SUPREME: Submarine Space Partitioning in Rhino by Quaestor3

M. Th. van Hees, Maritime Research Institute Netherlands (MARIN)

W.H. van den Broek-de Bruijn, Defence Materiel Organisation (DMO)

Synopsis

Within the scope of the Royal Netherlands Navy's Walrus class replacement programme, DMO embarked in 2015 in developing a set of tools to generate and assess submarine concept designs. MARIN took up the challenge to develop a design workflow SUPREME on the basis of its Quaestor3 knowledge engineering framework [van Hees 1997, 2003, 2009]. Within this workflow a number of proprietary and often confidential tools should be plugged in to perform e.g. propulsion system dimensioning, weight estimation and for assessment of trimming and compensation capabilities. These 'satellite' tools are mostly developed for, or by, DMO. As a design system, SUPREME becomes fully operational at DMO where workflow and proprietary tools come together. The developmental focus of MARIN is the SUPREME workflow which deals with design knowledge representation and design data management. The two major challenges in its development were the topological representation and the weight management methodology. This paper will discuss the first one, the development of a submarine topological representation using the CAD system Rhinoceros™ as externally controlled geometry generator. This combines high accuracy with adaptability and enables naval architects to constantly monitor the weight and volume balances and trimming / compensation capabilities of concept submarines already in the earliest stages of design.

Keywords:

Submarine design, spatial arrangement, knowledge engineering, workflow, binary space partitioning, CAD.

1. Introduction

The Defence Materiel Organisation (DMO) is involved in the procurement, maintenance and sale of materiel for all military personnel. The Maritime Systems Department within the DMO is the engineering firm that supports the Royal Netherlands Navy with technical knowledge during the different life stages of a platform. To support our role as an engineering firm, various computer programs are used, most of them are specially made for or by our department [Oers, et al., 2017]. Supreme is a software program that has been developed to model submarines at any desired level of detail [van den Broek-de Bruijn, 2016].

In ship design a long standing issue is the duality between surfaces and compartments. For practical reasons surfaces are used for objects such as shell, bulkheads and decks. Reasons to use surfaces may be e.g. weight estimation and the positioning of (large) components. Although these surfaces are separating the hull into compartments there is no obvious solution for assigning a particular surface to a compartment or vice versa. On the other hand, a compartment based representation is desired for space allocation, systems design and volume balance. The solution to this issue as developed for SUPREME is presented in this paper. The general idea is based on [Koelman, 2012].

The development environment used is MARIN's Quaestor3 knowledge engineering framework. The Taxonomy/Entity (T/E) modelling methodology of Quaestor3 is used for the development of SUPREME and is described in [van Hees, 2009]. This work is a continuation of the application of Rhinoceros™ (in short Rhino) in combination with Quaestor3 for the design of naval surface vessels by DMO (DESIS1-3, GCD²) and was also funded by DMO. Support in developing the Rhino Quaestor3 integration was provided by https://www.rhinocentre.nl with special thanks to Jess Maertterer.

2. Solving the surface vs. space duality

As stated above, the natural way to divide a space within a ship's hull is by means of decks, walls and bulkheads which represent the construction and describe the layout. By adopting surfaces as leading in the structural lay out, the taxonomy of a submarine topology may look as shown in Figure 1.

 http://doi.org/10.24868/issn.2515-818X.2018.005

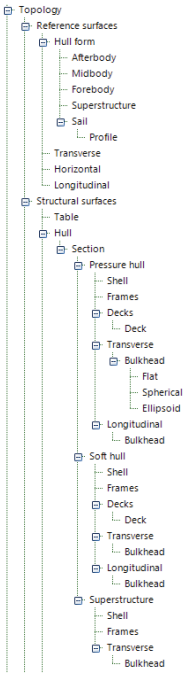

Figure 1: Taxonomy of a surface based model of a submarine

Figure 1 presents a straight forward object model of the structure from which data can be aggregated and used downstream in the design workflow – a taxonomy. It allows for integration with software for strength calculation, weight estimation and hydrodynamic performance which requires the hull form. The above taxonomy models a submarine as a number of sections containing bulkheads and decks forming spaces in tanks, both inside the pressure hull (*Pressure hull* branch) and outside the pressure hull (*Soft hull* branch). Although the arrangement in Figure 1 covers some important naval architectural aspects it cannot deal with surfaces and compartments at the same time in a flexible manner.

A compartment based approach is very useful, in particular if it can be instantly modelled in a CAD program like Rhino which can create complex closed shapes, compute their properties and can be externally controlled through an API. Although by itself not extremely complex, submarine design requires careful management of masses and volumes to achieve submerged equilibrium combined with a minimum required stability lever. The required accuracy in determining the centre of buoyancy of all parts contributing to the displacement and the position and mass of all components in the submarine is much higher than for surface ships. This is why SUPREME intends to model and manage design concepts from the earliest stages of design with the highest possible accuracy. The taxonomy of SUPREME was designed in particular for interaction with Rhino and to manage performance, weights and volumes.

