International Conference on Ships and Offshore Structures
ICSOS 2018
17 – 19 September 2018, Gothenburg, Sweden

An Open and Collaborative Knowledge-Based Approach for Automated Design of Hull Scantlings

Olivia S. Chaves[a], Henrique M. Gaspar[a], Henning Borgen[b]

[a]*Norwegian Univ. of Science and Technology, Ålesund 6009, Norway*
[b]*SINTEF Ålesund, Ålesund 6009, Norway*

Abstract

An open and collaborative web-based approach for automated design and redesign of hull scantlings is here proposed. Web-based is a trend for engineering applications due to its universal and easy-sharing properties, and the open source aspect promotes collaborative development and customization. This knowledge-based *app* is developed using open source web-based technologies and object-oriented parametric modeling. The parametric property of the scantlings model promotes efficient reuse of previous designs, given that there is no input that, once changed, will not automatically update the whole vessel scantlings; instituting the concept of unlimited editing possibilities. Also, the design process is enhanced by real-time feedback regarding the effects that every change causes in the structure, empowering designers with a more informed path towards an optimum design, bringing up more control and saving time when assessing design options. Although much more comprehensive and functional commercial programs that assist hull scantlings design exists, they contain a tremendous amount of proprietary information, which can only be partially accessed and processed, with limited functionalities to share designs and analyses. Designers with different needs in different companies with specific manufacturing facilities are usually using the same standard tools for scantlings, certified by classification society. If an open source tool is available, able to easily create an initial design and interface with more advanced tools, time can be saved on re-using and re-adapting current designs. A proof of concept is presented, where all basic steps involved in the process of defining a scantlings arrangement are somehow represented, as well as the basic design principles. Additionally, a knowledge base able to incorporate these steps, high-level structural elements and its relationships was created. For this concept, the focus is not on obtaining a classified design, but rather on the improvements that automation can bring to the process of achieving it. We close this paper with a call for more research in open and collaborative standards for structural design, with a reminder that rather than competing with actual state of the art commercial software, our approach intends to develop a supportive and easy to interface platform for fast and reliable structural design during early stages of design. The version discussed in this paper is in its initial stages, but custom features could be added to the system for including specific shipyard metrics and processing any data valuable for a specific user.

Keywords: Knowledge-based system; hull scantlings; web-based open source; automated structural design

1. Challenges and Opportunities in Ship Structural Design During Early Stages

1.1. Feature-Based Measurements and Implicit Design Information

The ship design data, the way it is today, is not meant to be reused (Erikstad, 2009). Traditionally, vessels are quite unique, and shipyards try their most to attend every requirement imposed by shipowners, preventing the industry to develop itself towards standardization, modularization and reuse, as other industries (e.g., automotive and computer) had done decades ago. With respect to ship structure, adapting previous projects may require extensive recalculation, depending on the modifications extent. Redesigning scantlings in the currently available software programs typically requires editing (e.g., updating profile, size and/or position) hundreds of elements manually and one by one.

The hull structural design has large influence on the spaces inside a vessel, lightweight, costs related to material and construction labor, among others. Such influences might not be easily assessed in early stages, if not due to the lack of tools, probably because of the information decentralization, where each tool (or personal spreadsheet) assesses one single aspect of the design (e.g., ship weight, total cost), or even because of the lack of valid reference vessels and/or data, especially when designing one-of-a-kind vessels.

Caprace and Rigo (2012) discuss a feature-based approach for estimating the cost in ship design. The method assumes that a product, that is the ship structure, can be described as a number of associated features such as material, cut-outs, welding length; and each feature has an associated cost. In comparison with other methods (e.g., statistical, parametric), this one may produce *safer* results even when no reference vessel or data is available. The downside, however, is the amount of information/details necessary to make the method accurate, which can be a challenge for system which information is only specified at late design stages, but it seems suitable for system where a list of all (major) elements is given from the beginning, such as scantlings.

By using modern objected-oriented programming tools (Gaspar, 2017), these features can hold its individual properties, such as cost and working-hours, and easily be associated to structural elements (e.g., plate, stiffener), providing a practical control over cost drivers, as well as any other feature-based measurement. Feature-based costing is not an innovative method, and some specialist have been using it in their spreadsheets for decades, but certainly with a lower level of details and cumbersome updates and versioning control. Ship structures can easily contain dozens of thousands of structural elements, which can contain hundreds of individual properties each; it is challenging to keep control of this number of elements without using dedicated software, consistent database and/or some programming. Designers simply cannot assess designs based on information that they do not have. By including into the scantlings tool information about other aspects of the design (e.g., weight, production metrics), a path towards a holistic approach is favored, and the cost of technical decisions (e.g., topology) can be measured through numbers instead of agued based on experience and/or an educated guess.

Tools used for generating hull scantling contain sufficient information for assessing more than they usually do, but this knowledge cannot be easily extracted, combined and/or post-processed in an efficient (automatic) way. For instance, these tools comprise a log of structural elements with profiles, dimensions, and material assigned to them, why can't the tool indicate the *weight* of a given scantlings with a simple *volume*density* calculation? or the *total surface area (painting area)?* Or *the welding demand?*

Given that the hull structure is the main component of the ship lightweight distribution, and the steel structure is one of the main cost drivers, a proper estimation of these numbers earlier as possible is paramount. It is understood that some estimates are more complex to get it right than others, however, no precise absolute number is required. Park *et al.* (2015) defends that, in optimization, the relative difference between parameters' values is more relevant than the accuracy of the parameter itself. Knowing the relative difference of pertinent parameters is already a progress, and it is assumed good enough for preliminary design stages. When assessing design option, having a reference number for comparison purpose about design aspects (other than compliance) enhances the design process and can lead to optimization.

1.2. Drawbacks Regarding Current Available Software for Scantlings Design

Undeniably, much of the real structural knowledge is experience-based, owned by experts, working in different design companies, which are all using the same (usually classification society approved) commercial software tools. By doing so, experts are constrained regarding how they can improve their work practices and extract knowledge from these tools, since they are all proprietary and black-boxes. These tools' knowledge bases are gigantic, but they cannot be accessed and/or expanded to include custom options or post-processing in a specific way that is useful for a specific client.

For a fact, the primary function of tools that assist scantlings design is to check compliance with classification rules. Besides the facilitation in the approval process, one of their greatest advantages is the support for correctly managing the combination of design loads scenarios, cases, and acceptance criteria to be both applied and assessed. It is understandable that a software designated to compliance check (with such a small number of users and no competitors) does not have design automation or optimization as top priorities.

Nevertheless, some of these tools have lately incorporated *"design wizards"* able to generate a scantling design for a given cross-section based on minimum requirements. However, this automation is still limited to a single initial step, rather than providing support throughout the entire design. If a rule is

well defined inside a knowledge based systems, why would the system let the user design out of the allowed range and further in the analysis, reprove it? Also, why can't the system automatically assign minimum allowed values when an element is created, instead of asking for an input every single time? Modifications are only computed when the *run analysis* button is pressed, often recalculating things that were not influenced by the latest design changes, therefore taking more time to provide results (feedback) and not making it clear which modification had which influence at which aspect of the design.

Obviously, there might be other requirements that may lead the user to modify (increase only) the automatically assigned minimum values, but undoubtedly there will also be plenty of cases where no change is required, and a significant amount of work is saved. When the design process initiates with a more restricted design space, playing around with parameters, to measure the effects in the overall structure, becomes more efficient, and no time is spent assessing options that are not technically allowed.

Besides, the design wizards' assistance is questionable, given that they eventually generate a design where elements are too close to each other, especially towards structures' extremities, which must be later manually corrected (i.e., deleted, recreated and assigned properties) anyway. For this reason, depending on the cross-section complexity, some designers may prefer not using it.

About parameters updates, there is a limitation regarding the chain reaction triggered by these adjustments, which can possibly cause design inconsistency. For instance, if main dimensions are modified (in the main ship data section), no update in the scantlings occur. That is, it is possible to have a vessel which maximum beam is specified as 20 meters (value used to calculate hull girder loads to be applied in the cross-section), and in the same project, a scantlings design with a 30 meters wide cross-section. In this way, the update process is kept entirely manual, being the designer the responsible for keeping track of all consequential actions to be taken after modifying a parameter, thus, hampering the reuse of previous designs.

Figure 1 exemplifies this limitation, with a screenshot from DNV's Nauticus Hull. Let us assume that a vessel is first defined with 240 meters length, the frames naturally distributed across the length, and the whole scantling fully defined. If the length is later modified to 180 meters, nothing happens with the rest of the design, except for inconsistence generation (as observed in Fig. 1).

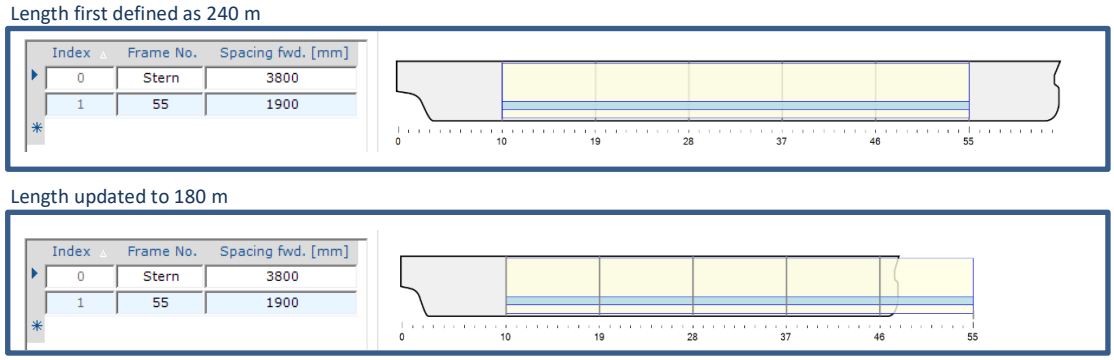


Fig. 1. Nauticus Hull project data before and after parameter update – bulkheads and scantlings are not updated when overall length is changed

1.3. The Potential for Automating Design Tasks and Better Update Support

The potential for automating design tasks lays on managing elements' relationships, managing design updates, automatic selection of elements properties, and expanding the rules database (or identifying the lack of them). There are classification rules covering elements' intersection (e.g., cut-outs), connectivity (e.g., welding) and continuity (e.g., longitudinal/transversal), which could be incorporated in the scantlings tool. For instance, if the maximum allowed cut-out depth is 40% of the Primary Supporting Member (PSM), the system could indicate when a Secondary Supporting Member (SSP) bigger than that is to intersect the PSM through a cut-out. Another example, the system could make sure that longitudinal elements supposed to be continuous are in fact continuous, in a way that modifications in the element's position in one cross-section would lead to automatic updates in another cross-section.

