15. An Object Oriented Tool for the Preliminary Design of Ships

Takao SEKIMOTO*, *Member*, Ken SHIMIZU**, *Member*, Takeo KOYAMA***, *Member*

(From J.S.N.A. Japan, Vol. 164, Dec. 1988)

Summary

To build a computer aided design system, it is necessary to develop a suitable model for representing design objects like a ship and design processes that determines the mechanism of redesigning strategy. In this research, design object is described as a combination of a date structure and a set of constraints which restricts the behavior of some related design parameters. And design process is represented as if–then rules which are expression of a skillfull engineer's knowhow of design.

The object–oriented representation form is employed to develop a computer aided ship basic designing system. In this system, design object, design parameter, constraint and designer are represented as "object". An object is a date structure with argorithms called method describing its own behavior or characteristic. Object–oriented representation is suitable for constructing design systems because : (1) Complicated date structures and their behavior can be easily represented, (2) Using the function of inheritance one can put many similar objects in order efficiently, (3) Message passing paradigm makes it easy to organize the whole system, (4) An object is independant and self–supporting so the system operating on the object–oriented environment has excellence in maintenance. We implemented constraint–riented system by defining constraints as objects.

In using this system, we can easily add or delete design parameters, constraints and if–then rules, so we are able to build or modify the model structure and knowledge–base flexibly. Finally, some application results of basic design of ships are presented to show the effectiveness of the system.

1. Introduction

The performance of the preliminary design of ships is still heavily dependant on the skill of experienced designers in spite of considerable amount of computerization. Today's CAE/CAD systems can do much with precise analysis and pretty drawings, but they are faulted for their inability to deal with the designer's intent and the knowledge hidden behind the data and drawings. The situation holds not only for ship design but also for engineering design in general. Although the application of expert systems to design is expected to bring breakthroughs, and though some efforts have been made, methodology and tools for developing practical expert systems for design have not been established yet[1),2),3),4),5),10)].

In this research, a design tool was developed and applied to the preliminary design of ships. The tool is a kind of expert system building tool and is intended to be used by a designer from the beginning. A designer is expected to describe his design knowledge during the design using the tool. In order to provide a suitable framework to represent the design knowledge, the knowledge was classified into categories, i.e. knowledge about the design object and knowledge about the design process, which are described in chapter 2.

More attention was paid to knowledge about the design object, because it requires a specific representation while knowlegde about the design process is considered rather easily representable using general and conventional if–

* Hitachi Ltd. (a graduate student of the University of Tokyo at the time of the research)

** Naval Ship Engineering Dept,. Mitsui Engineering and Shipbuilding Co., Ltd. (temporarily visiting the Center for Design Research, Stanford University)

*** Professor, Dept. of Naval Architectute & Ocean Engineering, The University of Tokyo

then rules. A set of design parameters and a set of constraints determine the structure and behavior of a design objbct in the developed tool. A constraint oriented approach, in which constraints play an important role in their own evaluation and propagation, was taken in this research[5),11),12)]. The implementation was made relatively easily be adopting object oriented programming introduced in chapter 3[7),8)]. The configuration and functions of the developed tool are described in chapter 4.

The tool was applied to the preliminary design of ships and proved to be useful as described in chapter 5.

2. Classification of Design Knowledge

In order to build a sophisticated computer aided design system, designers' unstructured design knowledge should be organized and represented clearly on a computer system. Classification of design knowledge is helpful in organizing design knowledge and developing knowledge representation for a design tool. In this research, design knowledge was classified into two categories, i.e. knowledge about the design object and knowledge about the design process, as described in the following sections. Requirements for a design tool to deal with each category of knowledge are also discussed below.

2.1 Knowledge about the Design Object

It is necessary in design to clarify what design parameters the design object has and what relations exist among the parameters. These two design elements, i.e. design parameters and relations, are very important as design knowledge since they determine the structure and behavior of the design object.

The value of each design parameter is restricted by its relations with other parameters. Those relations, or constraints, are imposed as physical laws, legal regulations, specification requirements, etc. Intricately coupled constraints make it difficult to get the optimal or a satisfactory set of the design parameter values. Therefore, the role and behavior of constraints in the design problem should be well understood. The following two aspects of constraints are considered to be important for developing a design tool.

(A) status evaluation

During the design, a designer decides what to do by examining the constraints to see they are satisfied or violated. In order to help the activity of the designer, a constraint should be re-evaluated automatically whenever any related parameters are changed.

(B) multi-directional propagation

A designer not only examines the constraints but also uses them to determine the satisfactory values of the design parameters. A constraint does not define a rigid procedure to determine a related design parameter, but declares the relations among paramters. Therefore, a design tool should support the function of multi-directional propagation. For example, an equation with n parameters should be able to be used to calculate the value of any one parameter, irrespective of whether it is in the left hand side or the right hand side, when the rest of $n-1$ parameters are determined.

Constraint oriented programming[5),11),12)], in whic each constraint is represented declaratively, is considered to be a promising approach to realizing a design tool with the functions described above. The approach was taken in this research.