Within the scope of the INNOVERO project, H. J, Koelman et. al. [Koelman 2012] developed a novel approach to ship space partitioning which basically overcomes the duality between surfaces and compartments. They used Binary Space Partitioning (BSP), a space partitioning methodology in particular suited to efficient 3D rendering.

The basic idea of BSP is that a space is divided in two using one or more lines (in 2D) or planes (in 3D) as shown in Figure 2:

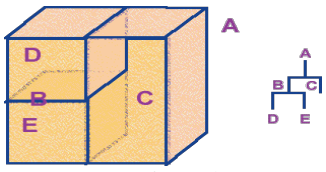

Figure 2: BSP of an arbitrary space

In order to overcome the surface / space duality in submarine design we also decided to use BSP as the organisational principle for the topology. The Taxonomy/Entity (T/E) methodology in Quaestor3 provides means to efficiently implement a variety of computational methods for design [van Hees, 1997, 2003, 2009]. In T/E applications, standard knowledge engineering principles are used. Instances of the multiple entity type necessarily contain the same set of properties and methods. Complex hierarchies can be created based on a single entity type which avoids an explosion of nodes required in the topology section of the taxonomy. If separate entity types would be required to represent a divider and a space, and depending on the organisational concept,

 http://doi.org/10.24868/issn.2515-818X.2018.005

each additional level in the BSP would require at least 2 times the number of nodes in the parent. Therefore, a *single* entity type should be used to represent both spaces and surfaces so that this can be achieved entity per level, each a child of the previous.

Standardisation of data and knowledge modelling provide advantages in terms of efficiency but flexibility is lost compared to e.g. C# or C++. Any tool or language can be used as far as it fits into the data structures in Quaestor3 T/E applications. We searched for a design pattern which overcomes the above limitations of T/E based computational modelling. Koelman's method uses BSP as connection between compartment and planes and hides the BSP from the user. Therefore, a representation with the least possible number of levels is achieved to keep it manageable for both system and the user. This is achieved by storing the *Divider* (or hierarchy of dividers), the *Primary Space* and *Secondary Space* on the same level as depicted in Figure 3:

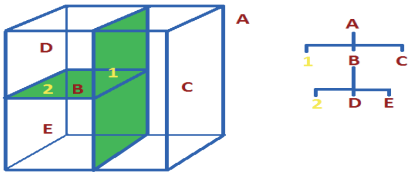

Figure 3: SUPREME Binary Space Partitioning

Application of the principle used in Figure 3 on a typical submarine layout results in a tree shown in Figure 4:

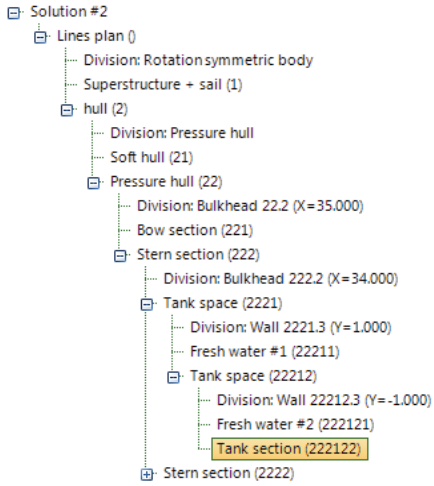

Figure 4: SUPREME BSP tree for a part of a submarine layout

The tree in Figure 4 follows the following principle: the *Divider*, *Primary space* and *Secondary space* all are instances of the same entity type. The first child of a *Space* is the *Divider* (which may have further dividers as child if recursive as explained later). The second child is the *Primary space* and the third child is the *Secondary space*. What is meant with *Primary space* and *Secondary space* is discussed in detail below.

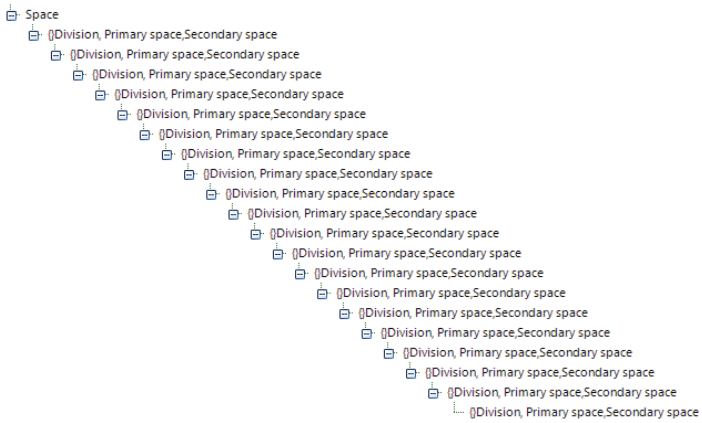

Figure 5: Example taxonomy of space partitioning

 http://doi.org/10.24868/issn.2515-818X.2018.005

The above taxonomy shows that the entity instances may have zero, one or three child instances. In the event of one child, the instance assumes the role of *Divider* to create complex dividers as explained below. In the event of three child instances, the first child assumes the role of *Divider* (with its possible divider child instances), the second instance is the *Primary space* and the third instance will be the *Secondary space*. Please note that all are instances of the same entity type, so the same set of parameters methods are defining both dividers and spaces. The actual compartments at a given location in the BSP tree are the second and third instances that are not partitioned, i.e. have no child instances.