Modern program languages, such as JavaScript' event-driven ability and prototyping methods, allow for easy-handling of actions to be taken (updates) after every single change in the design, not limited to any group of parameters. As mentioned, changes in the vessel's main dimensions are especially tricky to manage, as they have influence in most other parameters of the design (including the position of all elements), but with the modern object-oriented language features, the update management is facilitated,

given that a list of rules which a structural element should be checked for could be incorporated in the object, with a method with the corresponding analysis result. In the same way, the lack of applicable rules could also be identified, being especially useful for innovative designs. The current rules for classification are mainly, but not only, prescriptive. That is, they mostly define methods to be used in order to achieve acceptable safety levels (Bergström *et al.,* 2016), but provided sufficient documentation, other methods such as goal-based design are also accepted.

Goal-based rules directly define the acceptable safety level to be achieved, leaving the means to reach it more open. It can be used either when no applicable prescriptive rule is available, or simply in preference to the former. One of the approach's advantages is the space opened for new methodologies, and a downside is a possibly demanding search for feasible/optimum design option, as many analyses must be run. However, if this search process is automated, certainly introducing a higher degree of complexity than automating a prescriptive approach, the hazard could be avoided, and the process enhanced.

Finally, when automatically creating structural elements such as filling up the cross-section, an alternative approach to prevent elements to be placed too close to each other, or elements collision, would be creating altogether generic elements that hold a position property, and later distinguish (specify) them in different elements' types (e.g., stringers, frames) by adding other properties (e.g., profile, dimension, direction) to them. Also, when individually created, elements could retrieve the current state of the structure such as surrounding elements and local loads to automatically assign (or suggest) properties for the newly created element.

2. Knowledge-Based Approach for Early Structural Ship Design

2.1. Knowledge-Based Engineering for Automation of Repetitive Tasks

The fundamentals of Knowledge-Based Engineering (KBE) consist in automating repetitive non-creative design tasks in order to save resources. By using KBE principles for basic ship structural design, we aim *to reduce time and cost of product development, which is primarily achieved through automation of repetitive design tasks while capturing, retaining and re-using design knowledge* (Verhagen *et al.*,2012).

KBE is typically assisted by knowledge-based systems (KBS), which define designs with sets of rules (Fig. 2), rather than specific data instances, as conventional CAD systems do. In doing so, KBE systems allow the replication of design properties into large classes of similar objects. The knowledge embedded into KBE systems is varied. In addition to the designing rules, there is also the knowledge derived from people, rather than the product itself, such as professional knowledge, domain knowledge, users' maturity experience and users' feedback. All this knowledge embedded into a KBS allows the software to perform logic judgments and deduction, generating an intelligent (possibly optimal) design at top speed (Yang *et al.,* 2012; Cui & Wang, 2013).

Rule 1	*If*	A	*then*	B
Rule 2	*Either*	C	*or*	D
Rule 3	*For all values above*	F	*do*	G
Rule 4	*All that contains*	H	*must*	I

. . .

Fig. 2. Typical KBS's set of rules.

Defining hull scantlings is one of the underlying processes of the initial structural design, and the initial knowledge that we tackled to be incorporated in our open knowledge systems. This process involves tasks for determining the sizes and profiles of structural elements that will compose the ship structure, and it can be done by direct calculations to achieve sufficient strength. Once the structure experience loads, that can be either punctual (e.g., corners of containers) or distributed (e.g., sea pressure), these loads will be transmitted (dispersed) form one structural element to another, so that the overall structure resists the stresses caused by these loads. Therefore, when determining hull scantlings, it is fundamental to understand:

1. The elements composing the structure;
2. The difference between local and global loads;

3. Which element will suffer which load;
4. How the elements are related (fastened) to each other, which will consequently determine how the load will be transferred from one element to the next one.

The ship structure is a single solid piece of steel, which is made of several smaller structural elements (Fig. 3-a). The definition of structural elements and their nomenclature are quite broad, and not always concise neither consensual. For example, center girder and keel are two names that refer to the same element. In the rest of this section, we briefly present the way that structural elements are understood in the KBS presented in this work, as well as how they will be called and distinguished from each other, which focus more on the elements' structural function and basic characteristics, rather than how they are ordinarily named.

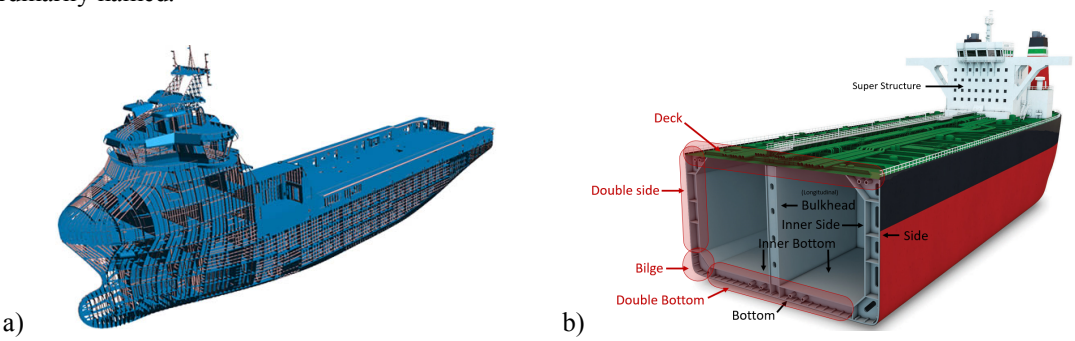


Fig. 3. Entire ship structure (except for shell plating) welded together into a single piece (from CADMATIC software) (a); Main structures that compose a ship structure (b).

2.2. Structural Elements

The terminology 'ship structure' refers to the whole ship structure as a single unity. It can be divided into main *structures*, that is, bottom structure, side structure, deck structure and bulkheads (Fig 3-b), which can be further divided into bottom and inner bottom, side and inner side, main deck and deck 1, deck 2 … deck N, and transversal or longitudinal bulkhead 1, 2, … N.

At its most basic instance, a ship structure can be represented (formed) only by plates and stiffening elements, in a way that all structural elements contained in a hull can fall into one of both categories.

2.3. Plates and Stiffening elements

Plates are elements defined by its geometry. Every single rectangular prism which height (thickness) is very small when compared to its other dimensions (breadth and length) is a plate. Also, the loads that are expected to be applied on it will only generate stresses whose resultants are, in practical terms, exclusively normal to the element's thickness (Timoshenko & Woinowsky-Krieger, 1940).

Stiffening elements is a term used to refer to any structural element added into the structure, except for plates itself, that culminates in increased strength of the structure. They can be subdivided into two groups according to their function (primary and secondary supporting members), and named in diverse ways, depending on where they are located within the structure (Fig. 4). Examples of stiffening elements are stiffeners, girders, frames, stringers and brackets, which can be seen in Fig. 5.

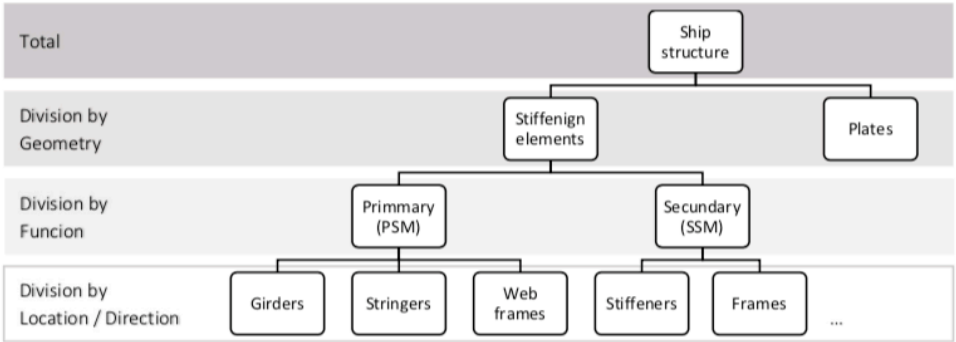


Fig. 4. Hierarchical breakdown structure of hull structural elements categories.

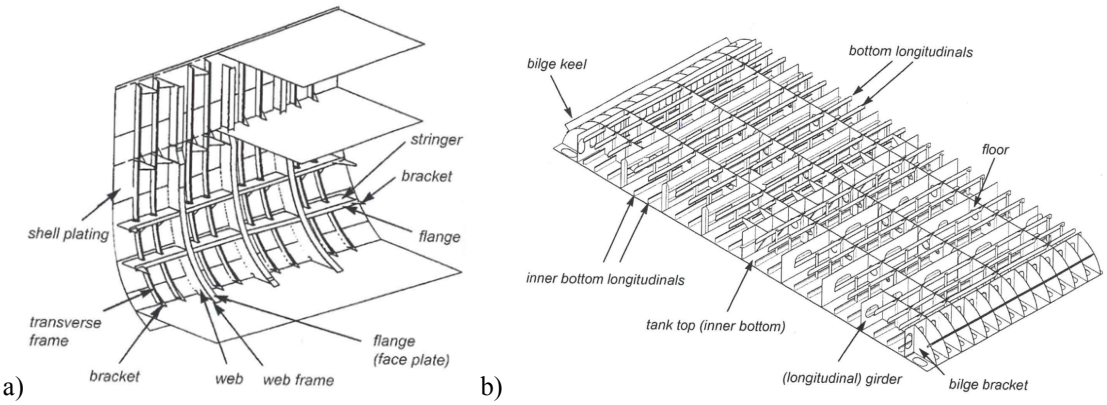


Fig. 5. Side typical structural elements (a) and double bottom common elements (b) (Fricke, W., 2016).

To not leave the consensual nomenclature aside and situate fundamental elements into the presented point of view, some 'types' of elements are briefly discussed in the sequence.