The design parameters and the constraints should be able to be added, modified or deleted easily during work with a design tool, because the structure of the design object is gradually defined through the design process by stepwise refinement.

2.2 Knowledge about the Design Process

Knowledge about the design object stated in 22.1 is concerned with the static, analytical and local status of the design object. There is another kind of design knowledge: dynamic, strategic and global knowledge about the design process, such as which parameter should be set first when there are many to be set,

which parameter should be changed in case of coupled constraints violations, etc. Knowledge about the design process represented as if–then rules[4] is expected to be albe to drive the design cycle of generation, evaluation and modification[6].

3. Object Oriented Programming[7),8)]

In this research, smalltalk–80, one of the most common object oriented programming languages, was used to develop a design tool. The basic features of object oriented programming with some respects to design problems are described in this chapter.

An *object* in object oriented programming is an entity which contains structured data and associated procedures. All kinds of data, such as numbers, lists and booleans, are dealt with as objects. A programmer can define a new object with a suitable data structure for his purposes. Since the data contained in an object are also objects, a complex data structure can be handled hierarchically to any depth.

Procedures contained in objects are called *methods*. A method of an object is invoked by a *message* sent to the object. Thus, in object oriented programming, execution of a program is realized by series of message passing events among a number of objects. Because the data and the methods are encapsulated in an object and message passing is the only way to access them, each object has very high modularity. This programming style helps in developing a system by describing the local relations declaratively when the overall procedure is difficult to recognize as in design problems. Akagi and Fujita handled the network of the basic design parameters of a ship, such as the length, the breadth and the draft, with object oriented programming successfully[9),10)].

Data structures and methods are defined in abstract objects called *classes*, which are a kind of prototype of each *instance*. Classes are organized in a tree–like hierarchy. A subclass object *inherits* the data structures and the methods of its superclasses in the hierarchy. A programmer can define a superclass by extracting the common characteristics and behaviors among similar objects, and then he only has to define specific things in each subclass, thus the programming efficiency and the maintainability of the program are increased. The class hierarchy also helps to organize the concepts in the problem, which is important in the design problem.

4. Configuration and Functions of the Design Tool

4.1 Overview

The configuration of the object oriented design tool developed in this research is outlined in Fig. 1. A "design object" and a "designer", corresponding to knowledge about the design object and knowledge about the design process respectively, are the major objects in the tool. The design proceeds through message passing between the two objects. The structure and the behavior of each class of objects are described in 4.2.

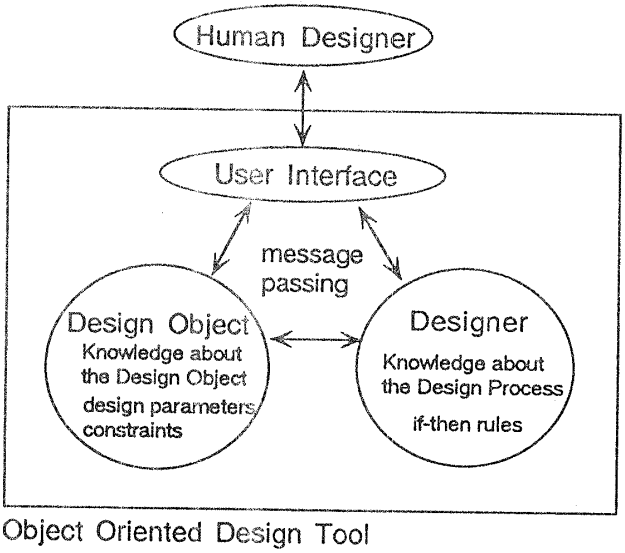

Fig. 1 The Configuration of the Design Tool

The interface for accessing the "design obgect" and the "designer" is very important for a human designer. The user interface for the tool is described in 4.3.

4.2 Principal Objects

4.2.1 Design Object

(1) Configuration

A class "Design Object" was defined to

represent knowledge about the design object. A "design object" contains a set of design parameters and a set of constraints as illustrated in Fig. 2. The relations among the design parameters are defined by the constraints. Each of the design parameters has pointers to its related consitraints and each of the costraints has poninters to its related design parameters. A set object is a king of list with variable size, and elements can be added and removed freely. The characteristics of the element objects of the two sets are further described in (2) and (3).

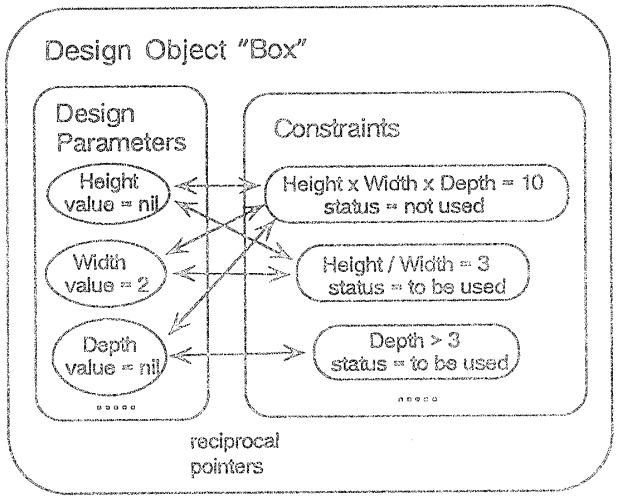

Fig. 2 An Example of a Design Object

(2) Design Parameters

A "design parameter" has a value and a list of pointers to related constraints. A variety of data, such as number, a string, a truth value or structured data consisting of those primitive design parameters, should be handled as the value of a design parameter, though principly numeric values were considered in the prototype system developed in this research.