3. Quaestor3 - Rhino interaction

The combination of Quaestor3 and Rhino is in use by DMO in various ship design tools since 2006. In Figure 6 a screenshot of DeSIS (2009) is presented.

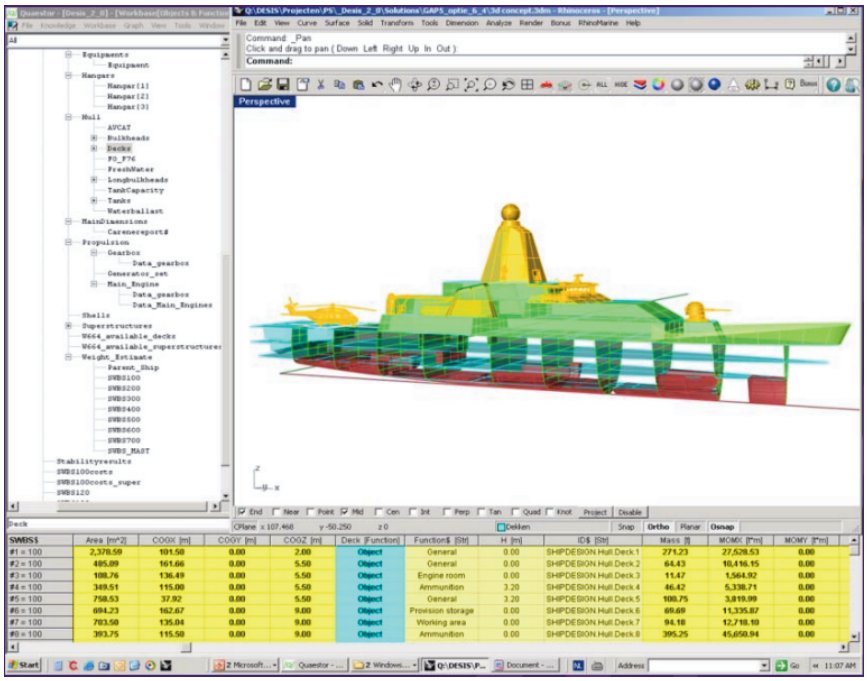

Figure 6: Screenshot of DeSIS design system by DMO (2009)

Similar to SUPREME these Quaestor3 models are primarily used to manage and apply design knowledge and to create CAD models through scripting. In models such as DeSIS and its successor GCD[2] (Figure 7), Quaestor3 generates scripts and controls Rhino to execute them in a logical sequence. From these scripts, Rhino creates or adapts a 3D model of a concept and returns properties of particular surfaces and volumes. Quaestor3 uses Rhino in these models as any other satellite tool like Excel or Matlab. It is unaware of the objects that exist within the CAD model and cannot highlight, hide or select objects in Rhino depending on your position in the workflow. Changes to a model can only be achieved by creating and running scripts. This makes the scripting approach unsuitable to create an *interactive* 3D modeller for conceptual design.

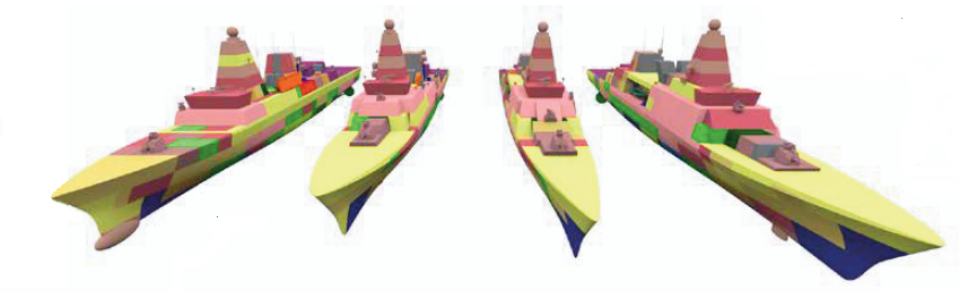

Figure 7: Various concept designs of naval combatants produced with GCD[2] [Takken, 2016]

To make SUPREME into an interactive 3D design environment it was necessary to combine Quaestor3 and Rhino into a generic CAD system. The idea behind generic CAD is to design by means of 'primitives' and by performing a limited number of operations on these primitives. The nature of submarine design is such that important parts of the topology can be created by e.g. axis-symmetric shapes such as cylinders, cones, spheres,

 http://doi.org/10.24868/issn.2515-818X.2018.005

ellipsoids and ring stiffeners. Components which are no simple primitives like sail, rudders, superstructures or substructures can still be created as primitives by scripting and merged into the rest of the geometry. Only basic operations on these primitives are required for e.g. joining, intersecting, splitting and Boolean operations:

- RHINOPRIMIT$(): Creation of open and closed revolved surfaces, joining of surfaces, copying objects, creating objects by execution of Rhino scripts, (bounding box) point objects, volume or area centroids, Boolean operations on surfaces, importing and placing Rhino component models as secondary objects. This function is used in SUPREME to e.g. create all components of the pressure hull including end bulkheads, ring stiffeners and penetrations.
- RHINO#(): Calculation of the (combined) properties of surfaces and volumes (closed surfaces), calculation of section properties of intersection plane of closed surfaces, volume properties of sections, e.g. for calculation of tank tables, bounding box coordinates. This function is used in SUPREME to calculate all properties required for the hydrostatics, weight and centre of gravity.
- RHINOPARTITION$(): Performs the space partitioning as described in this paper. This function creates the partitioning of the pressure hull and soft hull and creates all walls, bulkheads and decks that make up the dividers. Several aspects of the methodology and the rules applied are discussed in the following.
- RHINOTRANSFORM$(): Selects and transforms a geometric object from a library to their final size, orientation and position in the submarine.

4. Rhino control

Rhino is controlled by SUPREME to achieve the following:

- Create a CAD model from primitives, scripts and component models using RHINOPRIMIT$(), RHINOPARTITION$() and RHINOTRANSFORM$();
- Calculate properties from Rhino objects such as area, volume and centroid data using RHINO#();
- Manage CAD models created by SUPREME. Saving and loading files per solution is controlled by SUPREME;
- Maintain the consistency of CAD models with the data in the SUPREME workflow. Changes made in topology are propagated by SUPREME into the CAD model, resulting in the destruction and recreation of all objects that are related. Objects in the model are therefore associated via SUPREME. Propagation of changes in any computational model is standard behaviour of Quaestor3, it is made aware that certain parameters in the workflow represent Rhino objects which are immediately destroyed if their associated values become pending.
- Create a useful layer hierarchy in the CAD model so that it can also be used offline from SUPREME;
- Act as object browser for the CAD model, to show and highlight objects in focus in SUPREME and to hide everything else.

Due to computational demands of e.g. hydrostatics, based on the arrangement and partitioning of pressure hull, soft hull and appendages, many derived objects are created by combining their components in different ways. This is done by means of Boolean and joining operations in Rhino which can fail for a variety of reasons. An event logging mechanism in SUPREME traces all Rhino operations and the objects used. The application of different policies for unit tolerances for the creation, joining and intersecting objects have made errors in these operations relatively easy to resolve.

During a modelling session, Rhino is continuously controlled by SUPREME. User interaction is limited to the selection of an object in the model which instructs SUPREME to move focus to the entity were the object is created. Secondary objects from the SUPREME component library (motors, cabinets, bunks, batteries etc.) can be selected in Rhino and moved using DragDrop or BoxEdit where Rhino returns the new position to the corresponding item to SUPREME (Figure 8).

 http://doi.org/10.24868/issn.2515-818X.2018.005

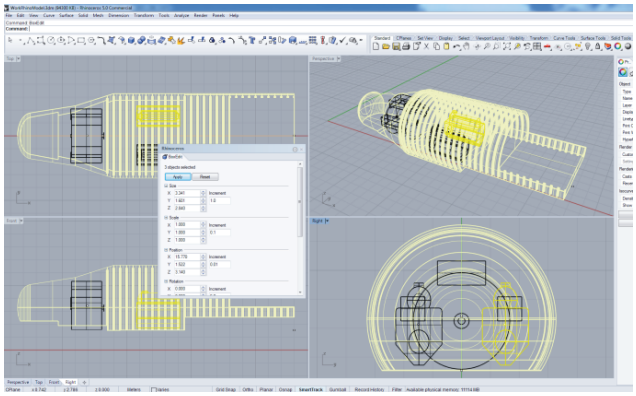

Figure 8: Placement / moving secondary objects

Secondary objects are placed relative to the reference point of the associated space or surface; so, when a space moves, the associated secondary objects will move with it.

5. Space partitioning with complex dividers

The basic objects used in space partitioning are decks, walls, bulkheads and closed surfaces. Although the outer shape of the hull is represented by a closed surface representing the hull lines, closed shapes can also be used as internal divider. An example is the outside pressure hull shape and all its protrusions (hatches, torpedo tubes, ring stiffeners). The generation of both the hull lines and the pressure hull inside and outside geometry, its primary stiffeners and the pressure hull penetrations are also part of the SUPREME workflow. Both the hull lines and pressure hull geometry are created on the basis of primitives and functions as described above and is not further discussed in this paper. This results in the Soft hull and the Pressure hull as compartments in the BSP tree which are further subdivided. We will now focus on the partitioning of these compartments primarily using decks, walls and bulkheads.