- *Stiffeners*: elements defined by their structural function. They are SSM, whose main function is to support the plates and transfer/disperse the loads to crossing PSM. Typical profiles for stiffeners are, e.g., flat bars, bulb and (unequal) angle bars. These elements may receive different names depending on their direction and/or which structure they are part of. Stiffeners which are continuous across the vessel length (longitudinal direction) are called *longitudinals*, and they may get prefixes depending on where they are located. For instance, longitudinals located in the bottom region are called *bottom longitudinals*.
- *Girders*: fundamentally primary supporting members. They can be longitudinal or transversal and can be found in all structures (deck, bottom, sides, and bulkheads), but with different names. They usually have a T-bar profile or are flat plates that connect two sides of a structure. For example, side girders connect the inner and outer plating of the double bottom structure, and stringers connect the inner and outer plating of double side structures or two sides of a bulkhead. A vessels' keel is not more than a longitudinal girder placed at the bottom on the center line, and it can also be called center girder.
- *Frames*: elements that run continuously in athwartships direction, responsible for maintaining the hull transverse form. They can be distinguished between frame and web frame, and the essential difference between them is their size (Fig. 5). Web frames are larger than frames, act as a PSM (similar to girders), and are spaced in multiples of the frame spacing, which are SSP (similar to stiffeners). Frames and web frames elements receive different names depending on the location they have, for instance, at the bottom region they might be called floors.

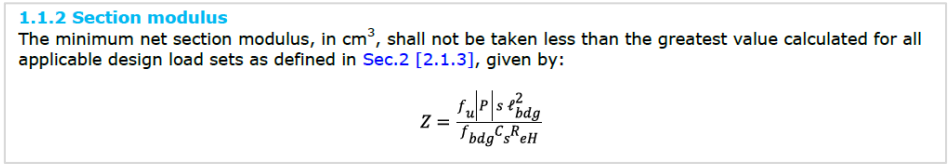
2.4. Design Loads and Constraints

Local loads are understood as the ones acting on a single structural element or a set of them, but not on the whole ship structure, for instance, the pressure due to sloshing inside a small ballast tank will act only on the tank's walls. On the other hand, global loads (or hull girder loads) are the ones acting on the entire structure. As a vessel typically has a slender shape, it will behave like a beam in waves. In doing so, the waves will cause stresses (and/or deformation) across the whole vessel extent. Examples of global loads are, therefore, wave bending moment and wave shear force.

Design constraints are just one more set of designing rules. They reduce the design space by defining boundaries that determine what is allowed or not. There are a few types of constraints (e.g., physical, economical), and they may be optional and/or vary for each design. For instance, rules (constraints) imposed by a given classification society (Fig. 6) do not need to be followed if a vessel is not supposed to be approved by this society. For this work, three types of constraints were defined and are discussed: design principles, constructions (yards) and experience based (best-practices).

Constraints imposed by *design principle* are highly recommended to be considered (at least for standard designs). They are fundamental and the basis for all others. For instance, by principle, when a load acts on a structure, it has a defined direction of propagation and must be carried by specific members. PSMs should support SSMs, which should support plates, and this is pure and fundamental physics. Classification societies establish their constraints based on a compilation of design principles, extensive

theoretical studies, experience, etc., and they are very well defined. For instance, at DNVGL-RU-Ships 2018 Pt.3 Ch.6 Sec.4, it is imposed that stiffeners subjected to lateral pressure should not have section modulus taken less than a given value (Fig. 6).



1.1.2 Section modulus

The minimum net section modulus, in cm^3, shall not be taken less than the greatest value calculated for all applicable design load sets as defined in Sec.2 [2.1.3], given by:

$$Z = \frac{f_u |P| s \ell_{bdg}^2}{f_{bdg} C_s R_{eH}}$$

Fig. 6. Example of a DNVGL rule for section modulus of stiffeners.

This means that there is no reason for a KBS to let the user explore design options that do not comply with this type of constraints. It is just a waste of time. Yet, commercial currently available KBS(s) for assisting hull scantlings design allow the user to (blindly) choose stiffeners with section modulus under the local requirement, which will be later accused as 'not allowed', and it will have to be changed. This is a common loop that structure designers go through frequently.

The design space is also constrained by the manufacturing technology available in the *shipyard*. Designers must keep control of this constraints by themselves since standard commercial programs generally have poor ability to handle customization. For instance, a plate with 50 mm of thickness might accuse no issues with respect to all applicable design rules/constraints (e.g., minimum thickness, bucking capacity) in the software, however, the shipyard might not have the capacity to cut it. This set of rules is very particular for each shipyard, and if implemented in a KBS, they should be placed separately from the standard ones, in order to facilitate updates and customization for other clients.

Also, in terms of manufacturing, there are constraints related to the accessibility, support, and sequence of construction. For instance, confined spaces that are supposed to be welded from the inside, cannot be too small so that no human or mechanical arm can reach that place. As far as the software does not accuse this kind of complications, the designer is the responsible for monitoring this type of constraints.

When initiating a new project, some variables might be considered fixed (constrained) just for the sake of reducing the design space and establishing a ground for initiating the design. Frequently, they are just given a value (based on experience), and the rest of the project turns around it. These are considered *best practices* constraints. For instance, parameters such as typical stiffeners and frames spacing are somehow fixed within a design office for a given type of vessel. In practice, they typically fall within the same range (stiffeners' spacing between 600 mm and 900 mm, and frame's spacing about 3 to 5 times the spacing between stiffeners), and at the beginning of a project, they are established merely based on standard practices.

The difference between this type of constraint and the previous ones is the level of flexibility. They can be initially fixed not because the yard is not able to manufacture structures with different spacings, and not because the classification society does not allow variations on these parameters. It is simply related to external factors (e.g., construction practice and cost) and easier for everybody if it remains the way it is. This characteristic has an impact on how this type of constraints will be implemented into the knowledge base, given that they constitute a different group of "preferably, but not ultimately, fixed" rules. If there is a good enough reason to change a constraint, it should be allowed, differently from the previous types, which should be ultimately forbidden.

2.5. Challenges and Limitations

Reports from classification society proprietary software (e.g., DNVGL Sesam) have a log of the design scantlings, loads acting on the structure and the results of the structural analysis, stating whether the elements will withstand the loads or not. The conclusion is given after the software tests all elements for all applicable rules that it contains (using its knowledge base). However, even if a structure design is made using the classification society proprietary software and documented with reports from this same software, stating that the design has sufficient strength, it does not mean that the design will be automatically approved. Engineers working with approval in the classification society will do hand calculations on top of the software results to verify them.

This human verification is necessary (not only but also) because not even the software developed by the same society who made the rules is able to handle (implement) all designing rules and constraints in their own software, showing a gap and slight lack of reliability. The point is not criticizing but expressing

the complexity of developing a KBS that covers 100% of the existing design knowledge, given that not even the biggest players in the market have managed to do it yet. The challenge even bigger when addressing innovative vessels, where the prescriptive KBS fail to support the design processes and verification, as these vessels include new unprecedented solutions (structural arrangement), not falling into any of the existing rules (categories), thus requiring custom-made verification.

The existing rules are already extensive[1] and only tends to grow. They are expanded whenever practical cases provoke learning, which means new knowledge, and technical committees agree that this knowledge should generally be applied. On top of that, there is also the unmeasurable amount of designers' experience that is quite difficult to settle together. Constant updates of the knowledge base require a tremendous amount of maintenance work. An open and collaborative approach for the maintenance of this KBS seems beneficial to all players.

3. Open Source Approach – Hull Structure as an Object

3.1. Open Source and Web-based Technology

Open source is a reality for maritime applications (Gaspar 2017; 2018), standing for a transparent and collaborative platform, based on free access to the software code, allowing users not only to see but also modify it. The maritime sector is not apart from the latest digital transformation trends either. It is possible to name commercial initiatives such as *My DNVGL*, which is an online hub for services, where customers can find a compilation of multi-purpose Apps, developed by DNVGL and third-party suppliers, for industrial applications, analytics, and data sets. Although these Apps are not open source, they are web-based (run in a web browser), and some of them are free to use.

In the past few years, the Web has been growing at a much higher rate (development over time) than engineering programming style has in the same period. Browsers engines, such as Google V8, have been both following and driving internet's growth rate, becoming today extremely optimized, and performing (in most of the cases) with an excellent cost-benefit when it comes to simplicity versus computational power, providing a flexible, fast-growing and efficient platform for engineering use.

Last but not least, interfaces designed for the Web are significantly more user-friendly than conventional software interfaces. Web pages must be visually attractive in order to hold user's attention, while a software that has unique capabilities, and few or no competitors, will be used anyway because of its functionalities. The more competition a commercial software has, the more investments in usability it does. Besides, making a good-looking web-based app is not demanding, there are free-to-use libraries available online with a user guide that are simple to implement and maintain.

3.2. Hull as an object of the Ship

One of the fundamental concepts from object-oriented programming is the inheritance concept. Usually, in a class-based programming language, objects are instances of classes, from which they can inherit properties and functions. In JavaScript objects inherit from special objects called prototypes. JS also allows encapsulation, enclosing all the functionalities of an object within that object so that the object's methods and properties are protected from the rest of the application, allowing particular properties and functions to be specific within the boundary of the object. These two concepts allow the build of applications with reusable code, scalable architecture, and multiple functionalities (Gaspar, 2018).

A vessel and its subsystems can be thus defined as objects (within objects), with general (pre-defined by the library) and customized (defined case to case) properties and methods. Such freedom in establishing a taxonomy is one of the key elements of web-based KBS to handle complex hierarchical structures. Fig. 7 presents the idea of a ship as an object, with properties like *Name* and *Length*, and methods such as *Sail()* and *Anchoring()*. This basic principle is applied for hull structural design, detailed as follows.

---

[1] DNVGL RU for Ships (2018), for instance, contains 5.294 pages of written designing rules, only for ships.