Any change of a value is made only by sending an appropriate message to the design parameter object, and the method invoked by the message not only changes the value but also sends re-evaludtion messages to its related constraints using the list of pointers, which guarantees the consistency of the status of the constraints with the values of the design parameters. (Fig. 3)

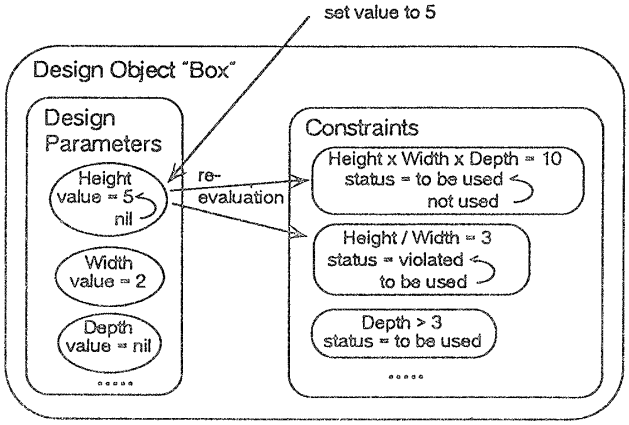

Fig. 3 Setting the Value of a Design Parameter

(3) Constraints

A "constraint" contains the body of the constraint, its status, a list of pointers to the related design parameters and design advice.

Equalities and inequalities consisting of addition, subtraction, multiplication and division were mainly considered as constraints in this research. More general functions in the form of y=f(x1, x2b ..., xn) can also be handled with the developed tool though they cannot be propagated multi-directionaly as described later.

A constraint expression, input as a string by a user, is parsed and decomposed into a set of primitive constraint objects; these consitute the body of the constraint. (Fig. 4) At the same time, a list of pointers to related parameters is generated, and a pointer to the constraint itself is added to the constraint list of each of its constituent parameters, to maintain the system's consistency. New design parameters are generated if the design parameters referred to by the constraint do not exist yet.

Each constraint has its self-evaluation method which is invoked by a message from a

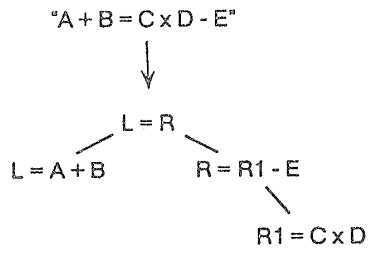

Fig. 4 Decomposition of a Constraint

related design parameter as described in (2). The result of an evaluation is assigned to the status of the constraint. The status of a constraint is one of the four :

"statisfied" —— The constraint is satisfied.

"violated" —— The constraint is violated.

"to be used" —— The constraint cannot be evaluated because of an unassigned design parameter, but can determine the value of the unassigned design parameter to satisfy the constraint.

"not used" —— The constraint cannot be evaluated or propagated because of too many unassigned design parameters.

An example of the status of a constraint is shown in Table 1.

Table 1 Status of a Constraint

Values of Design Parameters					Status of the Constraint "A + B = C x D - E"
A	B	C	D	E	
nil	nil	nil	nil	nil	not used
1	2	4	nil	nil	not used
1	2	4	5	nil	to be used
nil	2	4	5	6	to be used
1	2	4	5	17	satisfied
1	2	4	5	6	violated

In order to deal with loose constraints, a user can define the "margin" for each constraint specifying the difference to be allowed between the left hand side and the right hand side.

The propagation of "to be used" constraints is multi–directional ; e.g. a constraint "Area= Length* Width" can determine the Area from the Length and the Width, and can also determine the Length from the Area and the Width. In order to realize the function, each of the primitive constraints, e.g. adder, has a propagation method as in the example below.

```
propagate
  if status="to be used"
    if a=nil
      a:=b+c
    else if b=nil
      b:=a−c
    else if c=nil
      c:=a−b
  end if
```

Once those primitive methods are prepared, a complicated formula, being decomposed into a set of primitive constraints, can propagate to any of the related parameters depending on which is unassigned. A sequence of message passing which carries out constraint propagation is illustrated in Fig. 5.