Decks, walls and bulkheads only allow for single orthogonal space partitioning which is insufficient for more complex partitioning, needed e.g. for battery spaces or complex tanks. The partitioning function RHINOPARTITION$() implements a concept denominated ‘recursive cutting’ which provides the means to create a divider consisting of a number of parts in a number of steps in the BSP tree. Each of these steps is represented by a child node of the previous step. Eventually the parts in this divider hierarchy are joined into a polysurface. This polysurface is subsequently used to split the parent compartment represented by a closed surface with the Rhino BooleanSplit() function. This should result in two new compartments and a full representation of the parts making up the divider. A simple example of a recursive divider is presented in Figure 9: a combination of a deck part and a bulkhead part:

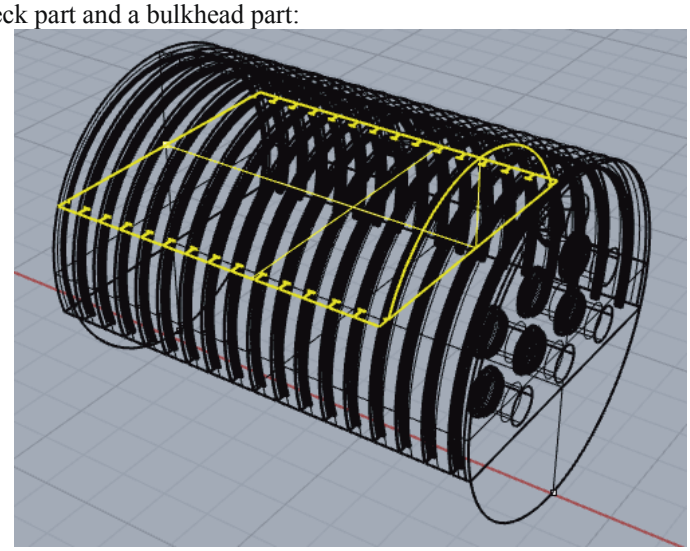

Figure 9: Example of a simple recursive divider in a compartment

In the BSP tree this part appears as follows:

 http://doi.org/10.24868/issn.2515-818X.2018.005

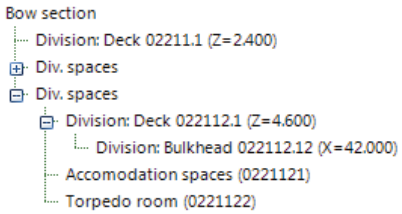

Figure 10: BSP tree section of a recursive divider in a compartment

In Figure 10, *Deck 022112.1* is the first child of *Div. spaces*. Similar to a space, a divider can also be divided or 'split' by its first child since a divider is a 2D space where a compartment is a 3D space. When a compartment is split in two by a dividing surface, the *Primary space* has to be selected, also defining the *Secondary space*. When a bulkhead is used, one needs to define the 'Primary space orientation' to be either *Forward* or *Aft*. For a wall it is respectively *Portside* or *Starboard*, for a deck *Above* or *Below* and for closed surfaces *Inside* or *Outside*. In all cases the selection is free except for upper level where space enveloped by the lines plan is partitioned by the pressure hull outer shape as explained earlier. On that level the Primary space orientation must be *Outside* making the Soft hull *Primary* and the pressure hull the *Secondary* space.

The 'Primary space orientation' is sufficient to define the order of the partitions in the BSP tree as is demonstrated in the parameter list for *Deck 022112.1* (Table 1):

Parameter	Value	Dimension
Division type	Deck (X-Y)	ID
Division name	Deck 022112.1 (Z=4.600)	Str
Divider position	Z=4.600	Str
Select horizontal reference plane	Base line	ID
Z offset	4.600	m
Z reference	0.000	m
Rotate around X	0.0	deg
Rotate around Y	0.0	deg
Division ready?	No	BLN
Primary space orientation	Above	m^4
Rhino ID	91b1a851-3bc6-4c43-aa45-a7d9176ff3ba	Str

Table 1: Parameters defining Deck 022112.1 to be divided itself

The bulkhead part of Figure 10, splitting the above *Deck 022112.1* is defined by the parameter list for *Bulkhead 022112.12* (Table 2):

Parameter	Value	Dimension
Division type	Bulkhead (Y-Z)	ID
Division name	Bulkhead 022112.12 (X=42.000)	Str
Divider position	X=42.000	Str
Select transverse reference plane	Station X = 42.000	ID
X offset	0.000	m
X reference	42.000	m
Rotate around Y	0.0	deg
Rotate around Z	0.0	deg
Division ready?	Yes	BLN
Keep parent Deck (X-Y) part	Aft	m^4
Keep Bulkhead (Y-Z) part	Upper part	ID
Rhino ID	29b216a6-36fd-4d79-b22c-8a18ee49ad37	Str

Table 2: Parameters defining Bulkhead 022112.12

The bulkhead defined in Table 2 is used to split the parent deck. By defining the part of the parent (Deck) to be maintained: 'Keep parent Deck (X-Y) part = Aft' it is decided which partition of the parent divider is to be deleted. In addition it needs to be specified which part of the current divider has to be maintained: 'Keep Bulkhead (Y-Z) part = Upper part'. It is also possible to use tilted decks, bulkheads and walls by rotation around X-Y, Y-Z and X-Z axes respectively.