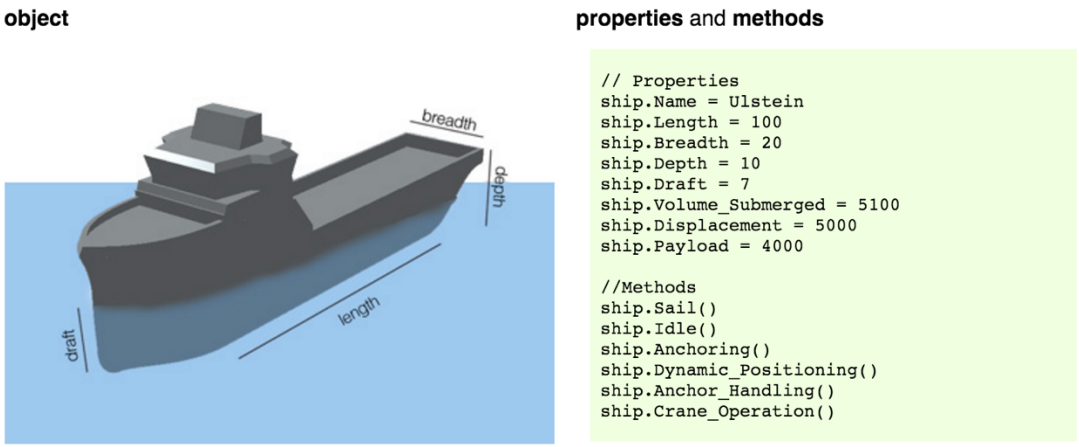


Fig. 7. A ship as a JS object, with properties (e.g., name, length) and methods (e.g., sail, idle) as basis for the Vessel.JS library structure (Gaspar, 2017).

When people communicate, they use words inserted in a subjective context which carry implicit significance. For instance, when someone in a shipyard says 'side girder', it is possible to infer that:

(1) this person is talking about a structural element,
(2) this element is longitudinal, and it is located at the bottom of a vessel,
(3) this element has a greater height than width,
(4) this element is not smaller than other elements surrounding it,
(5) this element will be fastened to other elements by continuous weld (assuming steel construction).

However, a computer system has no contextual perception to pick up significance from; so it needs to be told. The way objected-oriented languages work is through key-value pairs. A key can be seen as a password, and a value is a piece of information. When they are set together in pairs, the key works as a password used to access the respective piece information (value) assigned to it. Inside an object structure, keys might be used to assign a name to an object or to a property of this object.

Marine terminology is used to identify the different elements, and this terminology (not always consensual) is the one used to apply rules and check the structural performance of a design. However, it has been pointed out that for this approach, it is the properties of an element that defines the element itself, not the element's given name. So, if this terminology can be translated into properties, the rule check could be done accordingly.

That is, if a rule is to be applied to *side girders*, it could be translated to the computer language as to be applied for objects (elements) with the properties: 'PSM, 'longitudinal', 'located at the bottom', among others. Given the need to specify elements and translate significance between both languages, a taxonomy was established. This taxonomy is used for capturing information about the structure and will have a large influence on how the computer is able to understand each of the elements, as well as the connections among them.

The taxonomy takes into account the elements' properties, which might be common among some elements' types or unique for a single type. Examples of common properties for every single element are:

- A set of criteria it has to fulfill (e.g., acceptance criteria I, II, III or IV)
- A set of rules that it has to be analyzed for (e.g., design load scenarios and cases)
- Geometry properties:
  - · center of mass;
  - · second moment of area;
  - · length;
  - · profile (e.g. bulb, angle bar, T-bar);
  - · section modulus;
  - · volume;
  - · cross sectional area;
- Weight;
- Cost;
- Limit state;
- Unique id (e.g. name or SFI code);

- Position (x, y, z);
- Material (NV-NS, NV-32, NV-36, etc);

4. Open Source Platform for Hull Structural Design during Early Stages

4.1. Defining an Open Source Knowledge Base

For organizational and efficiency purposes, the knowledge base has been segmented into three groups: *structural design rules, database and other rules* (Fig. 8). Most of the rules are those regarding structural design, but it also contains a database of materials, elements profiles and multipurpose functions not specific to structural design, which are used by the other set of rules when requested.

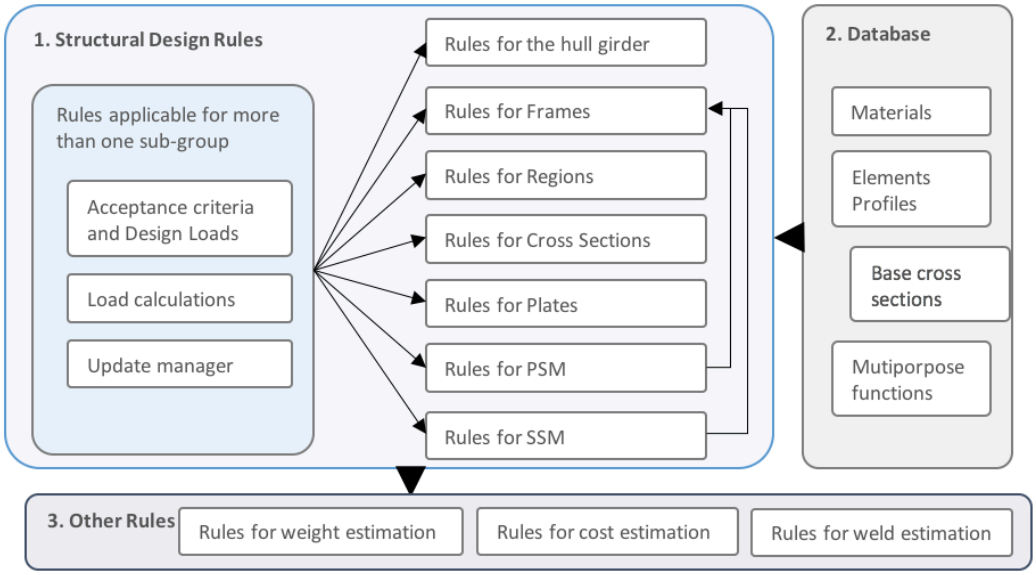


Figure 8 - Knowledge base structure developed

Additionally, there is a library of parametric cross-section shapes (Fig 9.), which refer to the hull outer shell. Once a shape is chosen, double bottom, double side, decks, and bulkheads can be added into it, and their positions set. Since the cross-sections are parametrically defined, they have the capacity to adjust themselves to variations of the vessel's main dimensions in the middle of the design process and update the (so far defined) scantlings accordingly.

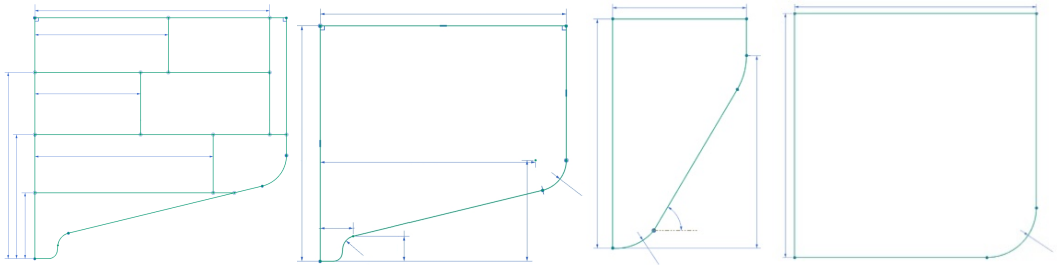


Fig. 9. Example of the database of parametric cross section profiles developed.

4.2. App Workflow, Map of Action and GUI

The App's workflow (Fig. 10) carries out the design process iteratively until the user is satisfied. The methodology consists of 7 steps, and it is based on when the App is initiated, it triggers steps 1, 2, 4, 5, 7 and back to 2 forming a loop until the answer of question 1 (Q1) is positive. This leaves the user at question 2 (Q2) with a ready-made *approved design* to start with, that is, compliant with the rules existing in the knowledge base. When the design is first generated, it uses pre-defined values (e.g., L, B, D, design load scenario, number of regions, cross section-structures, etc.) for creating the objects.

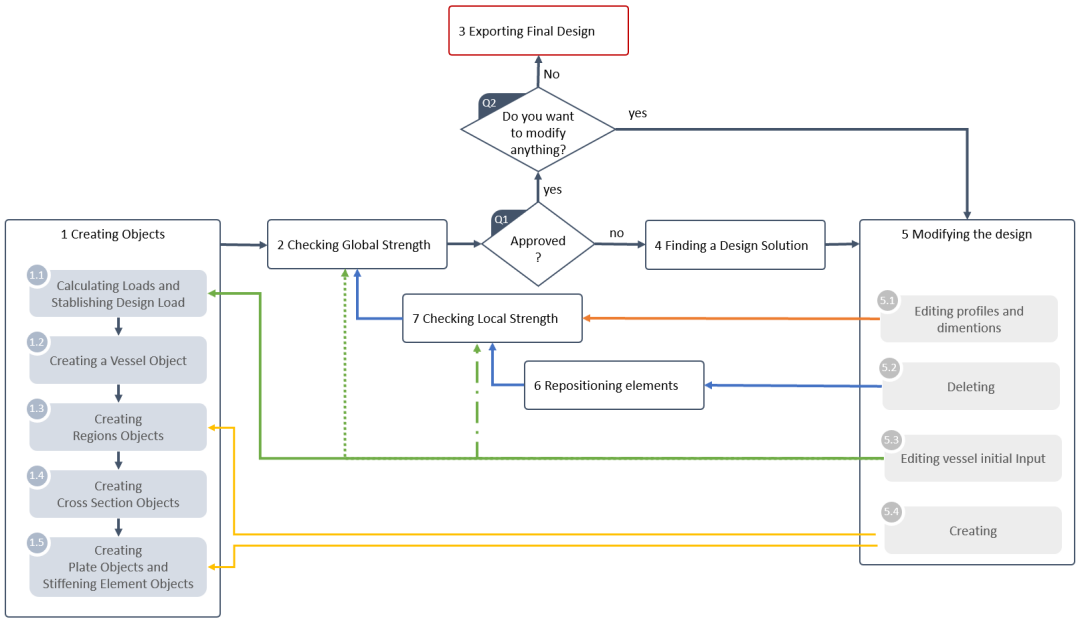


Figure 10 - App workflow

The integration of all parameters into one single up-to-date object is assured by *maps of action* (Fig. 11). Every input has its own map of action describing what needs to be done, that is, recalculated, repositioned, redrawn, reanalyzed and given a feedback, after the input is modified. These maps are event-driven, not requiring a user command for being executed, they are automatically trigged as soon as the parameter is modified. They also avoid unnecessary calculation of design aspects that are not influenced by the modification.

