will result in classes of Data object such as shown in Figure 11. To the Relationship Data object and Concept Data object are respectively ascribed the two Attributes of

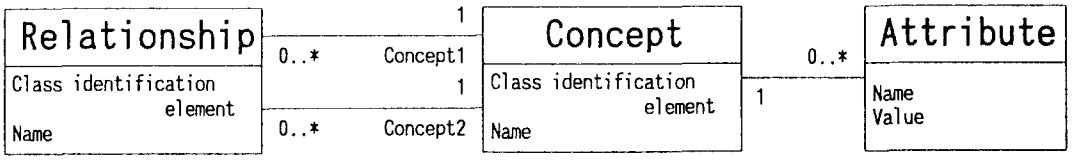

Figure 11: Classes of data object part of concept-relationship model

"Class identification element" and "Name":

- The "class identification element" serves to identify the kind of Concept. In the database, it constitutes the sole and unique element specifying the class to which Concept belongs. On the application side, it serves as a tag by means of which to identify upon retrieval from database the Behavior-governing Control object with which database the Activity-covering Control Object with which the Concept is to be linked.

- The "Name" serves to identify each individual Concept in a group of Concepts of a kind.

The Relationship Data object is linked to the Concept Data object through two roles "concept 1" and "concept 2". These roles serve to indicate the mutual relationships existing between Concept Data objects. Viewed from the Concept Data object side, a Concept will be linked to multiple Relationship Data objects.
The Attribute Data object possesses the two Attributes of "Name" and "Value":

- "Name" expresses the denomination of the Concept.

- "Value" expresses the objective worth of the Concept. If written in C++ language, it will be packaged in void* type, and converted to suitable alphanumerical symbols upon combination with Control object.

6 Implementation for the application part of a concept-relationship product model

The manner of implementation for the shipbuilding application part into database founded on Concept-Relationship Product Model is described in what follows.

6.1 Implementation with use of object-oriented language

In the shipbuilding Product Modeling system referred to earlier, applications were created covering the cases referred to in Chapter 4 of hull form fairing and curved surface patch defining, together with a third application covering an extremity treatment application for configuring the bow and stern full forms. The database part of this shipbuilding Product Modeling system was created by expressing the framework of Concept-Relationship Product Model as a class such as described in the preceding section 5.1, using Object-Oriented C++ language. For data management, the resulting Product Model was packaged into a marketed Object-Oriented database "ObjectStore". The three applications mentioned above were created one after another without trouble. The resulting application for curved surface patch definition presented the classes shown earlier in Figure 8.

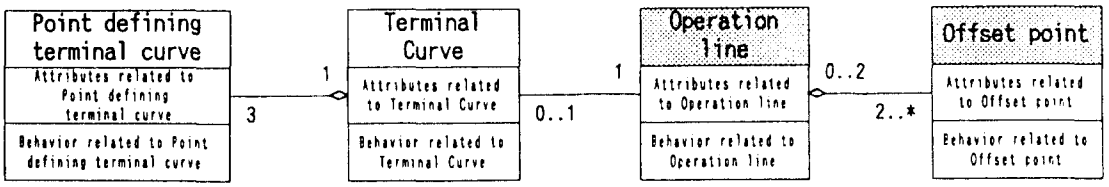

Figure 12: Classes for extremity treatment application

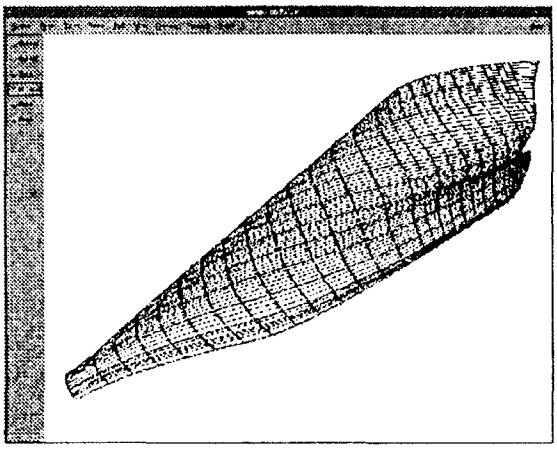

Figure 13: Curved surface patch defining application

The extremity treatment application yielded the structure of Figure 12, which reveals the necessity of two additional kinds of objects, i.e. a "Terminal curve" indicating the terminal configuration and "Point defining the terminal curve" indicating the points passed by the curve.

With conventional Object-Oriented Product Modeling, it would be necessary at this point to review the entire class, to study also the manner of utilizing in the newly added application the existing database created with full form fairing application, and based on the above to reexamine the overall application system.

With Concept-Oriented Product Modeling, all that was needed was to develop the information structure for covering the additional kinds of objects cited above that were indicated to be necessary. Development of the information structure to incorporate the additional objects proceeded without hitch under the care of different staff, to permit all three applications to operate on a common database (see Figure 13).

7 Merits and demerits of concept-relationship product modeling

Experience in applying the Concept-Oriented Product Modeling has revealed the following merits and demerits of the system.

7.1 Merits

- Information stored in extremely abstract form, which permits adaptation to a wide variety of information structures

- Easy adaptation to different approaches required for different Operations and Applications

- No need and pick out beforehand all relevant Operations and Applications, since and information models created or added as they become necessary

- Inputting the Conceptual structure of the ship as Data object of Concept-Relationship Product Model, to constitute the main design information structure, taking advantage of the little change of conceptual structure that is necessitated with progress of operations, which is a particularity of shipbuilding design work.

7.2 Demerits

- Redundant information models tending to be created by the highly abstract nature of information that is treated, and by the separate picking out of information for individual applications

- Effective database manipulation impaired by storage of Data objects separately in bits and pieces

- Clear image of the manner in which data are stored in database tending to become obscured with increasing number and variety of applications.

8 Concluding remarks

A method has been presented of a method of Product Modeling designated "Concept-Relationship Model", devised for complementing conventional product modeling, and examples of application in practice have been cited.

In order to assure practical application of the method and to extend the range of its applicability, it is intended to further develop the method through studies covering the following points.

(1) Functioning in a large-scale system
Application of the Product Modeling method has so far been limited to the upstream stages of design procedure, to constitute a relatively small-scale system covering a restricted range of information.
In remains to extend system application, for instance in space to cover individual components and parts, and in time to cover the downstream manufacturing stages, and this should call for ascertaining the extent of information requiring to be treated and the response characteristics to be expected.

(2) Applicability to network environment
Recent developments in communication networks have come to permit access of computer-aided design by a wide range of engineering work station, personal computes and the like,

and this is likely to raise demands for making the present system available to all such computer networks for reference and for data inputting.
Studies are to be directed toward devising a framework that will permit easy manipulation in a network environment, utilizing JAVA, Web and such other equipment and such standards as CORBA.

(3) Sharing system development know-how
The Concept-Relationship Product Modeling system is not far removed from the pattern technique of system development, which is coming to be widely utilized in Object-Oriented Product Modeling, and which permits utilizing advanced approaches to analysis and design procedures that are cataloged. The possibility is to be explored of rearranging the present system so as to permit mutual utilization and thus enhance overall development effectiveness and efficiency.

References

HATA, N., HAMADA, S., YAMASHITA, T. AND KONAKA, K. 1998 A concept-relationship modeling approach. J. of the Society of Naval Architects of Japan, **183**, pp. 463-472