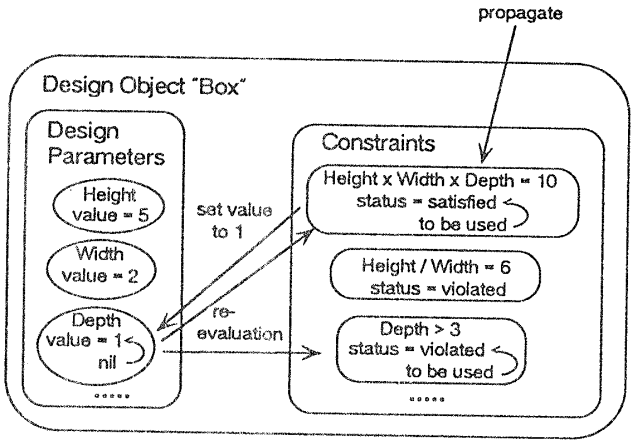

Fig. 5 Constraint Propagation

A user can specify one of the related design parameters as "design advice" for the constraint. When a constraint is violated during the propagation process, the design parameter specified by the design advice is recalculated automatically to clear the violation. This function is helpful in bandling a group of coupled constraints in which the proper recalculation order in case of violations is known.

4.2.2 Designer

A class "Designer" was defined to represent knowledge about the design process. The object contains a knowledge base consisting of if–then rules as an instance variable and inference mechanisms as methods. The rules represent heuristic knowledge of human designers in the form of "If 〈the status of design object〉, then 〈design/redesign action〉". Modularity of the rules makes it easy to generate, modify and maintain the knowledge base[4]. HUMBLE[13], a rule based system tool in smalltalk–80, was used in the designer object. HUMBLE performs backward reasoning and can deal with uncer-

tainty using so called certainty factors.

A designer object is expected to replace the role of a human designer by using if–then rules in the knowledge base. It controls the design process of generating and modifying the design artifact model represented by the objects described in 4.2.1. It might be considered to be possible to simulate the cooperative design process of a group of designers in the real world by instantiating several designer objects with different knowledge bases.

4.3 User Interface

User interface is very important for a design tool because design is essentially an interactive activity between a human designer and a design artifact model. A designer decides what to do by inspecting the temporary design artifact model, and the result of his action on the model is fed back to him to stimulate his thinking about the next action. In this research, smalltalk–80 provided a good environment to develop a user friendly system with its built–in window and menu management system.

Fig. 6 shows a "design browser" with which a user can generate, inspect and modify design objects. The three panes is the upper part of the window display the names and categories of the design objects. The lower left panes display the names and values of the design parameters. Each of the other four panes displays the constraints in the corresponding status (satisfied, violated, not–used and to–be–used, clockwise from the top left). The figure shows the situation just after generating a design object. It does not have any design parameters or constraints yet.

Each pane of the windows has its own pop–up, menu, such as add, delete, inspect, etc. In Fig. 6, for example, a user can select an "add constraint" menu in any of the four constraint panes. Fig. 7 shows the window after adding a constraint by typing "L*B*D=V" for the prompt of the "add constraint" menu. It can be seen that four design parameters and one constraint were generated. The constraint's status is "not used" because all of the four related design parameters have no values yet. A user can give values to the design parameters using the "set value" menu in the design

