A Novel Ship Subdivision Method and its Application in Constraint Management of Ship Layout Design

Deniz de Koningh, SARC B.V., Bussum / The Netherlands, ddekoningh@gmail.com.
Herbert Koelman, SARC B.V., Bussum / The Netherlands, h.j.koelman@sarc.nl.
Hans Hopman, Delft University of Technology / The Netherlands, j.j.hopman@tudelft.nl

Abstract

Conventionally, the rooms and spaces of a ship are either modelled as volumetric entities, or with the aid of bulkheads and decks. According to our knowledge, no simple representation exists where both entities can be modelled independently, and where automated conversion from one view (volumetric) to the other (planes) is possible. This paper introduces a simple yet effective approach, where a ship designer can mix the use of volumes and planes in any fashion. Furthermore, this modelling method is applied in a novel tool to manage ship subdivision constraints. As quite some numerical constraints are known a priori, they can be defined in a list, and assigned to specific subdivision elements. Examples are bulkhead locations or required tank volumes or deck areas. A constraint management tool is developed which evaluates the ship layout design during the design process. The designer will be able to modify or add constraints and the tool will support the designer by managing these constraints during the design process. If the hull form changes, all submitted rules will be updated according to the new main particulars. If one of the constraints does not comply, an adjustment or alternative can be chosen at that moment and the impact of this change is directly visible. The designer can also ask the tool to provide a ship layout design that complies best with the constraints entered. When the Constraint Management program is used, a feasible ship compartment design can be made in a quick manner and the designer is kept from making errors. This means that a correct ship layout model is available on which probabilistic damage stability calculations and weight estimations can be performed in an early stage. This method has been implemented in a computer program, so actual design examples will be discussed.

1. Introduction

In the period 2008 – 2011, in the Netherlands a joint-industry research project, baptised *Innovero* (see http://www.marin.nl/web/JIPs-Networks/Public/INNOVERO-1.htm), was commenced which aimed at improving the ship design process, and thus decreasing the time required to produce a feasible ship design. Summarised, an infrastructure is developed in this project where a confederation of software agents, conducted by a knowledge management tool (in particular Quaestor, e.g. *van Hees (2009)*, jointly tackle the various design tasks.

An important demand in *Innovero* is that the range of applicable software agents not necessarily has to be limited to dedicated naval architectural analysis software, but that also general purpose CAD-systems, such as Rhinoceros, Eagle or Autocad, should be considered. The consequences of this requirement are too wide to be discussed in full depth in a single conference paper; instead, there is one subject we will focus on, which is the representation and utilisation of the internal geometry of the ship. The reason is that conventional methods to represent the internal geometry (rooms, spaces, compartments, holds) have been developed in the past for naval architectural analysis tasks, such as the calculation of tank calibrations tables or damage stability. Because these representations are not necessarily the most suitable to be used in the design stage, e.g. with a general purpose CAD-system, they had to be reconsidered for *Innovero* purposes.

This reconsideration, and subsequently re-design, of representations for internal geometry is one of the two main subjects of this paper. The second one is the design and functionality of a tool that utilises this representation in order to guide the ship designer in complying with a number of geometrical constraints.

2. The ship's internal shape representation

2.1 Design requirements for the internal shape representation

In the *Innovero* project, an inventory of requirements for a representation method was made. Most of the requirements are more or less related to the implementation, such as:

- Compatible with (or convertible into a format compatible with) the applied naval architectural analysis software, which is PIAS, by company SARC, of the Netherlands. If representation conversion is required, it should not lead to over-detailed models, because they could hamper calculation efficiency in case of lengthy calculations, such as probabilistic damage stability.
- Ready to be applied in combination with various ship hull representations, e.g. a surface model, a solid model, or a wireframe model. The latter may even be rather sparse, if the hull is only defined upon cross sections.
- The applied method, and its underlying entities, should be sufficiently easy to understand, and comprehensive, in order to be utilised with macros or scripts of general-purpose CAD software.

However, the most important requirement was not related to software as such, but was included in order to assist the ship designer as much as possible. The issue is that for some design or definition tasks it is pleasant, for a human being, if the focus is on spaces, while for other tasks a plane-based focus is more desired. And with 'focus' the basic modelling entity is addressed, so, summarised, one time it is easier to model a compartment directly by its boundaries, and in other cases modelling by means of bulkheads and decks is more appropriate. Obviously, planes and spaces are interrelated, so we need to address their duality.

2.2 Methods for the representation of internal geometry

Representation methods, as applied in today's popular naval architectural software tools, have been investigated in *Lee et al. (2003, 2009)*, where it is concluded that they fall apart in two categories:

- Those where spaces are defined by their boundaries. Such a method is not suitable for our purposes, because they lack an explicit definition of, or reference to, realistic planes[1], so the duality between spaces and planes is not addressed.
- Those where a wireframe model of compartment boundaries is applied. Although in this case some relationship between spaces and planes does exist, their reverse relationship does not, so the duality is not fully addressed.

As a solution, Lee et al. proposed a data structure based on a non-manifold solid model, Fig. 1. This class of solid models, manifold and non-manifold, are discussed into depth in e.g. *Mäntylä (1988)*, but summarised, a non-manifold model can describe an object with only an outer boundary, while with a non-manifold representation also an object with an internal structure can be modelled. Such a 'solid model' may seem rather abstract, but in e.g. *Koelman (2003)* it is shown that practical applications may certainly be built upon them. Nevertheless, these representation methods are rather complex, where the non-manifold method is an order of magnitude more complex than the manifold version. Because *simplicity* was expressed in sub-section 2.1 as an important requirement, this category of methods was considered to be less suitable for our problem.