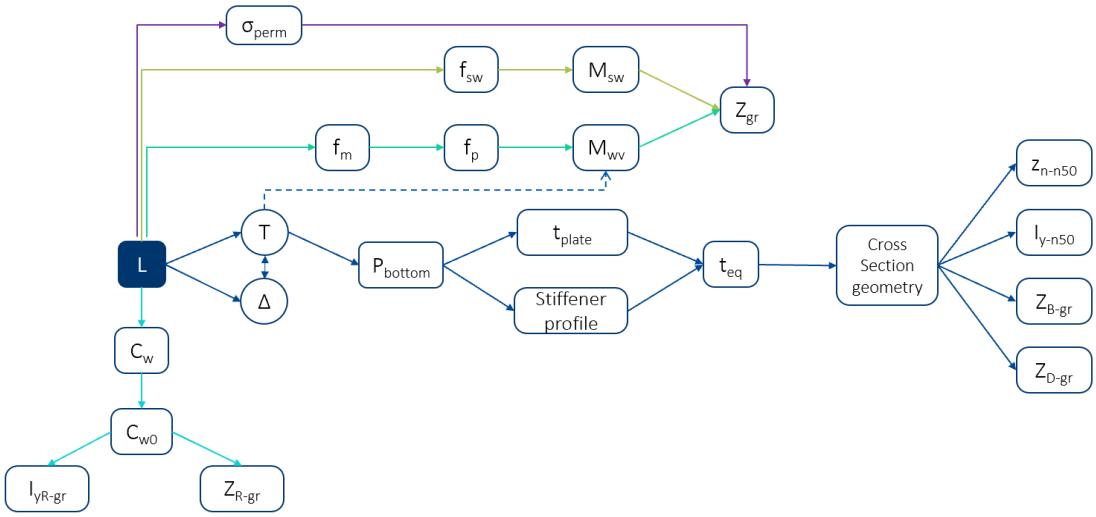


Fig. 11. Parameters to be updated with respect to the hull girder model when the length of the vessel is updated

An overview of the App's Graphical User Interface (GUI) is presented in Fig. 12. The first section is dedicated to inputting data, allowing the user import information about the vessel and/or manually edit it. The second section is for information purpose only. There, hull girder loads are displayed in a table and in a graphic. The third section combines a 2D visualization of the hull girder cross section, specific information about the scantling for each of the structures contained in the hull girder, and finally local and global structural assessment of the model. On the fourth section, the user is allowed define different cross section for different longitudinal extensions of the vessel (regions) and have a more detailed

scantlings design. Finally, together with the App's footer, there is an option for exporting the generated design. A more detailed explanation of the Apps functionalities is provided in the case studies.

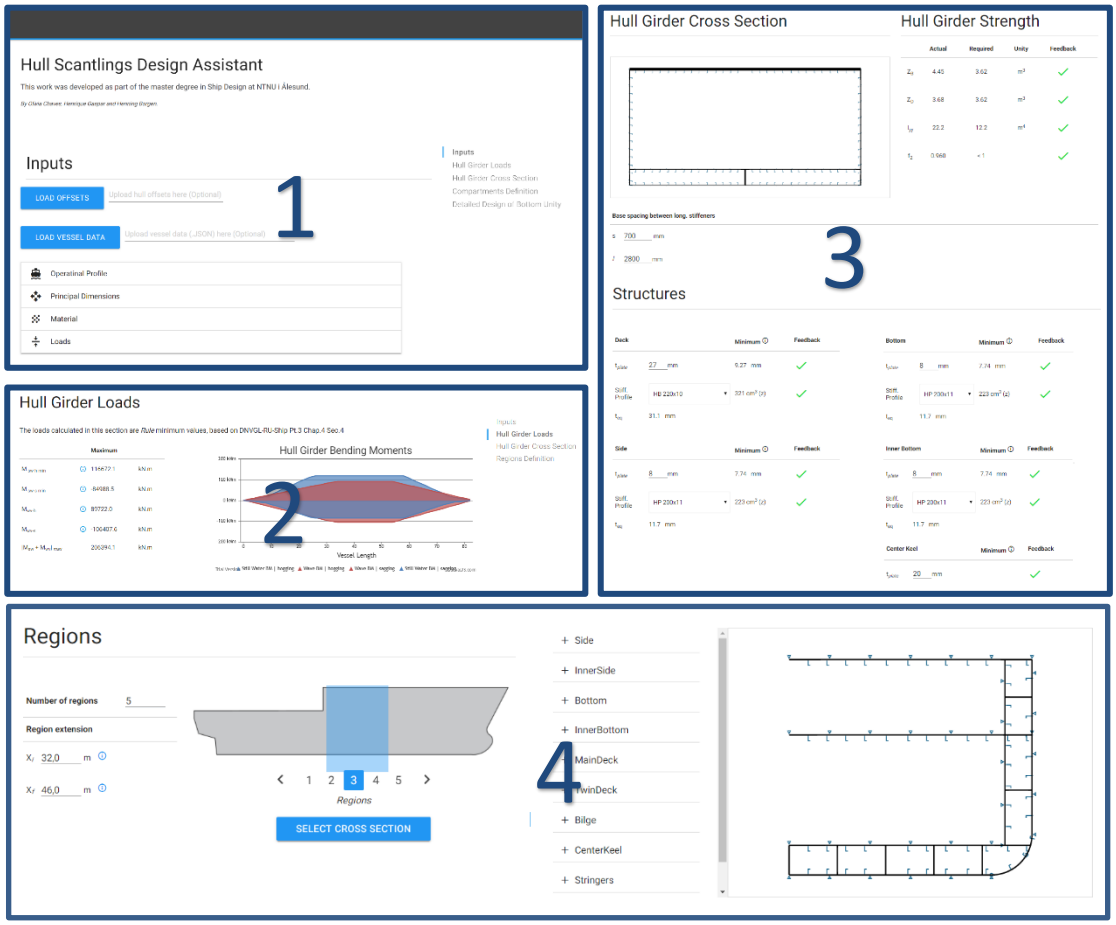


Fig. 12 – App GUI in four sections

4.3. Usability Enhancements

Aiming a better instructed path for assessing design options, instant feedback was incorporated into the App. The idea was exemplified for plate thicknesses and section modulus, and it can be expanded to any parameter which have a clear and straightforward benchmark for approval, such as those two. The requirement is calculated and indicated next to the actual applied value, and in case the last is equal to the first or higher, the feedback receives a green check mark, otherwise it returns a red mark (Fig. 13).

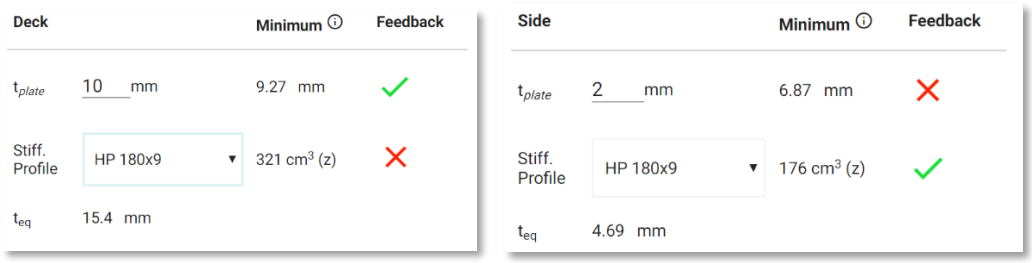


Fig. 13. Approved/Not approved real-time feedbacks

Also, when selecting elements properties, the App assigns different formatting for options that if selected will attend applicable criteria and elements that will not. This feature is also applicable for parameters which have a clear and straightforward benchmark. For instance, when selecting an element's profile from the drop-down menu, elements with insufficient section modulus will appear in light gray, while those which pass the criteria will appear in black, as shown in Fig 14.

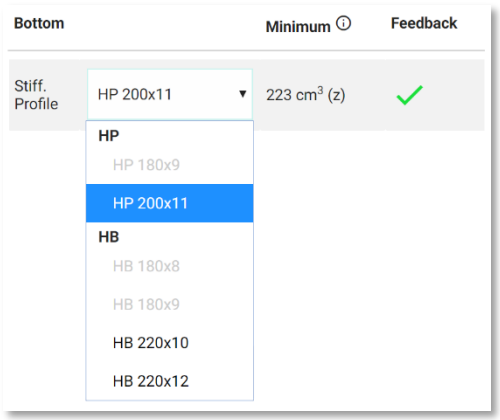


Fig. 14. Drop-down selector for stiffeners profile

5. Proof of Concept and Case Study

5.1. Proof of Concept – Stepwise Process for Structural Design

This section is dedicated to validating the approach and demonstrating the Apps' scalability. Although the most advanced version of it does not generate scantlings with sufficient level of detail for being classified, it does generate a design that is automatically created, rule-checked, and updated based on the implemented rules. The App's current state incorporates all basic categories of element previously discussed, and provides a framework with potential to be expanded, as demonstrated next.

Simple Case 1: The study began with the simplest possible representation of a vessel: a floating box. The intention was to verify if, based on this geometry, it would be possible to:

1. Calculate hull girder loads (bending moments);
2. Calculate parameters that indicate the hull girder (box) strength (Z_B, Z_D, I_{yy}, and f_2)
3. Establish criteria (lower limit) for the hull girder strength parameters to be approved ($Z_{R\text{-}gr}$, $I_{yR\text{-}gr}$ and $f_2 < 1$);
4. Perform a rule check to verify if the hull girder model passes given criteria and provide instant feedback about each one individually;
5. Updates parameters (L, B, T, D) and automatically update hull girder loads (bending moments), section properties (Z_B, Z_D, $Z_{R\text{-}gr}$, I_{yy}, f_2) and strength criteria ($Z_{R\text{-}gr}$, $I_{yR\text{-}gr}$).
6. Find the minimum thickness for the box (representing t_{eq}) that would make the model pass in all criteria, in case any of the criteria is not satisfied.

So far, all listed steps could be simply achieved with other engineering basic tools, such as Excel, except for step 6 that would require some macro or coding, but it is definitely feasible. This experiment also established the minimum level of information about the vessel (e.g., L, B, D, T, material, service area notation) required for accomplishing these tasks. Figure 15 shows the experiment results.

