Felipe Ferrari de Oliveira

An open web platform aimed at ship design, simulation and digital twin

Master's thesis in Naval Architecture
Supervisor: Henrique Murilo Gaspar
Co-supervisor: Ícaro Aragão Fonseca
June 2021

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Ocean Operations and Civil Engineering

NTNU
Norwegian University of
Science and Technology

Felipe Ferrari de Oliveira

An open web platform aimed at ship design, simulation and digital twin

Master's thesis in Naval Architecture
Supervisor: Henrique Murilo Gaspar
Co-supervisor: Ícaro Aragão Fonseca
June 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Ocean Operations and Civil Engineering

Contents

List of Tables

List of Figures

Preface

This thesis marks the end of my Master of Science degree in Naval Architecture Program in the depart Department of Ocean Operations and Civil Engineering (IHB) at the Norwegian University of Science and Technology (NTNU). The final project deals with several concepts taught during the master program and experiences acquire during the participation in several projects for the Ship Design and Operation Lab (ShipLab) at NTNU.

The workload demanded to accomplish the project was 30 ECTS and was developed during the spring semester of 2021.

Ålesund, 25th June 2021

[illegible]

Felipe Ferrari de Oliveira

Acknowledgements

It would be injustice to not acknowledge some important actors for my professional development as well as this thesis conclusion. Without their supports I most probably would no be able to achieve the quality degree this thesis was able to delivery.

First and foremost, I would like to thank the open source community as a whole by providing the libraries and tutorials that were the foundations of this project. It is amazing to see the quality of open source projects made by humans sharing content and helping each other, even in areas related to the most advanced technologies of this time. It would be impossible for me to thank each of the projects that contributed for this thesis, but I would like to highlight the importance of the Vessel.js and their developer team in ShipLab as the most crucial for this thesis accomplishment.

I would like to thank in particular my advisor, Henrique M. Gaspar, for all the investment in myself and its initiative in turning the open source philosophy closer to the students. Icaro Fonsceca also deserves special greeting for being one of the most important persons to create accessible documentations for Vessel.js, helping me personally with suggestions for the improvement of this thesis. I believe both are creating a legacy by promoting open source projects inside the ShipLab and deserve this recognition.

Finally I would like to thank the NTNU University, and indirectly the Norwegian society, for funding my education, providing high quality three dimensional models and vast sensor data for my thesis purpose. Without this support I would not be financially able to conclude master degree to become a high valuable professional.

Summary

This project identifies the web platforms as one of the main tools for data management, gaining special importance in the recent explosion of information that were led by the improvement of data handle in respect to the communication, acquiring and storage factors. Even those some solutions started to rise, the web platforms available in the industrial sector, including the maritime, are still scarcity. In that sense, this project proposes one open source solution for ship simulation and analysis that could be evaluated as alternative.

The proposed web platform architecture uses a set of open source tools that together provide an end-to-end framework for building dynamic web applications. This approach of architecture enhance the modularity aspect of the platform, turning easy the adaptation, innovation and expansion of the internal modules.

One concern that was handled by the web platform was the management of complex data from ship design and operation. The idea is to use analysis and simulations in those two different phases to validate or improve the models. The web platform will adopt a set of modules to present the information contextualized for each type of the project, adapting itself to show only the applicable information depending on the phase evaluated (design or operation).

On the project phase, the platform runs inside the Preliminary Ship Design (PSD) theory that runs more into a holistic instead of sequential methodology. In the design

phase the ship can be displayed in the initial and detailing mode. For the initial design the display shows a building block perspective, assessing the main components and its spaces with an initial definition for the location. In the detailing design a more definitive form is presented, with its components divided according to the Vessel Information System (VIS) taxonomy with a tree filter that helps to identify the main components. For the operation phase, the platform permits to stream the location of the ship through the time, using a three dimensional visualization, the value can be contrasted with the Maneuvering modeling developed during this project.

An essential aspect for an open platform is the ability to allow collaboration between several partners. This report shows three practical ways related to each tier in the platform architecture for collaboration, showing its features and presenting study cases the collaboration can be used to enhance ship simulation and analysis. This report also uses the creation of the Maneuvering model as background to show how to expand the platform through the back end collaboration.

Finally, the report evaluate critically the project, raising important aspects regarding the limitations of the platform, suggesting improvements and future tasks that will improve the platform user experience.

Chapter 1

Introduction

The recent costs reduction for data storage, computing processors and sensors are leading to an unprecedented explosion of information which has shaped the industry in different segments such as transport and pharmacy. The easily acquired data can now be stored, shared and calculated to validate and support decisions in a faster and cheaper way.

According to a study by McKinsey&Company (2016) the location-based services and retail sectors have been responsible for the greatest progress into capturing the potential value that data and analytics can deliver, where the thin margins facing retailers (especially in the grocery sector) and pressure from competitors create a strong incentive to evolve. On the other hand, manufacturing, the public sector, and health care have captured less than a third of the opportunity value that data and analytics domain presented between the years of 2011 and 2016, despite the increased potential of value creation during this period.

One of the approaches retailers used for this achievement was the incorporation of social media web platforms to analyze the customers' behavior in order to contextualize marketing campaigns, and apply statistics experiments to validate the models. In the maritime sector a few good solutions involving web platforms application start to rise, however, like the industrial sector as a whole there is still a considerable unexplored horizon to use data more effectively with the inclusion of digital platforms in order to minimize costs in the production and operation. Some solutions reached through digital platform applications involve centralizing the information and simulations of an engineering system into a single architecture shareable between several stakeholders.