Another interesting paper is *Alonso (2008),* where many implementation details are unfortunately not discussed, but from which it is apparent that the described software system allows for multiple repre-

[1] By *a realistic plane* we mean a plane, or part of it, that does really exist in a ship. It will be obvious that such a plane does not necessarily has to extend over the full intersection between the plane and the ship hull.

sentation methods. The conceptually simplest method is a compartment bounded by six planes[2], but there is also another method based on, quote, a ‘successive split of an initial space’, with a further detailing: ‘from the first level compartment definition the system allows the iterative subdivision of the created compartments by user-defined planes. This process can be repeated as many times as needed to obtain the desired detail of subdivision’. Certainly an interesting idea.

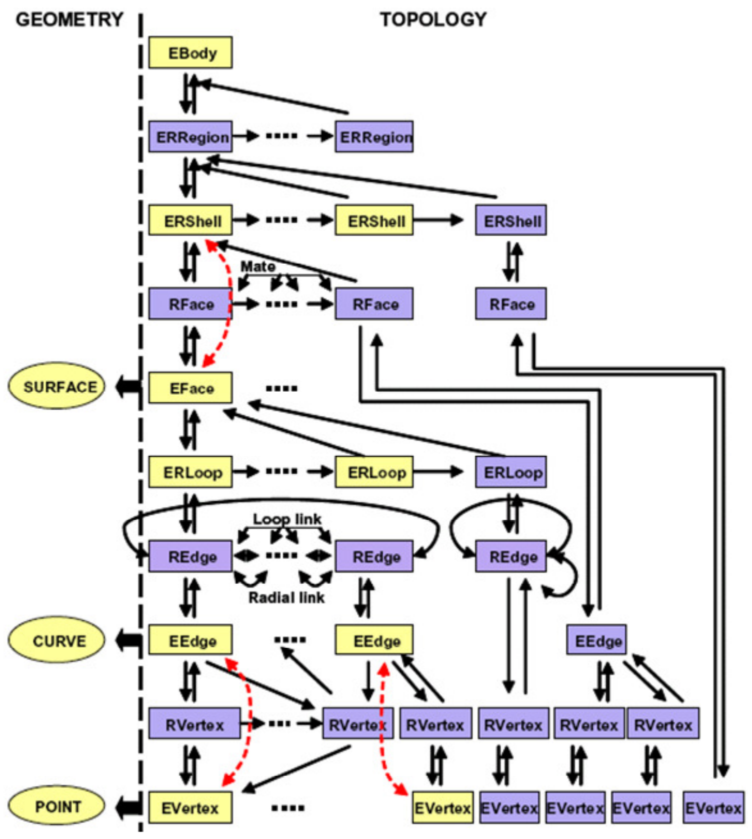


Fig.1: Data structure of the non-manifold solid representation of *Lee et al. (2009)*

2.3 The BSP method

Starting from the most important design requirement of sub-section 2.1, the capability to tackle the duality between spaces and planes, we may conclude that the *spaces* part can be fulfilled by conventional methods. However, for the modelling of planes, *realistic planes,* the question arises what is the *handiest* method to represent and define those planes. And with *handiest* we mean which method intrinsically fits to the ‘logic’, the intuition and the expectation of the average ship designer. In discussions with ship designers and a designer’s panel, held back in 2006, it became apparent that the ‘space splitting idea’ is considered to be rather intuitive; with this method an empty hull form is split in two by a plane, those two resulting spaces are subsequently split in two by other planes etc. etc., until the subdivision is obtained. If we look beyond its particular fields of implementation, this ‘space splitting idea’ is similar to the well-known *Binary Space Partitioning* (BSP) method, where a space is recursively split in two, resulting in closed cells, in which we see the ship’s compartments. The BSP method (see http://en.wikipedia.org/wiki/Binary_space_partitioning for an introduction) originates from interactive computer games[3], but has also been used for modelling purposes.

[2] Quote: ‘In most of the cases, the compartments are limited by six surfaces oriented according to the principal geographical directions of the ship. The aft and the fore compartment limits on the x axis, the port and starboard limits on the y axis and the lower and upper limits on the z axis’

[3] In particular boys games, such as shooter games and football games.

Conventionally, the recursive subdivision is represented by a binary tree, which is also a suitable internal representation in a computer program. An example of a BSP-application in a plane is given in Fig. 2, where the shaded 2D figure on the right is recursively split by the planes *a* through *f* (which form the nodes of the tree at the right side of the figure) and the cells 1 through 7 (which form the leaves of the tree).

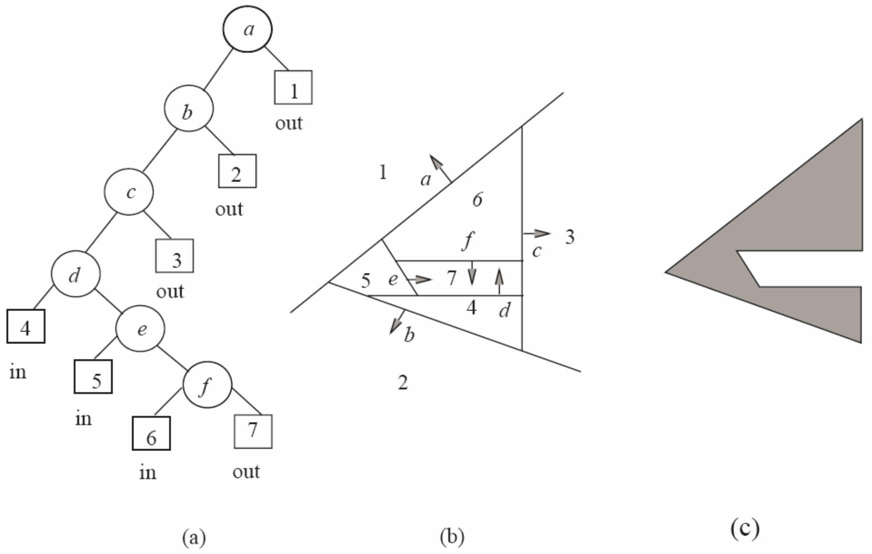


Fig. 2: Two-dimensional example of the BSP tree; from *Comba and Naylor (1996)*

Examples, and other applications of the BSP, can be found in, amongst others:

- General descriptions of the BSP, its properties and a number of basic algorithms are presented in e.g. *Thibault (1987), de Berg et al. (1998), Schneider and Eberly (2003), Naylor (1998), van den Bergen (1999).*
- Conversion of a B-rep solid model to BSP, discussed in *Thibault and Naylor (1987), Naylor (1992), Jiang (1996), Schneider and Eberly (2003), Ghali (2008).*
- An integration between B-rep and BSP is proposed in *Vanecek (1991).*
- Conversion of BSP to a B-rep solid model is proposed in *Buchele and Roles (2001),* for objects with curved boundaries, and in *Thibault and Naylor (1987)* and *Comba and Naylor (1996)* for polyhedra.
- Boolean operations with BSP's in *Thibault and Naylor (1987)* and *Naylor et al. (1990).*

2.4 The BSP method, applied to internal ship modelling

We have concluded that the BSP approach may fit the requirements of sub-section 2.1; it is capable of representing planes as well as volumes, supports the intuitive method of 'space splitting', is conceptually simple to understand, while the referred references of the previous sub-section provide sufficient tools to convert from and to other representations. By the way, the phrases of *Alonso (2008)*, as cited in sub-section 2.2, suggest that their method is somewhat similar to the BSP approach, which gives the comforting feeling that we are not the only ones sailing the uncharted waters of the application of a 'space splitting' method in ship compartmentation.

However, a native BSP representation would not always be the best entity to present the ship layout to the ship designer, for the reason that in a BSP tree a compartment or plane may be subdivided in many smaller sub-compartments or sub-planes, which hampers the grand overview of the design. For that reason, a data structure was designed where the program user, the ship designer, is working with the following familiar entities:

- The ‘compartment’, which is an enclosed space within the ship.
- The ‘physical plane’, which is a realistic plane within the ship, so a bulkhead or a deck. The physical plane may be bounded, which means it does not extend over the entire space in the ship hull.
- The reference plane, which is a virtual and unbounded plane, only intended to speed up modelling and modification action.

The BSP forms the glue between those three entities, and is not available to the program user as a separate entity. This data structure is depicted in some more detail in Fig. 3.

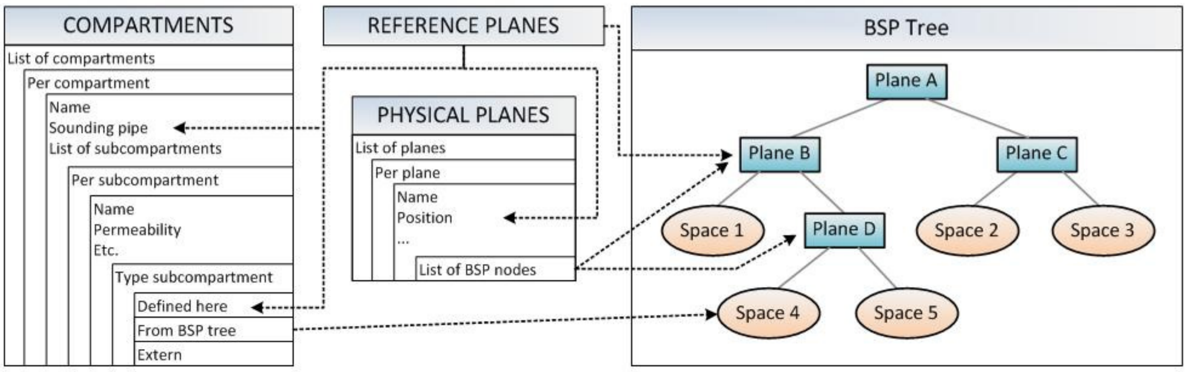


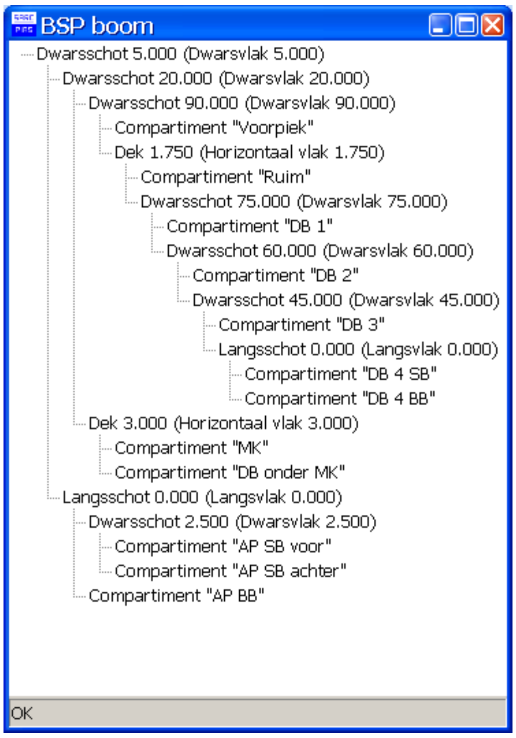
Fig 3: BSP tree as central representation for the compartment design