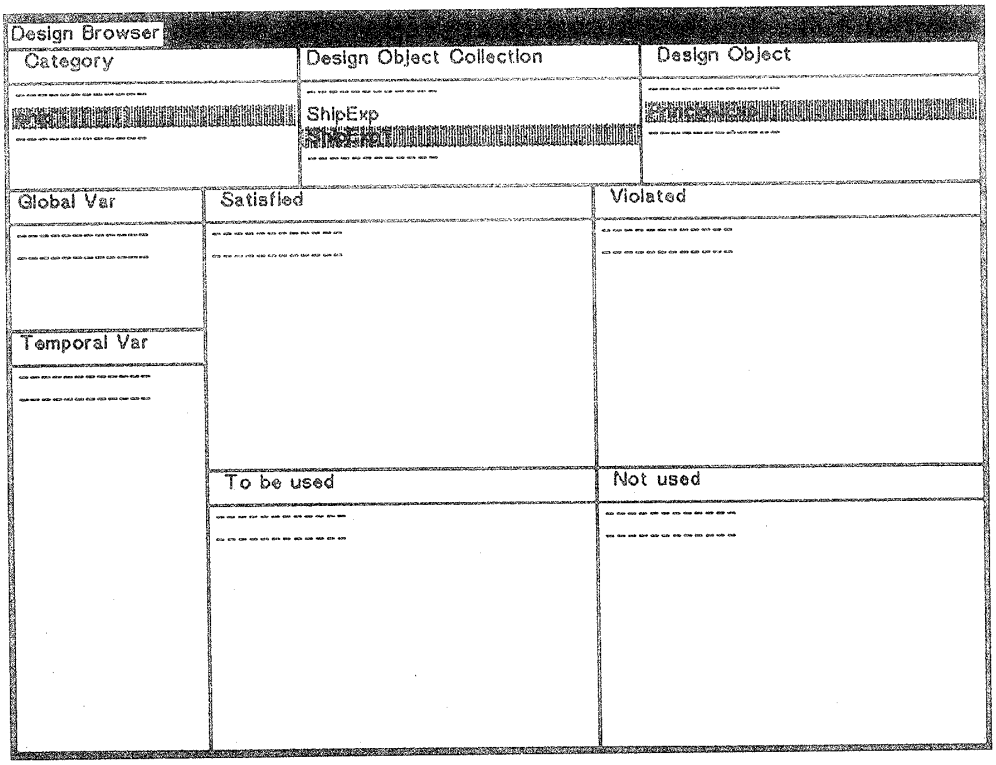

Fig. 6 Design Browser

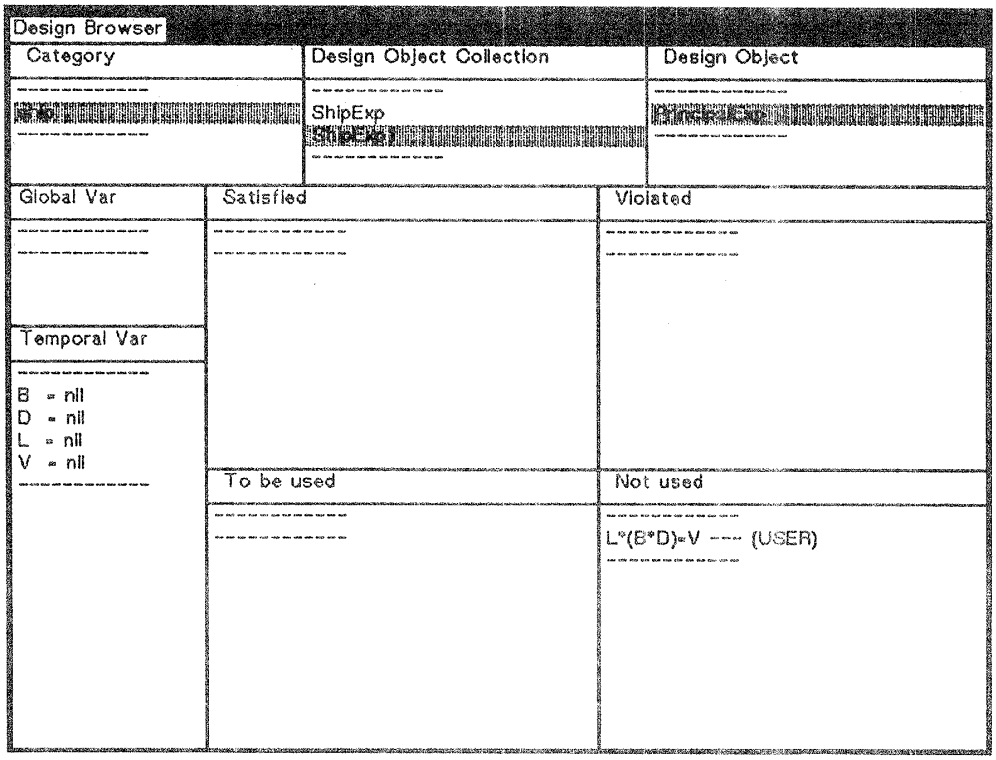

Fig. 7 Adding a New Constraint

parameter pane. Constraints move to the appropriate pane, e.g. the to-be-used pane in the bottom left, when their status are changed as a result of a change of a design parameter value. The four pane system of constraints visually helps a designer to recognize the present status of the design object. A user can also control the propagation and satisfaction of constraints. By manipulating the design parameters and the constraints as described above, a user can represent his knowledge about the design object interactively.

The tool can be used without any designer objects. In fact, in the first use of the tool in a given field, it is appropriate to do the design without designer objects ; a human designer is better. During the design process conducted by a human designer, some patterns of design actions (e.g. the way a violated constraint is satisfied) may be found. Then, this knowledge about the design process can be written down as if-then rules using the rule editor shown in Fig. 8. Knowledge base can gradually grow up in this way. A designer object with the constructed knowledge base accesses the design object to perform the design when "HUMBLE" is selected in the pop up menu in the design browser.

5. Application

The design tool developed in this research was appled to the preliminary design of ships. First, the design parajeters and constraints in ship design were recognized and represented using the tool. Second, knowledge about the design process was extracted as if-then rules while observing the design process. After those two steps, a similar design with different requirements was performed automatically by the tool.