Simple Case 2: Provided the hull girder model from simple case 1, simple case 2 aims to verify if it is possible to:

1. Divide this box into different structures (deck, bottom, and sides), in a way that each structure can have its own properties (e.g., material and plate thickness);
2. Add more structures (e.g., double bottom, and center keel), under the same rules as the existing ones;
3. Consider different loads acting on different structures;
4. Consider different design load scenarios (static and static + dynamic);
5. Calculate the equivalent thickness required for each structure based on its properties (loads, cross section and material);
6. Stablish a standard plate dimension (a, b) and further divide these structures into plates, which will have their own position, dimensions and loads acting on it;

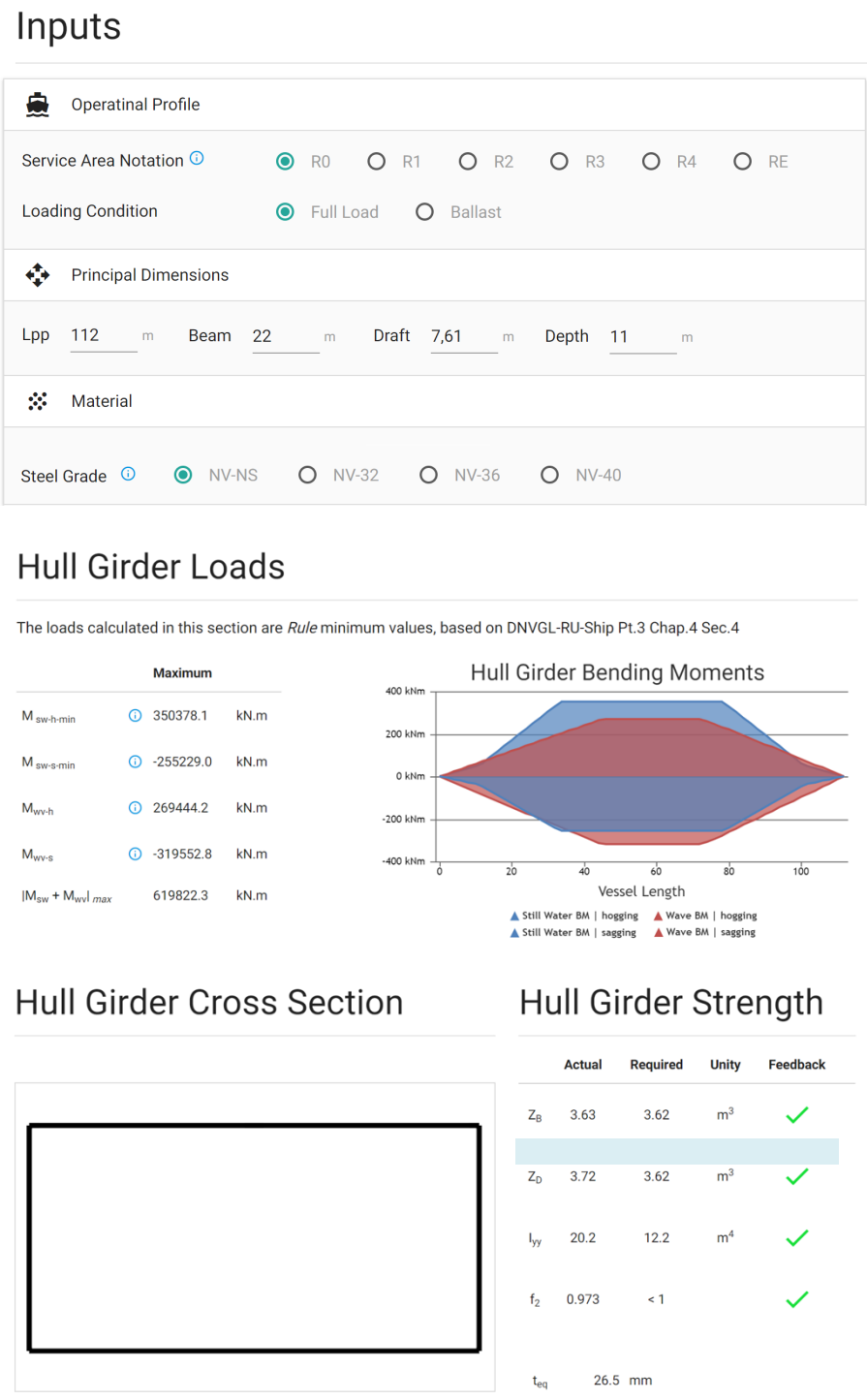


Fig. 15. Simple case 1 – a floating box

For this second case, it was necessary to add more inputs and the number of structural elements escalated from 1 to 65 in this step. The structures are now being individually checked, and at the cross-section visualization (Fig. 16), it is possible to see that the deck line is thicker than the others, representing the higher thickness selected for this structure by the App, in order to satisfy the hull girder strength criteria.

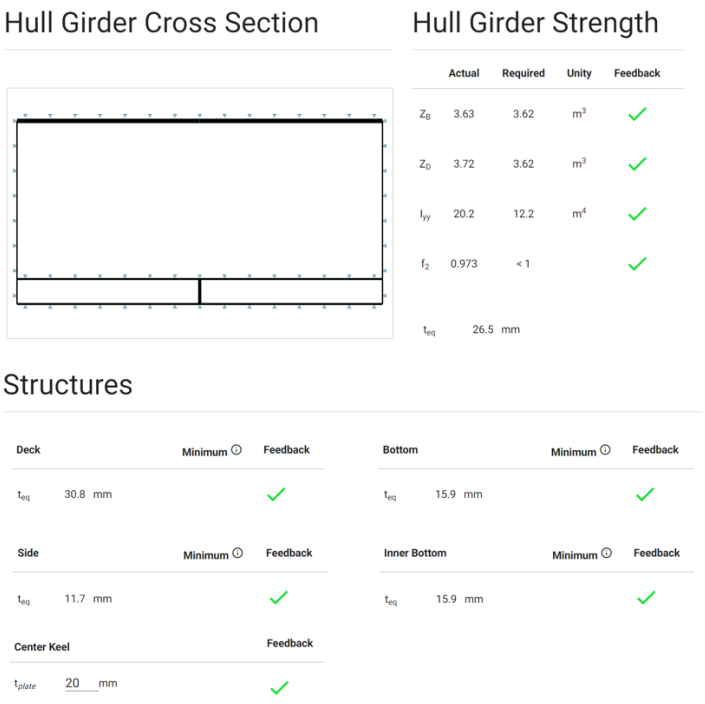


Fig. 16. Simple Case 2: Hull girder

Simple Case 3: As a continuation of Simple Case 2, step 3 aims to verify if it is possible to:

1. Add more structures – decks and longitudinal bulkheads (double side);
2. Make a cross section with rounded surfaces, such as the bilge;
3. Divide the model into regions, each one with a different cross-section and maintaining the stiffening elements position (y , z);
4. Add primary supporting members at the bottom and sides (side girders and stringers);
5. Add secondary supporting members to all structures of the cross-section;

All listed propositions were achieved, and now the cross section has roughly 155 elements between structures, plates and stiffening elements (stiffeners, side girders and stringers). The results are shown in Fig. 17.

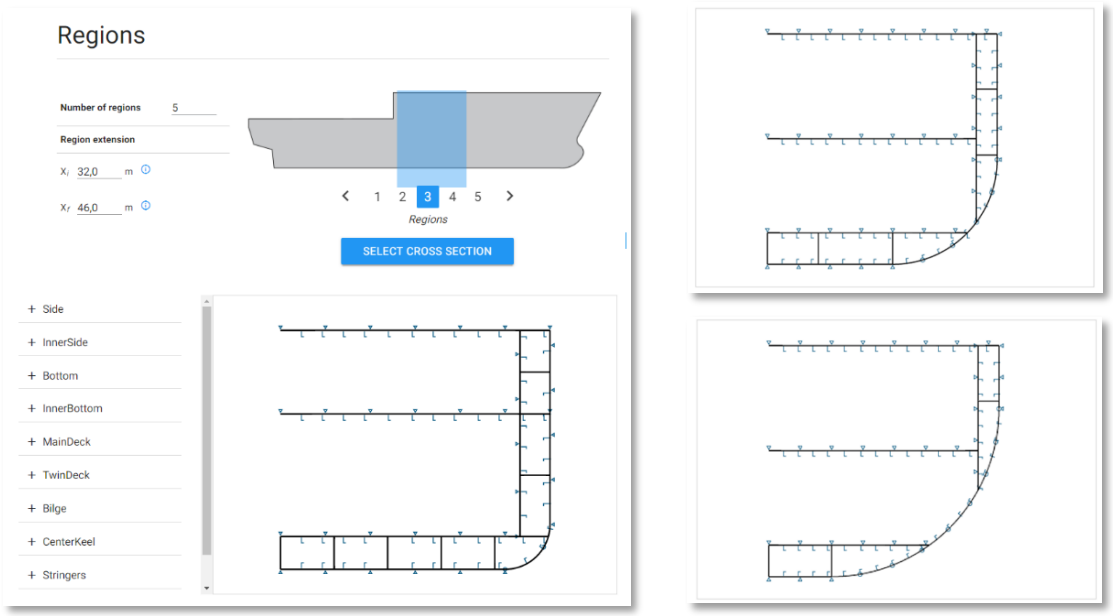


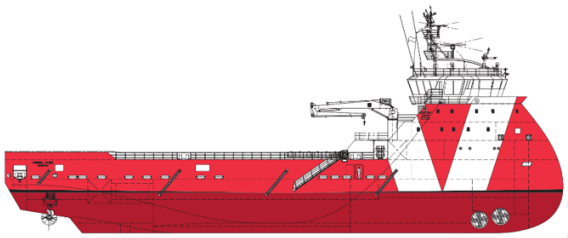
Fig. 17. Simple case 3: Compartments and cross sections with additional cross-section with different bilge radius

5.2. Case Study: PSV Representation

Provided that we have regions with different cross-sections from Simple Case 3, the next step aims to verify if it is possible to:

1. Calculate global requirements for different sections and longitudinal positions;
2. Have more complex cross-section shapes, as typically found in the bow and aft regions;
3. Vary vessels main dimensions and scale the cross section according;
4. When scaling the cross-section, recalculate the requirements for the whole vessel (locally and globally), and provide a feedback indicating if the cross-section meets the new criteria;
5. Extract additional information from the scantlings for each region such as weight estimates, steel cost, volume and welding length.