More complex cases are handled in the same manner with the addition that all orthogonal parents are to be partitioned ('cut') with the current one until a divider is reached with the same (orthogonal) orientation (a deck meets a deck, a wall meets a wall of a bulkhead meets a bulkhead). As a result, the following 'cutting rules' are applied when you create e.g. a bulkhead:

- It will cut its parents by itself if it is either a wall or a deck but it will stop cutting parents once it reaches a surface with the same orthogonal orientation, in this case a bulkhead;
- In its turn, any divider added to the tree is cut with all parents until either the top is reached or a divider with the same orthogonal orientation;

 http://doi.org/10.24868/issn.2515-818X.2018.005

- Any divider added should have an orthogonal orientation different from its parent;
- Any divider added should create only two spaces together with its parents;
- A tilted deck, bulkhead or wall is considered to be orthogonal in these cutting rules.

A designer should consider the order in which the operations are performed. Although a different sequence of cutting may eventually lead to a similar result it may also create more parts than strictly needed, e.g. a deck which consists of more than one piece.

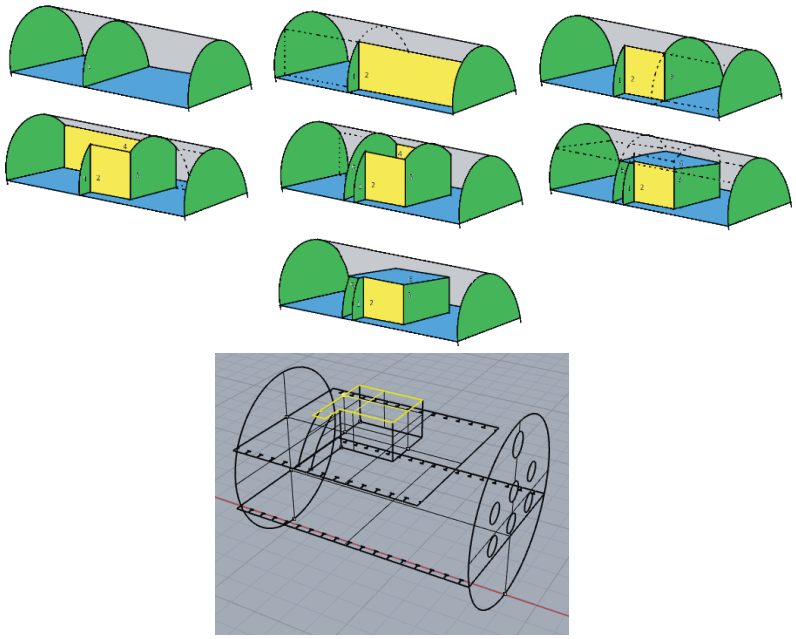

Figure 11: Creating a partition in 6 steps

6. Solving the 'closed trunk' limitation

This methodology makes it possible to create reasonably complex spaces such as stepped battery rooms or a complex well for e.g. a main electric motor as long as the above rules are followed. The example in Figure 11 also illustrates a limitation of the above rules for 'divider cutting'. They do not allow to close the dividers upon themselves, i.e. it is not possible through these rules to create a closed trunk like shown in Figure 12:

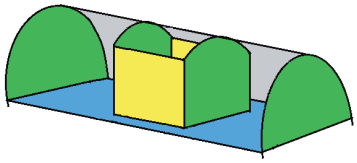

Figure 12: Closed trunk from two walls and two bulkheads

Creating this closed trunk by means of only the above divider cutting rules is not possible as shown in Figure 13 where the process has been simplified to the 2D case in top view:

 http://doi.org/10.24868/issn.2515-818X.2018.005

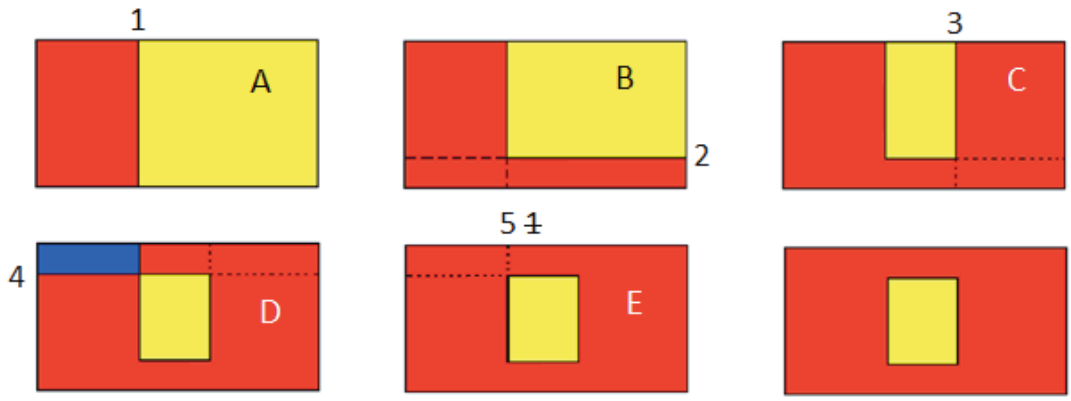

Figure 13: Creating a closed trunk from two walls and two bulkheads

By following the cutting rules in four steps we end up in D) with three spaces since Wall 4) cuts Bulkhead 3) and Bulkhead 3) cuts Wall 4) and then hits the barrier in the form of Wall 2) which has the same orientation as Wall 4). The solution to this problem is to add a 5th divider, being a copy of the version of Bulkhead 1 already cut by Wall 2) in step B). This Bulkhead 5) will cut wall 4) and will be cut by Wall 4), cannot cut Wall 2 (already done) and is blocked by Bulkhead 3) against further cutting. The original bulkhead from step A) will be removed by deleting the top node of the complex divider and moving its child nodes one level upward.