2.5 The computer program

Based on the described data structure and tools a computer program was produced. The program, which is a module of the well-known PIAS suite, is characterised by the following properties:

- If it has two faces; it can either work as a stand-alone design tool with a *Graphical User Interface* (GUI), or it can act as a server, to serve other processes (or other agents). In the latter fashion it can e.g. provide properties of the internal ship entities, such as the shape of decks or bulkheads, or the tank volumes and moments of inertia.
- It has multiple ways to communicate with other software, which is either the knowledge management system as applied in *Innovero*, or other end-user software, such as a general- purpose CAD system. Currently, implemented communication methods are *file*, named *pipes* and *TCP/IP*.
- Includes inter-process communication by means of XML.
- Support for export of the ship compartmentation, e.g. in 3D VRML file format, or as a user-defined 2D layout plain DXF, which can serve as an sub-layer or a general arrangement plan or a tank arrangement plan.
- Has multiple import and export options, notably from and to other PIAS modules, such as those for intact or damage stability.
- Includes tools for the management of design constraints. This is the subject of the next section.

In Fig. 5, an example of the GUI is presented. It consists of several dedicated windows; one presents a 3D overview of the hull and its internals, three windows present three orthogonal sections, and there are windows with a list of physical planes, reference planes and a compartment tree. It is interesting to see that the BSP is not included in this GUI, because it is irrelevant for the daily use of the program. However, for debugging purposes, the BSP can be visualised of which Fig. 4 gives an example.



BSP boom

```
Dwarsschot 5.000 (Dwarsvlak 5.000)
  Dwarsschot 20.000 (Dwarsvlak 20.000)
    Dwarsschot 90.000 (Dwarsvlak 90.000)
      Compartiment "Voorpiek"
      Dek 1.750 (Horizontaal vlak 1.750)
        Compartiment "Ruim"
        Dwarsschot 75.000 (Dwarsvlak 75.000)
          Compartiment "DB 1"
          Dwarsschot 60.000 (Dwarsvlak 60.000)
            Compartiment "DB 2"
            Dwarsschot 45.000 (Dwarsvlak 45.000)
              Compartiment "DB 3"
              Langsschot 0.000 (Langsvlak 0.000)
                Compartiment "DB 4 SB"
                Compartiment "DB 4 BB"
    Dek 3.000 (Horizontaal vlak 3.000)
      Compartiment "MK"
      Compartiment "DB onder MK"
  Langsschot 0.000 (Langsvlak 0.000)
    Dwarsschot 2.500 (Dwarsvlak 2.500)
      Compartiment "AP SB voor"
      Compartiment "AP SB achter"
    Compartiment "AP BB"
```

OK

Fig. 4: Part of the BSP of a ship layout design

3. Constraint Management

3.1 Introduction

Ship designers have to deal with numerous types of rules and regulations imposed by classification societies and regulatory authorities. These rules prescribe guidelines for the design of a ship in general. These rules generally dictate requirements for the positioning of bulkheads and compartments in the total ship layout, (damage) stability, and freeboard. With many different, mostly empirical, formulas the exact boundaries of these rules are determined. Most of these rules can be found in different rule-books, text or via digital (static) index files.

Besides the regulatory rules as described above, the designer has to take owner requirements into account. These rules generally prescribe requirements for compartment volumes and dimensions, deck areas, speed, noise, building costs etc. of which some are known in advance in the form of a list of requirements and others are gradually specified during the design process.

It would be useful to develop a constraint management tool that collects the different types of design constraints. Besides, it must be able to check if the geometrical model complies with the submitted design constraints. If some of the constraints are not satisfied, feedback on the confliction region(s) must be given to the designer. Even more, a solution proposal should be provided which complies best with the submitted set of design constraints.

3.2 State of the art

In the shipbuilding industry a lot of research is performed on knowledge based design. However, to our knowledge, no program exists which meets the requirements as stated in paragraph 3.1. Some research is done on knowledge-based design in the maritime industry and other sectors; in *Augusto and Kawano (1998)* a ship structural design is optimised with a nonlinear search algorithm. The focus of this research lies in the optimisation of the hull structural parts. The way of defining the constraints in the form of a penalty function can be used in the same way for the constraint management problem.

In *Yu et al. (2010)* a new method for offshore platform design is presented. This method allows better reusability and changeability of the compartment layout during the design process by describing the layout model as a parametric design. The method uses a 2D arrangement as a basis and performs parametric changes in this 2D plane, deck and bulkhead heights must be entered externally and are not adjusted during the optimisation routine. The described method works well for simple compartment layouts, but cannot be used for ship compartment design where vertical subdivision is of equal importance as transverse and longitudinal subdivision. Apart from an arrangement evaluation, hydrostatic and FEM calculations are also considered in the described method. The general approach of this implementation may be used for the addition of other constraints like initial stability and trim requirements in the constraint management system.

In *Lee and Lee (1997)* an intelligent compartment design system is developed which supports compartment design of a crude oil tanker. The nature of the research shows many similarities with the constraint management problem. However, the whole constraint management structure focuses only on design rules of crude oil tankers and is therefore not able to handle general constraints. Mainly IF-THEN statements are used to implement the specific rules, which results in a program with predefined constraints that is not capable of handling other types of constraints.

3.3 Implementation

3.3.1 General constraint description

In the 'to be developed' constraint management tool ideally all types of rules and requirements as mentioned in paragraph 3.1 are taken into account. Purely considered from the calculation-difficulty point of view this will not cause any problem. All evaluations deal with calculations that are not very complex to perform. However, there are some evaluations that nowadays still take a considerable time to calculate. E.g. the different evaluations and calculations to perform a probabilistic stability calculation are not complex, but all the different possibilities that need to be calculated make the calculation exhaustive. With evaluations that still take several hours to calculate, some of the design constraints will not be taken into account at the moment.