The five items listed above were proved possible, and their application for a PSV, based on PX121 (Fig. 18) is demonstrated in the following.



Real Data					
L	83	*m*	T_{design}	6	*m*
Lpp	79	*m*	D	8	*m*
B	18	*m*	Deck loading	10	*t/m²*

Assumed Data					
C_B	0.68		Material at deck	NV-32	
Service area notation	R0		Material at Bottom	NV-32	
Tank top height	1.1	*m*	Material elsewhere	NV-NS	
Double side dist. from CL	7.8	*m*	Tween Deck height	5.2	*m*
s_{stiff}	700	*mm*	Load on twin deck	8	*t/m²*
S_{span}	2800	*mm*			

Fig. 18. PX121 (Ulstein, 2016), used as case study

The App initiates itself with two default design load scenarios defined: static and static + dynamic, and arbitrary vessel data. Thus, the scantlings design process started by editing the vessel data, inputting the values previously specified (Fig 19.).

When the vessel data is modified, the App generates a visual representation of the hull girder model according to the new dimensions, and automatically recalculates everything that is influenced by these parameters. The global loads (represented by bending moments) are then recalculated according to DNVGL minimum specification, as well as the bending strength via the hull girder section modulus, which is later set as 'required section modulus' and becomes the criteria for approving or not the global strength.

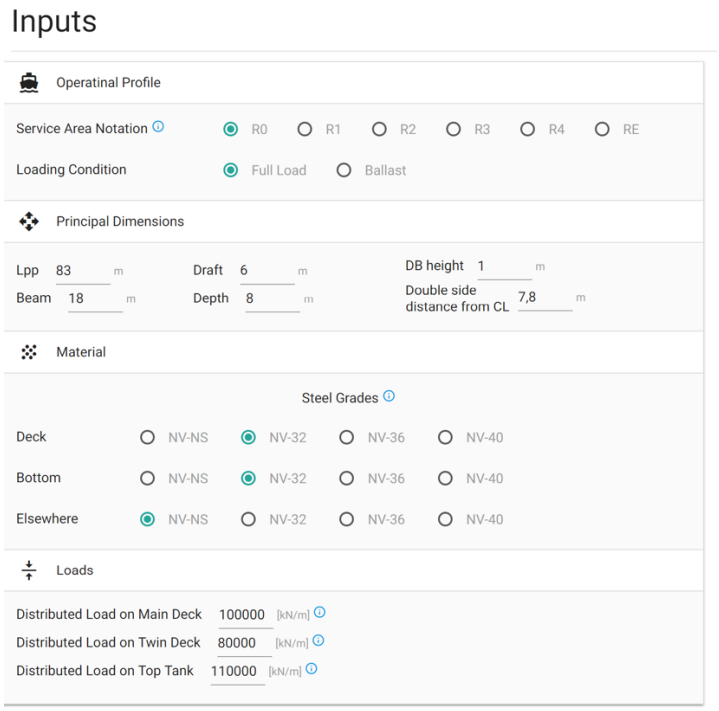


Fig. 19. PSV inputs

Still at the hull girder model, when the spacing between stiffeners is specified, the App again recalculates the minimum plate thickness allowed, using local loads (sea pressure for bottom and sides, and the inputs for the decks and double bottom). The same is done for stiffeners, where the minimum section modulus for each structure is recalculated, using local loads and the permissible stress at the given structure, considering its material and longitudinal position (Fig. 20).

Once the minimum section modulus is established, the App accesses the stiffener profiles' database, selects the one with lowest section modulus above the minimum required, and assigns the profile to the element (object). Edition on the stiffeners' profiles and plate thicknesses are allowed, but not necessary. It was decided to continue the design with the automatically chosen values. Based on this scantlings, the hull girder section modulus is calculated, for the bottom and the deck, and both checked against the required (previously calculated).

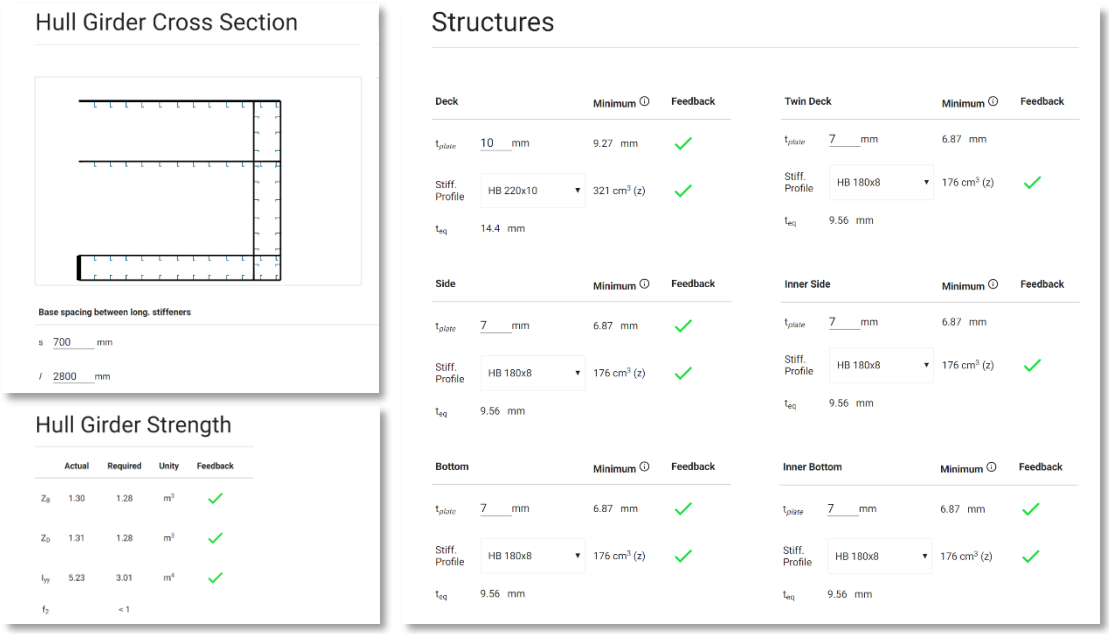


Figure 20 – Screenshot with partial feedback from

The hull girder model is left on standby for a moment, and the regions began to be calculated. In this case, five cross sections were created to represent the vessel, which was (consequently) divided into five regions. Subsequently, a window would pop up with the options for the cross sections (Fig. 21). It is possible to notice that each cross-section has a distinct set of parameters which define them. These are enabled or disable depending on their relevance for the given section being displayed.

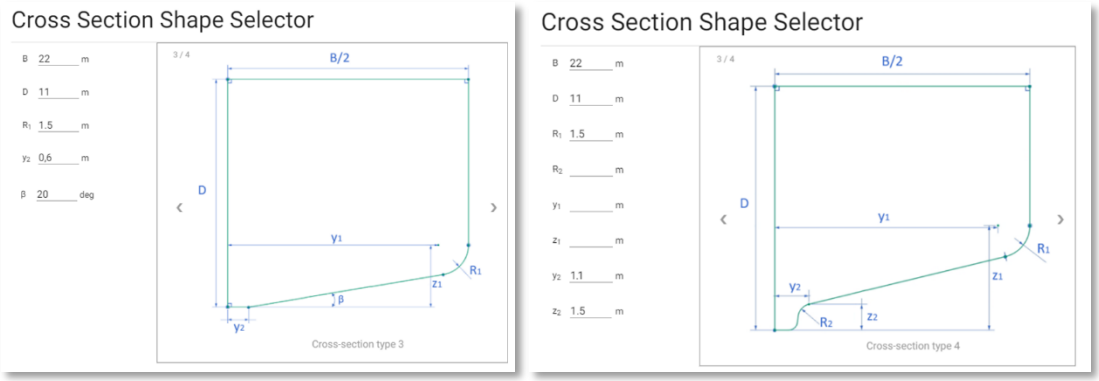


Fig. 21. Two examples from the parametric cross-sections library, adapted for the PSV

Once the cross-sections are selected, the App automatically *fill it up* with structural elements. When an element is created, it is done through a *constructor*. The parent object calculates the position of the next element to be created and which element type it should be (PSM or SSM), then invokes the constructor, passing this information. For example, for creating elements at the deck structure, the cross-section (parent object) takes the spacing between stiffeners defined (by the user or not) for the deck, the

first point of the deck ((y , z) coordinates), and the orientation of the deck (horizontal). Then, it starts counting the intervals (spacing) in the horizontal direction and creating elements (invoking the constructor) at multiples of the spacing, setting every fifth element as PSM and the others as SSM, until it reaches the last point of the deck.

The plates, primary and secondary supporting members are created in the regions' section in a similar way as they were in the hull girder model, according to local requirements (and following the same limitations). Region 3 is created before the others, because since it is the middle one, the App recognizes it as the main section. This is relevant because the position of all other elements in the neighbor regions (within ± 20% of the vessel length from midship – in this case regions 2 and 4) should follow the positions of the elements from the main region (region 3), to ensure longitudinal continuity.

The cross-sections among the regions generated by the App are shown in Fig. 22 (region 3) and 23 (regions 1,4 and 5), with the main properties for each region, such as inertia, area, volume, weight, welding length and steel cost. Aside of the cross-section visualization there is a log of elements that compose the structure, and its properties, that can be edited both in groups or individually.