5.1 Design Objects

During the definition of the design parameters and the constraints, it was found that there were too many design parameters and constraints in preliminary ship design to handle at a time. Several design objects were defined to deal with a ship by meaningfully decoposing it. The major four objects were principal particulars, powering, midship sectiond and

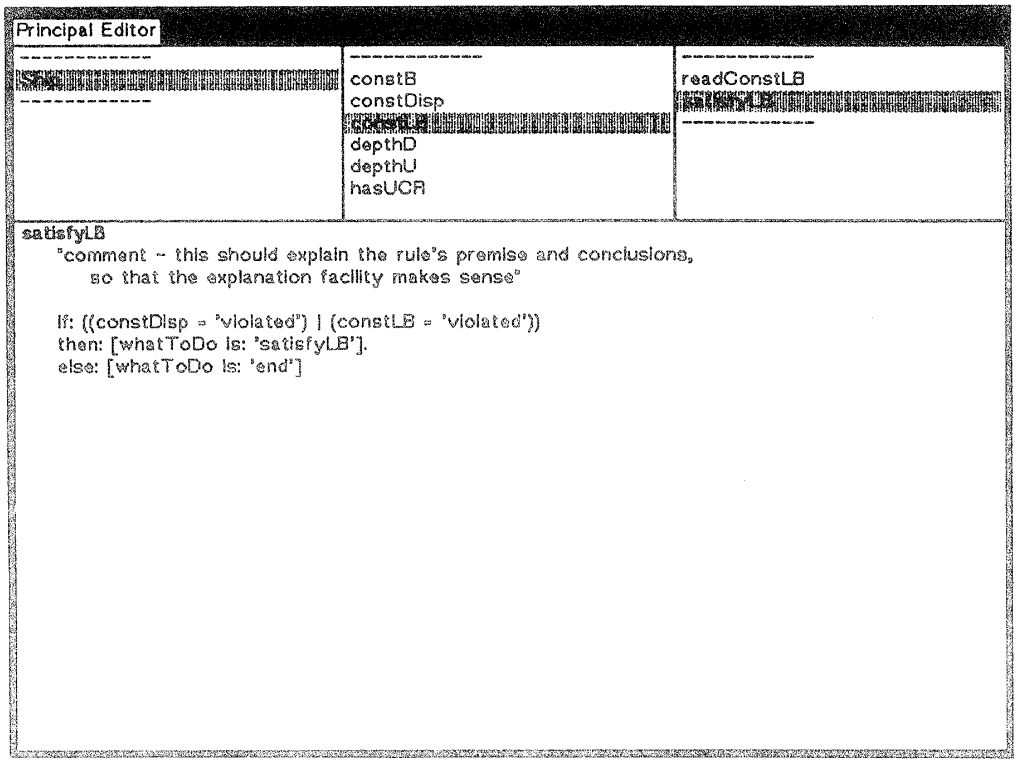

Fig. 8 Rule Editor

compartments.

The principal particulars contained the principal dimensions and the characteristic attributes of a ship (e.g. L : length, B : breadth, d : draft, Cb : block coefficient) as the design parameters and the relations among them (e.g. "L*B*d*Cb* 1.025=displacement", "displacement=deadweight+lightweight") as the constraints. Loose "statistic constraints" were also defined, such as an empirical estimation formula of the hull steel weight taking L, B and D as parameters. The powering contained the estimation formulae of the resistance coefficients, the formulae to calculate required horse power, etc. The midship section represent the forms and the arrangement of the shell, the decks and the longitudinal bulkheads in the transverse section of the ship. The compartments contained the positions of the bulkheads, formula to calculate the longitudinal moment of the cargo in a compartment considering the weight and the center of gravity, the inequalities defining the range of the trim allowed by legal regulations, etc. A function to maintain the consistency among the duplicated data in shose four objects were implemented in the tool.

5.2 Design Process

In the beginning of the design, most of the design parameters had no values and most of the constraints had not-used status. A human designer assigned values to some of the design parameters at this point. Then, some of the constraints changed their status to "to be used". In this phase, the human designer assigned values to other design parameters by controlling the propagations of the to-be-used constraints as well as by setting values explicitly. Knowledge about the constraint propagation control, such as which constraint to choose when there are many to-be-used constraints, was captured and represented as if-then rules. Though the propagated constraints not satisfied, some of the constraints became violated while the number of assigned design parameters increased. Fig. 9 exemplifies the situation. In this case, the displacement calculated from the hull form is conflicting with the sum of the estimated lightweight and the required