For the constraint management project we have restricted ourselves to address the constraint problem as far as it can be included in the design phase, in a processing time that allows interactivity. Implementing the calculation-intensive evaluations will result in long calculation times that will influence the workability in a negative way. For the time being, all constraints that deal with plane positions, areas and compartment volumes are considered only.

A way is found to describe the varying list of *design constraints* in a general way. It is found that the majority of the common constraints belong to one of the groups:

- amount (e.g. number of bulkheads),
- position (e.g. tanktop position),
- area (e.g. deck area),
- volume (e.g. fuel oil tank volume).

Besides belonging to a group, each constraint boundary can be of the type:

- minimum (at least…),
- maximum (at the most…)
- minmax (in-between ... and …).

When the constraint *group* and *type* definition is completed, it is known which boundary values should be given to the constraint. Where the min and max type only require one (lower *or* upper) boundary, the minmax type requires two (lower *and* upper) boundaries. With the two properties ‘group’ and ‘type’ combined, the majority of constraints can be described in a general way. Some examples are given in Table I.

Table I: Example of constraints assigned to a group and type.

Constraint	*Group*	*Type*	*Min*	*Max*
Fuel oil tank volume at least 900 [m^3]	Volume	Min	900	-
Collision BH between 88.2 [m] and 91.0 [m]	Position	MinMax	88.2	91.0
RoRo deck area less than 2200 [m^2]	Area	Max	-	2200
At least four bulkheads	Amount	Min	4	-

Finally, the constraint boundary values must be calculated. As mentioned in sub-section 3.1, the constraints occur in all shapes and sizes. Usually they are given in empirical formulas, but also lookup tables and graphs are used to determine the constraints. To be able to implement all these types of constraints, a general mathematical formula editor is required. Instead of adding it to the constraint management tool itself, the strength of the existing formula worksheet *MS Excel* is used. Within Excel almost all types of mathematical equations can be entered. The use of look-up tables and logical rules (IF…THEN) is also a possibility. By using the *Dynamic Data Exchange* (DDE) communication channel, a link is made between the constraint management tool and an *Excel* worksheet. In this worksheet the ship main dimensions are imported and constraint formulas use these parameters to calculate the constraint values. The final outcome of the formula is sent back to the constraint management program as a lower or upper boundary of the constraint.

3.3.2 Linking the constraints to the geometric model

In our new program, a geometric model consists of planes which partition the ship hull space in different compartments (see paragraph 2.4). Horizontal planes will serve as decks, vertical planes as transverse or longitudinal bulkheads. An entity in the current ship layout design contains some general properties (name, number, weight group, etc.) and geometrical information (position, corner points, direction, etc.) of the model components. In addition to that, for each plane or compartment the connected constraints can be chosen from the list of design constraints.

3.4 Mathematical description of the model

With the link between the geometric model and the constraints, all required input is given to perform a model evaluation. In the following part the constraint management problem is described in a mathematical form based on a standard optimisation problem in conformity with the description as in *Augusto and Kawano (1998)*. The standard optimisation algorithm consists of design variables, an objective function and the description of constraints with penalty functions. These last two are evaluated as a whole in the combined *penalised objective function*. The design variables serve as the input for the optimisation problem and are changed systematically to satisfy the main optimisation goal: minimising the value of the *penalised objective function*.

This general description of the optimisation problem is used as a basis for satisfying the constraints. Although the common goal of an optimisation problem is to minimise the objective function, the constraint management program of this paper uses it for the satisfaction of constraints. Therefore it is not an optimisation task, but a constraint satisfaction problem, with the objective function only occupying a secondary role.

3.4.1 Set of equations

The design variables specify the design and the goal is to find the variables for which the design is best. Written in the form of a vector, with n independent components, the design variables can be given as:

$$X = \{x_1, x_2, x_3, \ldots, x_n\}, \qquad (4.1)$$

For the constraint management program, the plane positions are the design variables. Every x represents a plane position, and n represents the number of planes that are taken into account. The plane positions determine the ship layout and by judging their position the quality of the compartment design is measured. The planes can be moved in their orthogonal direction only.

When the constraint management tool evaluates the compartment design, there already exists a ship layout. This layout is made by the designer, so this can be seen as the preferred compartment design, given that no other constraints apply. The main goal of the constraint management system is to find a solution that complies best with all the constraints, but which needs the minimum amount of plane movements; after all we do not want to annoy the designer. This last requirement directly determines our *objective function*; while satisfying the constraints, minimise the plane position adjustments to match the layout design best with the designer's intentions. The *objective function* is described as:

$$f(x) = q \cdot \sum_{i=1}^{n} \left(x_{i,\,old} - x_{i,\,new} \right)^{\delta}, \qquad \textit{(4.2)}$$

with q, δ as scale factors to control the influence of the object function on the penalised optimisation function.

Subsequently, the description of the constraints is described in the following part. Usually, design problems are subject to series of inequality constraints in the form $g(x) \geq 0$. In case of a maximum constraint $x < x_{\max}$, this can be rewritten as:

$$g(x) = -x + x_{\max} \geq 0, \qquad \textit{(4.3)}$$

which is the same format as the standard inequality constraint. In paragraph 3.3.1 the various types of constraints were discussed. All these constraints can be written in the standard form and are thereby described digitally.