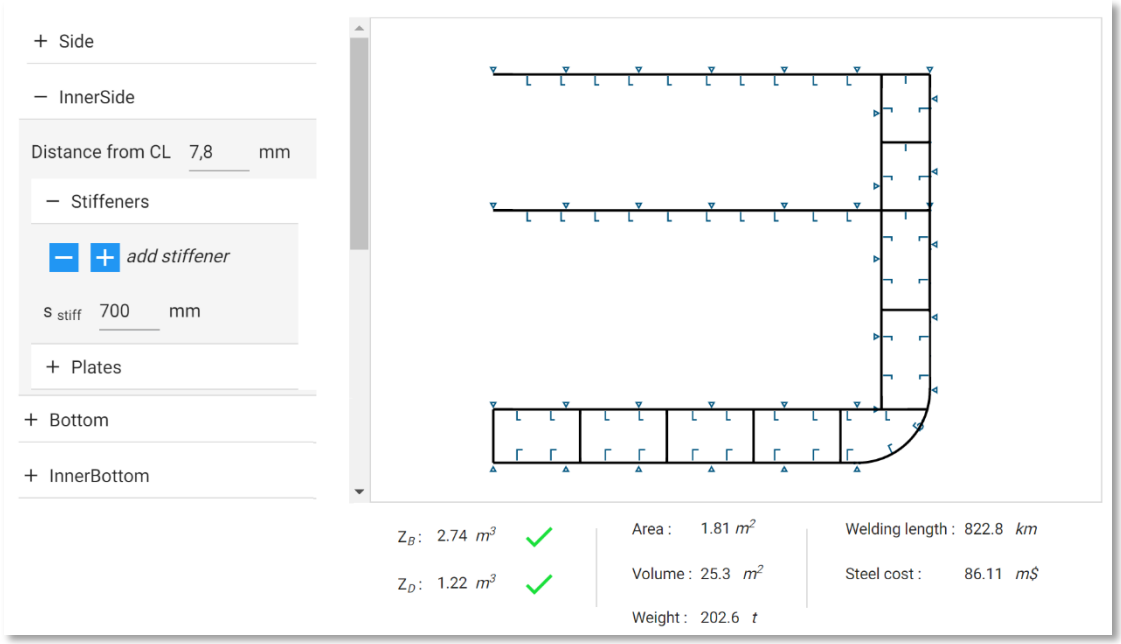


Fig. 22. Region 3 – midship section

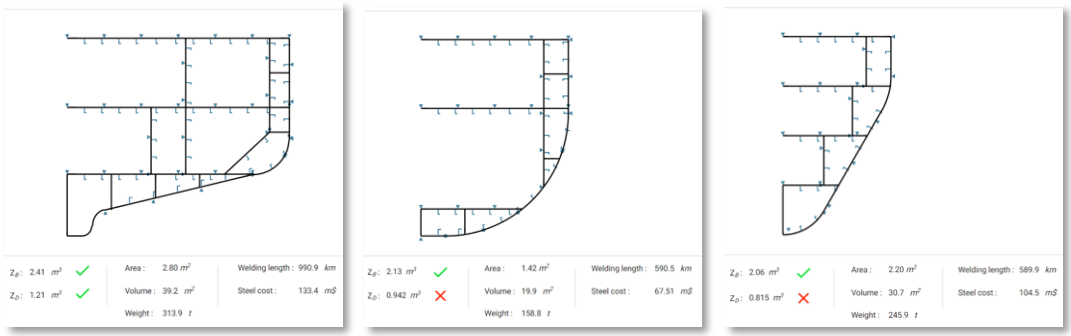


Fig. 23. Regions 1, 4 and 5

After the regions have been fully created, the plate thicknesses and element's profiles that belong to a structure (e.g., deck, side, bulkhead), that is also represented in the hull girder mode, are then compared with each other, and the greater value among them is adopted for proceeding with the design. Again, the global strength check is represented by the section modulus requirement. This time, the gross section modulus is calculated in function of the vessels' length. As both the moments and the permissible stress vary with the vessel length, the gross section modulus also does; resulting in lower requirements towards

the ends than at midship. Then, the highest gross section modulus within a region's range is picked and compared with the actual section modulus for the bottom and the deck of the region's cross-section.

At this stage, the automation for finding a design solution if the global requirements are not met is intentionally limited. Since there are too many elements and too many ways of achieving global strength, the user is freer here to modify the design without triggering an action from the App than he/she is when modifying the hull girder-related parameters. If this last check is ok, the design is done and ready to be exported (available in JSON format) and can be done by hitting the 'export design' button. The vessel data will be downloaded with the *ship.json* name and it can be used in other open tools such as *Vessel.JS*, (Gaspar 2018) or imported back for further edition in the proposed App.

6. A call for collaborative development - extending the knowledge base and creating an open community

The methodology applied for generating the KBS for structural design here presented was based on traditional methods presented in textbooks, such as Ship Structural Design (Fricke, 2016). The goal or our work is not to propose a radical change on the proof-based methods, neither discourage the use of reliable commercial tools, but to plant a seed on open and collaborative methods to early stages structural design, saving time on repetitive tasks and enhancing the design process by correlating it with other design aspects influenced by the scantling, empowering the designer with knowledgeable control over design modification and its effects, along with real-time feedback.

As this work proposes a tool for generating scantlings, structural design principles could not be left aside, and a simple prescriptive approach was chosen for representing a few of them. However, these principles are not the subject of study, neither any particular optimization approach; the focus at this initial step was devoted to demonstrating that, whatever method the user may choose, this collaborative platform has the potential to accommodate it, and enhance the process by combining efforts on automation, usability and the use of implicit design information, while checking for compliance. Moreover, given the openness of the architecture, empirical and theoretical rules can be combined more people is willing to developed upon the current version of the code.

There are a few commercial tools in the market that include a knowledge base and guides the scantlings design upon this knowledge, but this KBS is mostly proprietary. They incorporate a massive amount of knowledge about, for instance, criteria to be applied, loads scenarios to be analyzed, load cases to be considered, formulas for finding the limit states for each element, a database of structural elements and its geometrical properties, among others. However, none of them provide instant feedback about design changes and allow the user to assess the design with respect to weight, cost, painting area, welding demand, among other properties that could be obtained from the scantlings, not even comparatively.

Finally, each shipyard has a particular set of attributes (e.g., construction practices, facilities, suppliers) that influences the metrics of the hull construction. If an open source system is used, there is a possibility to custom these metrics specifically for each shipyard into the system. Additionally, absolutely all information about the design can be accessed, and not only the information contained in the user interface or the report. This raises the possibility to add features to the system that processes any data valuable to a specific user.

We close this paper with a call for our colleagues and students to consider developing future structural engineering design and analysis in open and collaborative platforms. The concept here presented is a working in process, and much of the library structure and methods can be improved in the years to come, a trend observed in other disciplines, such as seakeeping and fuel consumption simulation (Chaves and Gaspar, 2016; Fonseca *et al.,* 2018).

Increasing the knowledge base can be achieved by implementing more rules and expanding the databases. For most of the cases, adding more rules is not more complicated than inserting one more iteration loop (e.g., *for loop*) and/or a condition operator (e.g., *if else*). It could also proceed towards accuracy, by improving the existing rules and substituting the simplified formulas by more complete versions of them. It can address classification rules, and in this case, the accuracy would be obtained by reproducing the exact formulas and procedures as established in the rules, or it can address the additional information taken from the scantlings model, such as weight and cost estimation. This second operation has the potential to be customized for each shipyard, by including specific production data.

The App could also be expanded with respect to the level of details contained in the design, by including, for example, rules for establishing the weld type/technique applied for each element, or where to have cut-outs and their shapes. Additionally, it could handle elements transitions and collision, not

allowing abrupt changes of plates thickness or elements' profiles, or SSM too close or colliding with PSM, for instance.

It seems natural to include an extension for documentation, providing means to generate reports in PDF format according to a given template, and finally, it could also progress towards integration with other KBS by exporting the structural design data in a readable format by other software programs, proprietary or not. Another option is to integrate with the Vessel.js project (Gaspar, 2018), which is being developed on the same basis – open source web-based applications for specific ship design tasks. It would allow a 3D visualization of the structure, in order to assist the designer to assess spaces and perceive collision issues more easily, without the need to model the structure in another software with 3D capability just for this purpose.

With respect to the automation part of it, it can grow towards more sophisticated methods, such as machine learning and artificial intelligence. The post-processing could incorporate a sensitivity analysis study. Finally, the option of letting the user to have control over the priority of actions to be taken by the App, when suggesting a design solution, could be enabled. In this way, the computer would 'imitate' the designer's procedures as if he/she would be doing it, but thousand times faster. The assumption is that, if there are enough rules in the knowledge base, the design could be fully automated.

Acknowledgements

This article was partly supported by Ship Design and Operations Lab at the Department of Ocean Operations and Civil Engineering (NTNU in Ålesund), SINTEF Ålesund and by the EDIS Project, in cooperation with Ulstein International AS.

References

Bergström, M., Erikstad, S. O., & Ehlers, S. (2016). Assessment of the applicability of goal-and risk-based design on Arctic sea transport systems. *Ocean Engineering*, *128*, 183-198.

Caprace, J. D., & Rigo, P. (2012). Towards a short time "feature-based costing" for ship design. *Journal of marine science and technology*, *17*(2), 216-230.

Chaves, O. & Gaspar, H. M., 2016. A Web Based Real-Time 3D Simulator for Ship Design Virtual Prototype and Motion Prediction. 15th COMPIT, Lecce.

Cui, J.-j. & Wang, D.-y., 2013. Application of knowledge-based engineering in ship structural design and optimization. Ocean Engineering, pp. 124-139.

Erikstad, S., 2009. Modularisation in Shipbuilding and Modular Production. Trondheim, s.n.

Fonseca, I. A., Gaspar, H. M., Ryan, C. F. & Thomas, G. A., 2018. An Open and Collaborative Object-Oriented Taxonomy for Simulation of Marine Operations. 17th COMPIT, Pavone.

Fricke, W., 2016. Ship Structural Design I-III. Hamburg: Hamburg University of Science (TUHH).

Gaspar, H. M., 2017. JavaScript Applied to Maritime Design and Engineering. 16th COMPIT, Cardiff.

Gaspar, H. M., 2018. Vessel.JS an open and collaborative ship design object-oriented library. IMDC 2018, Helsinki.

Park, Y., Jang, B. S., & Du Kim, J. (2015). Hull-form optimization of semi-submersible FPU considering seakeeping capability and structural weight. *Ocean engineering*, *104*, 714-724.

Timoshenko, S. & Woinowsky-Krieger, S., 1940. Theory of Plates and Shells. New York: McGraw-Hill Book Company.

Ulstein Group, PX121 Folder (www.ulstein.com), accessed in Jun 2018.

Verhagen, W. J., Bermell-Garcia, P., van Dijk, R. E. & Curran, R., 2012. A critical review of Knowledge-Based Engineering: An identification of research challenges. Advanced Engineering Informatics, Volume 26, p. 5–15.

Yang, H. Z., Chen, J. F., Ma, N. & Wang, D. Y., 2012. Implementation of knowledge-based engineering methodology in ship structural design. *Computer-Aided Design 44*, pp. 196-202.