The resulting CAD model contains all compartments, their child compartments and all dividing surfaces. The information needed for e.g. the weight and volume balance is computed by Rhino upon request by SUPREME, for Rhino objects representing dividers and compartments. Results are fed as input to further analyses and tools that are plugged into the SUPREME knowledgebase.

7. What does it take to create a spatial arrangement?

Figure 14 shows a worked example of a submarine spatial arrangement.

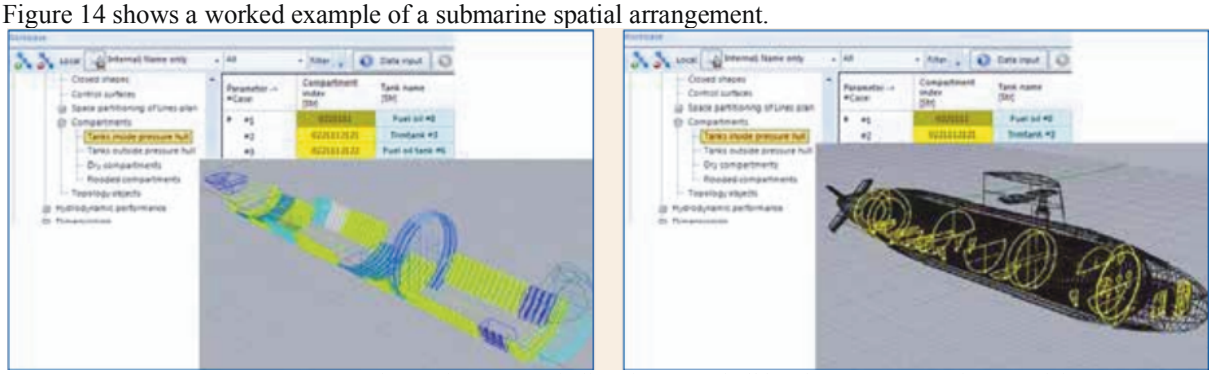

Figure 14: View on tanks inside pressure hull and view on bulkheads present in a Rhino model

The effort it takes to create an internal and external spatial arrangement of any hull can be judged on the basis of the number of decisions a designer has to take during the process. This method is used in this paragraph to estimate the needed effort.

The hull lines and the pressure hull inside and outside geometry are provided as input to the space partitioning process. These closed surfaces are single polysurface objects which include penetrations such as torpedo tubes, hatches as well as the primary stiffeners on the inside and/or outside of the pressure hull, if any. In any spatial arrangement there are four main types of topological objects:

- A divided compartment (any compartment that is further divided into smaller compartments);
- An undivided compartment (an actual compartment, a room, tank, etc.);
- An undivided divider (closed shape such as outer hull and pressure hull inside and outside geometry, a bulkhead, deck or wall, possibly tilted around their corresponding axes, both angles zero if orthogonal);
- A divided, or recursive divider, a divider that is cut or divided by its parent vice versa.

What is the minimum information you need to create any of the above topological objects?

1. Per divided or undivided compartment:
    a. Division ready = No (Undivided), Yes (Divided)
2. Per divided or undivided divider:

 http://doi.org/10.24868/issn.2515-818X.2018.005

a. Divider type: closed shape, bulkhead, wall or deck;
b. X, Y or Z coordinate;
c. Rotation angle 1 (0 if orthogonal);
d. Rotation angle 2 (0 if orthogonal);
e. Division ready = No (Undivided), Yes (Divided);
f. For undivided divider:
    i. Primary space orientation (see above)
g. For divided (recursive) divider:
    i. Keep divider part (see above)
    ii. Keep parent part (see above)

The example in Figure 14 contains the following spaces and surfaces:

- Divided compartments: 47
- Undivided compartments: 51
- Undivided dividers: 49
- Divided dividers: 29

Although most arrangements will be orthogonal for which both rotation angles of the dividers will be zero we consider the angles as input allowing for any orientation of the dividers. The total number of choices (excluding any further space or area related properties such as name, type, density etc.) will then be:
47 x 1 + 51 x 1 + 49 x 6 + 29 x 7 = 595 values

Of the above four topological types only the divided compartments are not used any further after their creation. The undivided compartments represent the actual spaces in the submarine, each defined by a closed polysurface. Both the undivided and divided dividers represent surfaces which form the complete dividers between compartments. Their area and position are input to the calculation of weight and centre of gravity.
In summary, to create a spatial arrangement containing 51 compartments and 49 + 29 = 78 surfaces we need to provide 595 input values. Since all compartments are derived from the dividers, on average only 595 / 78 ≈ 8 values are required per divider. These values include position, orientation, compartment hierarchy and the relations between dividers and compartments. Considering an unbounded plane in a 3D space already requires 6 values (single point and normal vector), this means on average only 2 values per divider describe the spatial relationships between dividers and compartments.