The compliance of the different constraints is described in the form of a penalty function. As described in *Augusto and Kawano (1998)* there exist *interior* and *exterior* penalty functions. When ship design constraints are considered, usually a starting point in an unfeasible area is given. The goal is to reach a feasible area regardless where exactly within that area. In that case an *exterior penalty function* is used since it is the only one capable of investigating an infeasible area. The *exterior penalty function* used for the constraint management program is:

$$p(X, r) = r \cdot \sum_{i=1}^{k} \alpha_i \cdot \left(\min\{0, g_i(x)\} \right)^{\beta_i}, \text{ with:} \qquad \textit{(4.4)}$$

$$\begin{aligned} \min\{0, g_i(x)\} &= 0, \ \textit{if} \ \ g_i(x) \geq 0 \\ &= g_i(x), \ \textit{if} \ \ g_i(x) < 0 \end{aligned} \qquad \textit{(4.5)}$$

The penalty value is zero in case the constraint [*(4.3)*] is satisfied. If not, the squared difference of the boundary and the actual value will be the penalty for that specific constraint.

Finally, the parts given in the foregoing paragraphs are combined in the *penalised objective function* $\phi(x, r)$, given by:

$$\begin{aligned} \phi(x, r) &= f(x) + p(X, r_k) \\ &= q \cdot \sum_{i=1}^{n} \left(x_{i,\,old} - x_{i,\,new} \right)^{\delta} + r \cdot \sum_{i=1}^{k} \alpha_i \cdot \left(\min\{0, g_i(x)\} \right)^{\beta_i}, \end{aligned} \qquad \textit{(4.6)}$$

in which f(X) stands for the *objective function* and p(X,r$_k$) for the *penalty function*. The *penalised objective function* describes the total performance of the compartment design. The lowest resulting value of this formula will represent the layout model that complies best with the evaluated constraints. With the q, δ, r terms, the influence of the different terms relative to each other can be controlled.

The α_i, β_i terms determine the scaling of the different constraint penalties to each other only.

3.4.2 Constraint satisfaction

In order to find the minimum of the *penalised objective function,* a Quasi Newton method is used. The overall minimum of this function will indicate the combination of plane positions that satisfies the constraints best. The strength of approaching the constraint management program this way is that, regardless of the feasibility of the set of constraints, a solution (a minimum) will always be found. By inspection of the value of the *penalised objective function* it can be seen if the solution found is a feasible or an infeasible one.

In case of an infeasible compartment design, it is difficult to judge which action has to be taken. An infeasible design means that two or more constraints are mutually conflicting. Which of these constraints is the 'problematic' one cannot be judged by the constraint management tool itself. After all, the design constraints are all included by the designer in the first place; he is the one to decide what to do in the infeasible situation.

When the combination of constraints is infeasible, the α_i scaling factors in *(4.6)* highly influence the outcome. By changing the individual scaling factors the dominance of the constraints can be controlled. In order to give the designer control over the scaling factors, the *constraint equaliser* is introduced. In this window (additional to the main GUI) each constraint is represented by a *trackbar*. By moving this *trackbar* up or down, the constraint dominance is increased or decreased (see Figs. 7 and 8). If the designer is able to change the weighing of each individual constraint, the outcome of an infeasible constraint combination can be influenced and preference can be given to a specific constraint. In this way the interpretation of the unfeasible design is left to the designer, but the influence of the entire constraint system is given instantly.