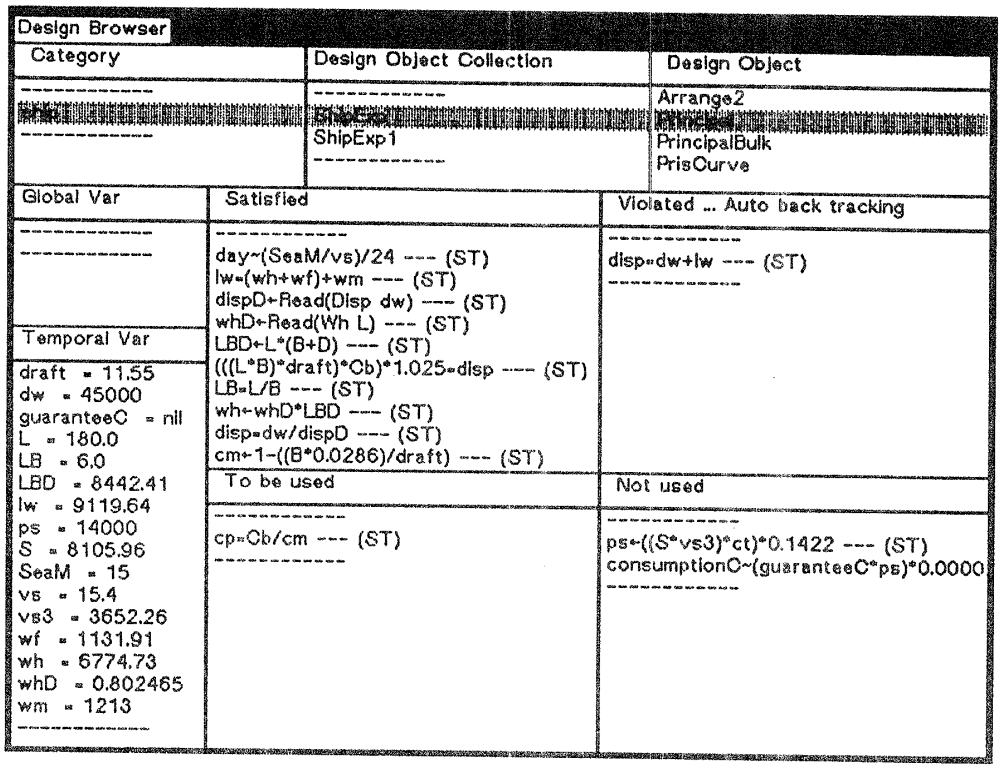

Fig. 9 An Example of Constraint Violations

deadweight. The human designer satisfied the violated constraint by resetting the displacement. Some of the simple patterns to reset the design parameters were stored as "design advice" for the constraints, and the other patterns were represented as if–then rules.

In the application example, several designer objects, such as principal particulars designer and a compartments designer, were instantiated reflecting the fact that several design objects were deflned to represent a ship.

For example, the compartments designer had a rule shown below:

If (the trim requirement is violated),
then (change the position of the collision bulkhead).

Fig. 10 shows an example of the design by a designer object. The upper window is the monitoring window of the designer object which is deciding what to do for the design object. The status of the design object. The status of the design object is shown in the design browser window below in the figure.

The design finished when all of the design parameters had a value and none of the constraints stayed on the violated status list. After several runs of the design, during which if–then rules were captured, it became possible to perform the whole process of design with the designer objects. At this stage, the human designer only had to input design specifications such as required deadweight or service speed.

6. Conclusions

A research project to develop a sophisticated computer tool for the preliminary design of ships has been carried out.

It was recognized that there are two kinds of design knowledge, i.e. knowledge about the design object and knowledge about the design process. A general purpose design tool to deal with the both kinds of design knowledge was developed with objebct oriented programming. The features of the design tool are as follows.

(1) A "design object" and "designer", corresponding to knowledge about the design object and knowledge about the design process respectively, were defined as the major objects in the tool. The design proceeds through message passing between the two