8. Conclusions

The SUPREME compartment definition presented in this paper is a solution to the need for both a surface and compartment based representation in submarine design. With SUPREME, an interactive and powerful 3D modelling methodology was created which integrates computational tasks in the design process. SUPREME results in well-structured and accurate geometrical CAD models which can easily be modified and used outside SUPREME for more detailed engineering. Important to submarine design, the weight and volume balance can be continuously updated providing the remaining budget in volume, weight and centre of gravity.

Currently, SUPREME is limited to submarine design application. However, its concept for space partitioning is by no means limited to submarine design. Although restricted in its capabilities and applications, the creation, partitioning and management of complex shapes and the concise weight management approach is considered an important improvement over traditional methodologies followed in early submarine design. SUPREME and the results it provides are expected to remain useful also in more advanced stages of design as explained in [van den Broek-de Bruijn, 2016].

 http://doi.org/10.24868/issn.2515-818X.2018.005

9. References

van den Broek-de Bruijn, W. (2016). Supreme - Submarine Performance and Requirements Evaluation Method. *Undersea Defence Technology (UDT)*. Oslo.

van Hees, M. Th.: Quaestor: Expert Governed Parametric Model Assembling, Doctors thesis, Delft University of Technology, ISBN 90-75757-04-2, February 1997.

van Hees, M. Th.: Knowledge-based Computational Model Assembling, Summer computer Simulation Conference, SCSC 2003, July 20-24 2003, Montreal, Canada, pp. 285-294.

van Hees, M. Th. (2009). Quaestor: taxonomy-based compositional modeling and product configuration. In S. E. Ovestad (Ed.), 10th International Marine Design Conference Trondheim, Norway: pp. 630– 647.

Koelman, Herbert J.: An approach to modelling internal shapes of ships to support collaborative development. Proceedings of TMCE 2012, May 7–11, 2012, Karlsruhe, Germany, Edited by I. Horváth, Z. Rusák, A. Albers and M. Behrendt, Organizing Committee of TMCE 2012, ISBN 978-90-5155-082-5 685.

Oers, B. v., Takken, E., Duchateau, E., Zandstra, R., Cieraad, S., Broek-de Bruijn, W. v., & Janssen, M. (2017). Warship concept exploration and definition at The Netherlands Defence Materiel Organisation. *Set Based Design Workshop,* (p. 25). Washington D.C.

https://www.rhinocentre.nl: website of RhinoCentre in the Netherlands

Takken, E and van Oers, B. MARIN plays an important role as the Netherlands studies future naval, MARIN Report December 2016

10. Glossary of terms

API: Application Programming Interface, exposes functionality of software to be accessed and controlled from third party software.
Binary Space Partitioning (BSP): a process of subsequent splitting of a closed surface by means of simple planar or more complex surfaces into smaller closed surfaces, each splitting operation should result into two new closed surfaces (spaces/compartments), in this paper referred to as the primary and secondary space;
CAD: Computer Aided Design, tools for 3D modelling, for this application Rhinoceros™ was used;
Compartment/space: Closed surface representing any singe closed space (in a submarine);
Divider: Surface or set of joined surfaces that splits a closed surface in two new closed surfaces;
DMO: Defence Materiel Organisation;
Entity; node in a taxonomy containing a list and or table of parameter value combinations, either with fixed values, user input values of computed values. If computed, also the expressions are included in the node. Values can also be an entity (hierarchy).
Instance: a copy of an Entity with input and computed values, if any, as part of a design workflow;
Knowledge engineering: the process of structuring design data and computational methods knowledge to enable its use through an inference engine or by instantiation into (in this paper) submarine design concepts or related analyses;
Primary space: first space resulting from binary partitioning a parent space;
Rhino: Rhinoceros™ , see www.mcneel.com;
Secondary space: second space resulting from binary partitioning a parent space;
Taxonomy: a hierarchy of entities representing an artefact, (computational) process or workflow;
T/E: the Taxonomy/Entity knowledge engineering concept as embodied in Quaestor3;
Topology: within the scope of this paper the relationships between spaces and surfaces and their references to the positioning of components through scaling, rotation and translation.

11. Glossary of terms

MARTIN TH. VAN HEES graduated at Delft University in naval architecture early 1983 and worked at Wilton Fijenoord Shipyard until late ’86. Since then he held several positions at Marin in Wageningen in ship powering, ocean engineering and software engineering. Next to ship hydrodynamics, his research interest is knowledge based systems in which he completed a PhD at Delft University of Technology in 1997. Currently he holds the position of software architect focussing on the Quaestor3 framework and its applications in design and analysis.

 http://doi.org/10.24868/issn.2515-818X.2018.005

WENDY VAN DEN BROEK - DE BRUIJN holds the position of Senior Naval Architect at the Maritime Systems Division of the DMO. She is involved with design tool development and design studies for the future Royal Netherlands Navy submarine replacement program. In 2007 she graduated at Delft University of Technology with her MSc research on a submarine design synthesis model.

 http://doi.org/10.24868/issn.2515-818X.2018.005