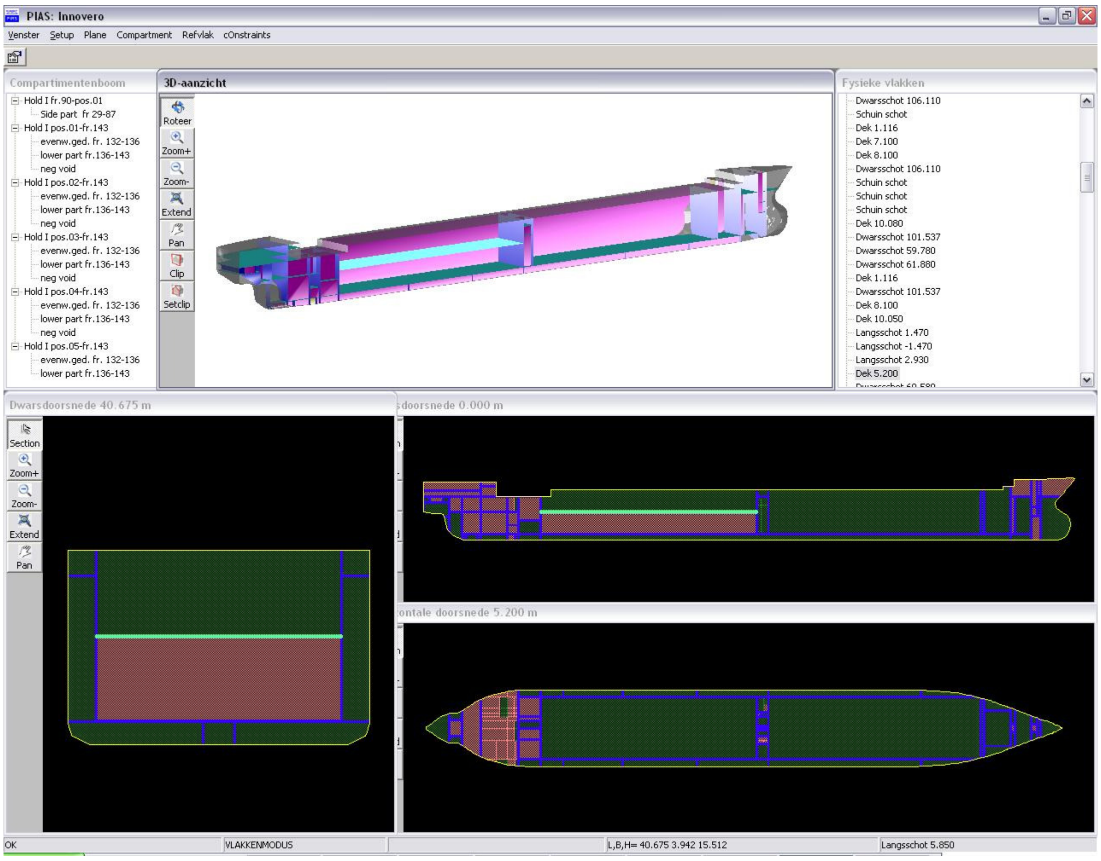


Fig 5: Graphical User Interface

4. Application examples

In Fig. 5 the main GUI of the new compartment design program is displayed. Visible are the three orthogonal projections and one central 3D view on the ship. Both the tree of compartments (left) and a list of physical planes (right) are visible. The ship layout design of Fig. 6 was created within 15 min. In the colours blue, pink and green the orthogonal planes can be seen. Oblique planes are coloured yellow.

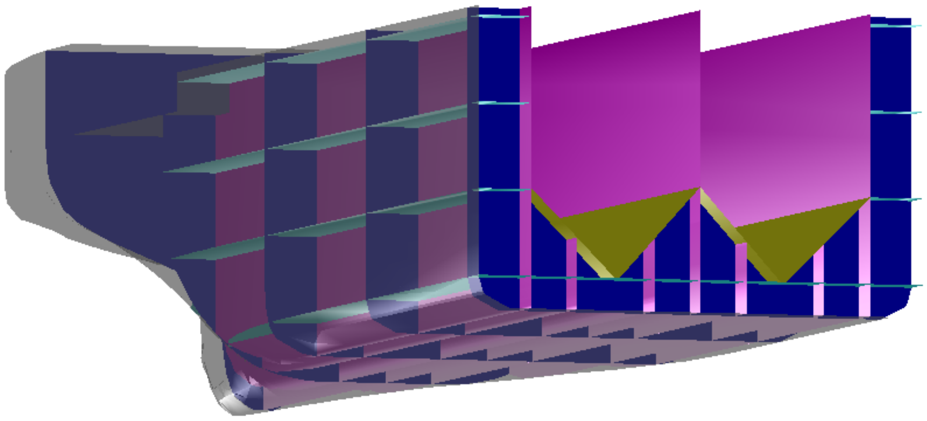
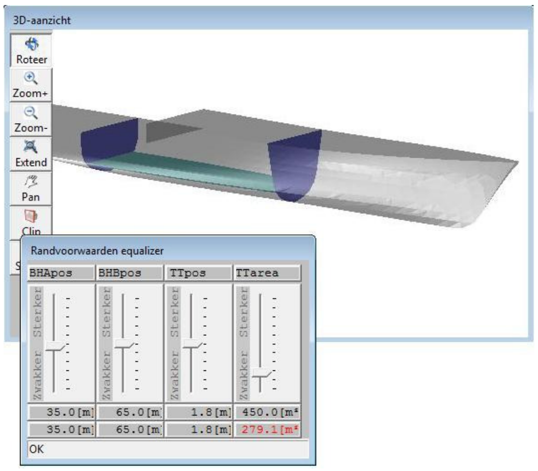


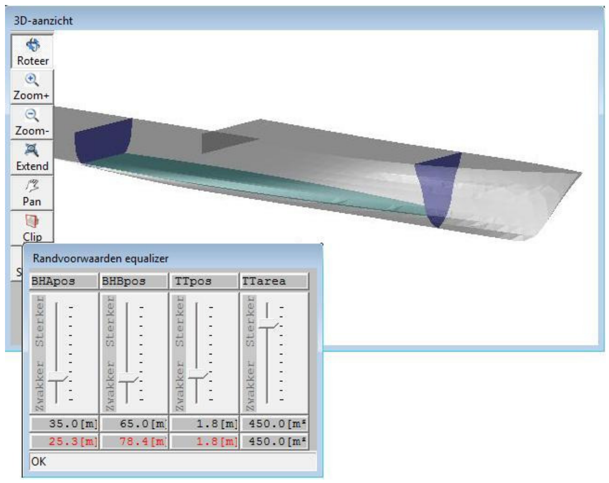
Fig 6: Example ship layout

In Figs. 7 and 8 a practical example of the use of the *constraint equaliser* is displayed. The model consists of a simple compartment design, just for good understanding. Two bulkheads (coloured blue) are linked to the 'bulkhead aft position' and 'bulkhead forward position' constraints. The tanktop (green) is connected with the 'tanktop position' and 'tanktop area' constraints. The constraints and their scaling can be seen in the *trackbar* windows. A fact is that this combination of constraints is infeasible. In order to meet the tanktop area constraint, the forward and aft bulkhead (which bound the tanktop) need to be moved forward and backward respectively. However, their position constraint does not allow them to move further than a specific position, which is 35 [m] for the aft bulkhead and 65 [m] for the forward bulkhead. So moving the bulkheads to increase the area constraints violates the position constraints. The other way around, i.e. moving the bulkheads to their constraint-positions, will violate the 'tanktop area' constraint. By shifting the *trackbars* the dominance of the different constraints can be adjusted. In Fig. 7 the scaling factors are changed to satisfy the position constraints, in Fig. 8 the *trackbars* are moved to meet the 'tanktop area' criterion.