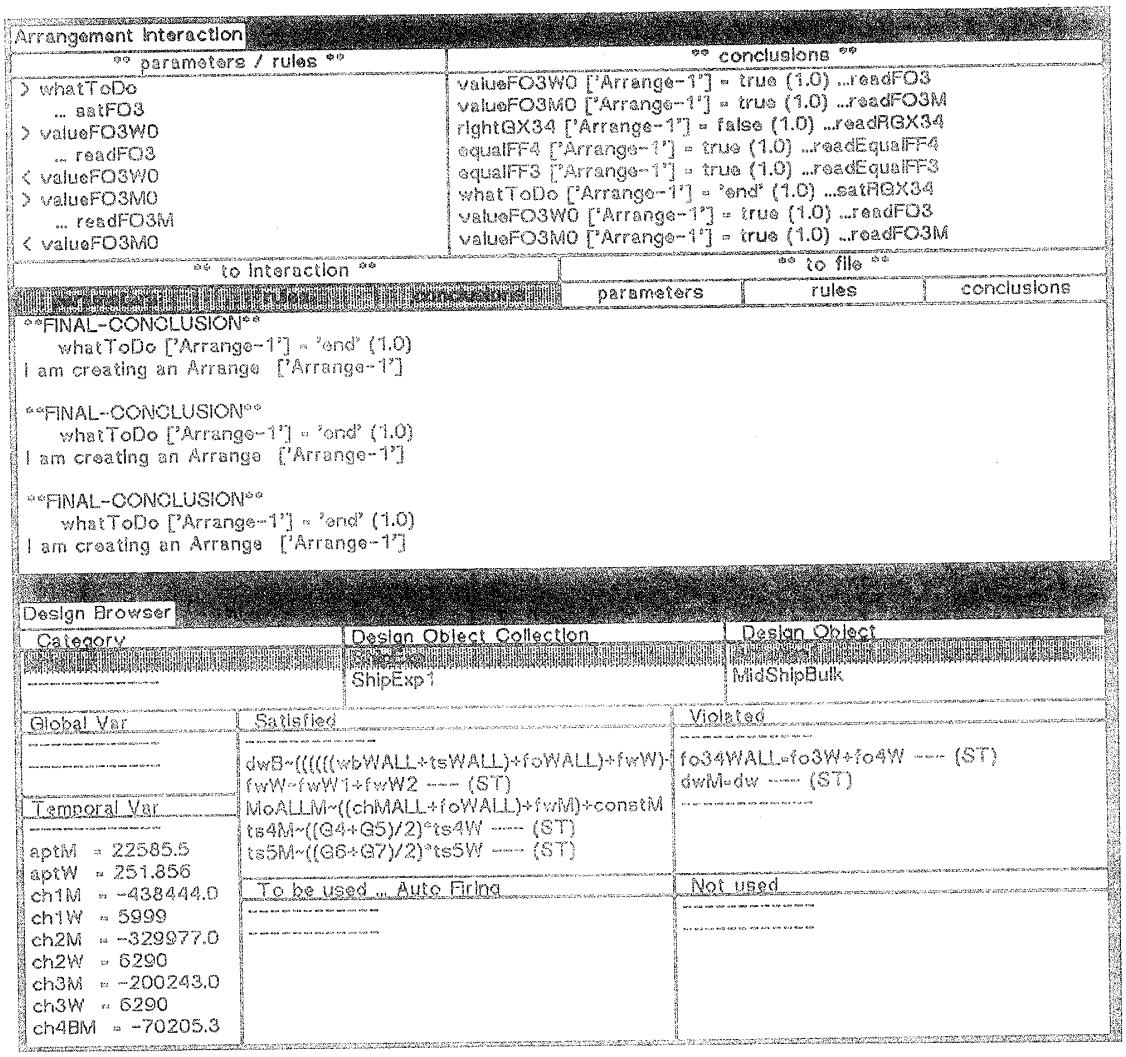

Fig. 10 Design by a Designer Object

objects.

(2) A design object consists of set of design parameters and a set of constraints, which can be added, modified and removed freely during the design. Any change of a design parameter is posted to its related constraints by message passing to re-evaluate them: thus, the consistencies between the design parameters and the constraints are guaranteed.

(3) Each constraint is represented as formula, which is easy to handle for a human designer, and has functions not only to evaluate itself but also to propagate itself to determine the value of its related design parameters multi-derectionally.

(4) A designer object, which has a knowledge base consisting of if-then rules representing a human designer's know-how, can control the design process.

(5) User interface with multiple windows and pop up menus helps a human designer to carry out the design, which is an interactive activity by nature.

The tool was applied to the preliminary design of ships. First, the design parameters and the constraints in ship design were recognized and represented using the tool. Second, knowledge about the design process was extracted as if-then rules while observing the design process. The freedom to add/modify/remove the elements, the constraints' functions of automatic re-evaluation and multi-direc-

tional propagation, and the friendly user interface were very helpful for a human designer. After those two steps, a similar design with different requirements was performed automatically by the tool.

Some points were recognized as the topics for further research. It was found that a single design object became difficult to deal with when the number of design parameters and constraints increased ; it is desirable to allow a design object to be constructed as a part-whole hierarchy. In order to represent the forms and the spatial arrangements of design objects, not only arithmetic constraints but also geometric constraints should be implemented in the tool.

Acknowledgement

The authors gratefully acknowledge the support of the Shipbuilding Research Association of Japan and the Systems Technologies Committee of the Society of Naval Architects of Japan.

Reference

1) Weiss, S. M., Kulikowski, C. A., A Practical Guide to Designing Expert Systems, Rowman & Allanheld, 1984.
2) Shimizu, K., "An Expert System for Subdivision Design of Tanker", the Journal of the Society of Naval Architects of Japan, vol. 164, 1987.
3) Kaneko, H., "An expert system for the design of merchant ships", Master's Thesis, The University of Tokyo, 1986.
4) Hayes-Roth, F., Waterman, D. A., Lenat, D. B. (ed.), Building Expert Systems, Addison-Wesley, 1983.
5) "Mechanical design systems leap to Intelligent CAD", special issue on Nikkei AI, Nikkei McGraw-Hill, 1987.
6) Kasahara, T., "Computerization and Systemization in Basic Planning", Systems Technologies in Shipbuilding, the systems technologies committee of the Society of Naval Architects of Japan, 1984.
7) Suzuki, N., Object Orient, Kyoritsu shuppan, 1987.
8) Umemura, K., Introduction to Smalltalk-80, Science-sha, 1987.
9) MacCallum, K. J., "Understanding Relationships in Marine Systems Design", Proc. of First IMSDC, 1982.
10) Akagi, S., Fujita, K., "Building an Expert System for the Basic Design of Ships", Kansai Zousen Kyoukai-shi, vol. 206, 1987.
11) Borning, A., "The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory", ACM Trans. on Programming Language and Systems, Vol. 3, No. 4, 1981.
12) Sussman, G. J., Steel, G. L. Jr., "CONSTRAINTS-A Languate for Expressing Almost-Hierarchical Descriptions", Artificial Intelligence, Vol. 14, 1980.
13) HUMBLE User's Guide, Fuji Xerox, 1987.