BHApos	BHBpos	TTpos	TTarea
35.0[m]	65.0[m	1.8[m]	450.0[m²
35.0[m]	65.0[m	1.8[m]	279.1[m²

Fig 7: Infeasible constraint combination, position constraints comply



BHApos	BHBpos	TTpos	TTarea
35.0[m]	65.0[m	1.8[m]	450.0[m²
25.3[m]	78.4[m	1.8[m]	450.0[m²

Fig 8: Infeasible constraint combination, area constraint complies

5. Conclusion and subjects for further research and development

The developed *Binary Space Partitioning* (BSP) method, as discussed in this paper, seems to work very well for designers in the early phase of the design process. The method requires minimum input by the designer. However, he or she has to think ahead to be able to use the right order by which the spaces are split into two adjacent new subspaces. More work has to be and can be done to further improve this new subdividing method. An additional feature would be to also be able to add volumes which can be subdivided later on. This way the superstructures and deckhouses can be included in the design. Also the possibility to re-use (parts of) subdivisions could help to speed up the design process. The topology within a particular class of ships is often very similar or even the same. Also the topology of parts of the design can be identical to existing designs. This way, a new ship could be generated by using pre-defined parametric “building blocks”. In this way, parametric models of alternative design options can be generated quickly and into more detail, providing also the possibility to analyse and optimise the designs when used as input for analysis tools.

The constraint management method as programmed and described in this paper is also a very interesting feature to be explored further. The program as it is right now can deal with the design constraints related to the positions, areas and volumes as defined by the BSP method. However, some valuable additions can be made like an initial stability and trim calculation. The constraint management method has only been used for one particular design topology. This method, however, also gives the designer the opportunity to create alternative topologies while the computer monitors the compliance with the constraints. In addition, a next step forward would be to combine this method with the option of also being able to vary the dimensions and other design parameters of the hull form. This way the effect these changes will have on layout, areas, volumes and capacities as well as on additional performance characteristics of the design like resistance, weight and costs can be investigated and included in the process of finding an optimal solution.

References

ALONSO, F.; GONZALEZ, C.; PINTO, A.; ZURDO, F. (2008). *Integrated management of ship compartments.* 7th Int. Conf. on Computer Applications and Information Technology in the Maritime Industries (COMPIT) 08, Liege.

AUGUSTO, O.B.; KAWANO, A. (1998). *A mixed continuous and discrete nonlinear constraint algorithm for optimizing ship hull structural design.* Ocean Engineering 25/9, pp.793 - 811

BUCHELE, S.F.; ROLES, A.C. (2001). *Binary space partition tree and constructive solid geometry tree representations for objects bounded by curved surfaces.* 13th Canadian Conf. on Computational Geometry.

COMBA, J.L.D.; NAYLOR, B.F. (1996). *Conversion of binary space partitioning trees to boundary representation*, Geometric Modelling: Theory and Practice, pp.286-301. Springer.

DE BERG, M.; VAN KREVELD, M.; OVERMARS, M.; SCHWARZKOPF, O. (1998). *Computational Geometry*. Springer.

GHALI, S. (2008). *Introduction to Geometric Computing.* Springer.

VAN HEES, M. (2009). *Quaestor: taxonomy-based compositional modeling and product configuration.* 10th Int. Marine Design Conference, Trondheim, pp.630-647.

KOELMAN, H.J. (2003). *Application of the H-rep ship hull modelling concept.* Ship Technology Research 50/4, pp.172-181.

LEE, K.H.; LEE K.Y, (1997). *Knowledge-based nonmonotonic reasoning process in ship compartment design system.* Expert Systems with Applications 13/2, pp.145−154.

LEE, K.Y.; LEE, S.U.; CHO, D.Y.; ROH, M.I.; KANG, S.C.; SEO, J.W. (2003). *An innovative compartment modeling and ship calculation system.* 8th Int. Marine Design Conf., Athens, pp.683-694.

LEE, S.U.; ROH, M.I.; CHA, J.H.; LEE, K.Y. (2009). *Ship compartment modeling based on a non-manifold polyhedron modeling kernel.* Advances in Engineering Software 40/5, pp.378-388.

LYSENKO, M.; D'SOUZA, R.; SHENE, C.K. (2008). *Improved binary space partition merging.* Computer-Aided Design 40/12, pp.1113-1120.

MÄNTYLÄ, M.J. (1988). *An Introduction to Solid Modeling.* Computer Science Press.

SCHNEIDER, P.J.; EBERLY, D.H. (2003). *Geometric Tools for Computer Graphics.* Morgan Kaufmann.

SMITH, J.M.; DODGSON, N.A. (2007). *A topologically robust algorithm for Boolean operations on polyhedral shapes using approximate arithmetic.* Computer-Aided Design 39/2, pp.149-163.

THIBAULT, W.C. (1987). *Application of binary space partitioning trees to geometric modeling and ray-tracing*. PhD thesis, Atlanta

THIBAULT, W.C.; NAYLOR, B.F. (1987). *Set operations on polyhedra using binary space partitioning trees.* SIGGRAPH Comput. Graph. 21/4, pp.153-162.

VAN DEN BERGEN, G. (1999). *Collision Detection in Interactive 3D Computer Animation*. PhD thesis, Eindhoven.

VANECEK, G. (1991). *Brep-index: A multidimensional space partitioning tree.* 1st Symp. Solid Modeling Foundations and CAD/CAM Applications, pp.35-44.

YU, Y.Y.; CHEN, M.; LIN, Y.; JI, Z.S. (2010). *A new method for platform design based on parametric technology.* Ocean Engineering 37/5 ‐ 6, pp.473 ‐ 482.