1.1 Background tand Motivation

The willingness for creating a digital hub for engineering system representation that allows different simulations as well as contextualized data visualizations is not a novel concept. Its origins can be tracked to Apollo's programs in the 60's where physical replicas of certain systems were built to mirror the operation of a space ship, leading later to the motivation for developing a real time digital system representation. This concept is called today as digital twin, being defined by Boschert and Rosen (2016) as a comprehensive physical and functional description of a component, product or system, which includes more or less all information which could be useful in all — the current and subsequent — lifecycle phases. Digital twin should also support mathematical models for the system simulation, helping operation and planning of improvements for the devices it represents.

The maritime industry which comprehends floating structures such as ships and offshore wind turbines, would gain access to multiple advantages by incorporating web platforms for twin ship operations. For instance, on maritime the analysed objects are often treated as complex systems, involving several components which together must accomplish a specific function. All the systems in a ship behave in an interrelated behavior, so there is no single component which can be exchanged without affecting in different scales other systems. With that chain reaction behavior, a simulation that works by a holistic approach is better to describe a real operation and verify how the studied scenarios may affect the systems. In face of that high complexity degree and spectrum of information, a platform can work as a hub, organizing and contextualizing the data according to the analysed subsystem or scenario, easing user awareness and enhancing the information exchange. The most ideal platform should be also able to concisely contextualize the acquired data depending on the life-cycle studied, allowing the exchange of information between phases according to the concept defined as Digital Thread.

Another important aspect is that the simulations are concerned by different stakeholders, in some cases even by different institutions. This means that in a simulation chain lead by distinct teammates, the information gathered by one designer must be processed and treated before passing to the other stakeholder, who will parse it again to finally obtain the desired result. This approach is the convention in most companies nowadays because of differences between software types and data formats. Even if this process may seem simple in a small scale, it is laborious and becomes susceptible to errors when the amount of variables and simulations required increase, such as in a simulation with a complex of a twin ship operation. By applying a common data structure through a platform, all simulation results can be easily exchanged, i.e., when a simulation is executed, its results are stored in a data base where all the other stakeholders will have access to parse their respective information. In that way the platform can mitigate the risks of non unified model versions and simulation scenarios by running the logical operations in a web environment with the same universal resource locator (URL) for all possible users.

In addition, it must be taken into consideration that the maritime simulations require input information that is not easily obtained. One typical example is the hydrodynamic simulations, in which the non-linearity of the Navier-Stokes equations requires numerical methods that in current personal computer's processing capacity can take a range of hours or even days to be calculated. Once again a platform shows its advantage because it can be linked to a common external database that stores the information independently, granting access to third parties to calculate those information using specialized tools for complex numerical methods meanwhile the platform itself can stream the results to personal computers through a web interface. In that sense the platform reduces the risk of the information being siloed between few agents responsible for the decisions and calculation, giving a common established place to access the stored data.

One practical example that motivated this project was the necessity ShipLab has to unite into a single architecture several analysis and simulations application accessible in 'ShipLab Website' (2021), providing a more concise way for a long term ship evaluation, constituting a data source of ships version that can easily be suited to its several analysis and simulation types. ShipLab is the institution responsible for the development of several projects to enhance the digitization in ship industry and a close partner of this project. Vessel.js library, one of the functionalities created by the ShipLab team for conceptual ship design with an object-oriented paradigm is one of the most crucial tools used in this assignment, and will be used as a model for the platform improvement.

1.2 Thesis Scope

The employment of web platforms for digital twin operation is a holistic practice, that assimilate knowledge from different fields such as engineering, information technology and business intelligence. This project will aim to construct a platform using primarily the engineering perspective via subjects of digital twin operations, data management and maritime simulations, as represented in Figure 1.1.

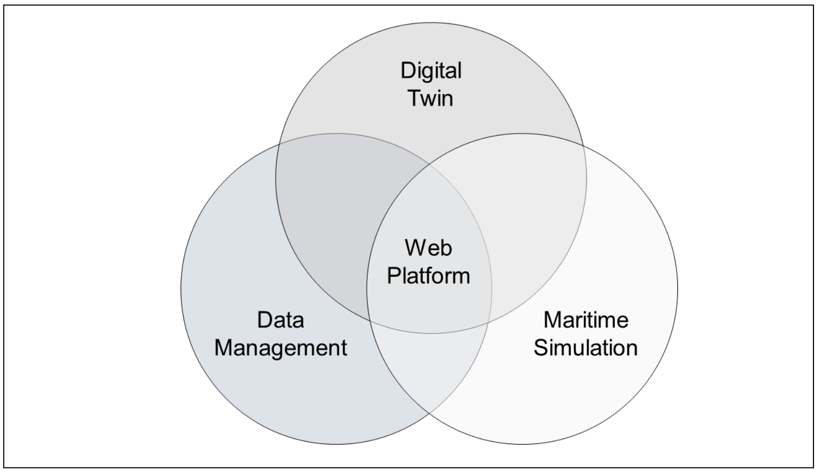


Figure 1.1: Venn diagram with the knowledge areas applied in this project.

In the intersection of those three areas is located the web platform as a tool to handle those activities. The ultimately goal for a web platform for twin ship operation is to attain the digital thread of the system which according to Taber et al. (2019) is "a single source of reliable data creating consistency, collaboration, and alignment across functions by real-time data synchronization of related upstream and downstream derivative information in different life cycle phases". , organizing and making use of the system data from conceptual design to discard life cycle phases.

The present project proposes to apply a web platform as a management source for only ship operation and design, but in future work the platform could be extended to fulfill the requirements for a digital thread of a ship project. The platform will include those two phase as a form to bridge the gap of information between design offices and operators, a common obstacle in maritime industry as exemplified by I. A. Fonseca et al. (2021) in which the ship owner often operates independently from the design office after the ship is tested according to a standard row of criteria in a sea trial. Figure 1.2 depicts the conceivable data loop between design and operation phases, which will be the focus on this project.

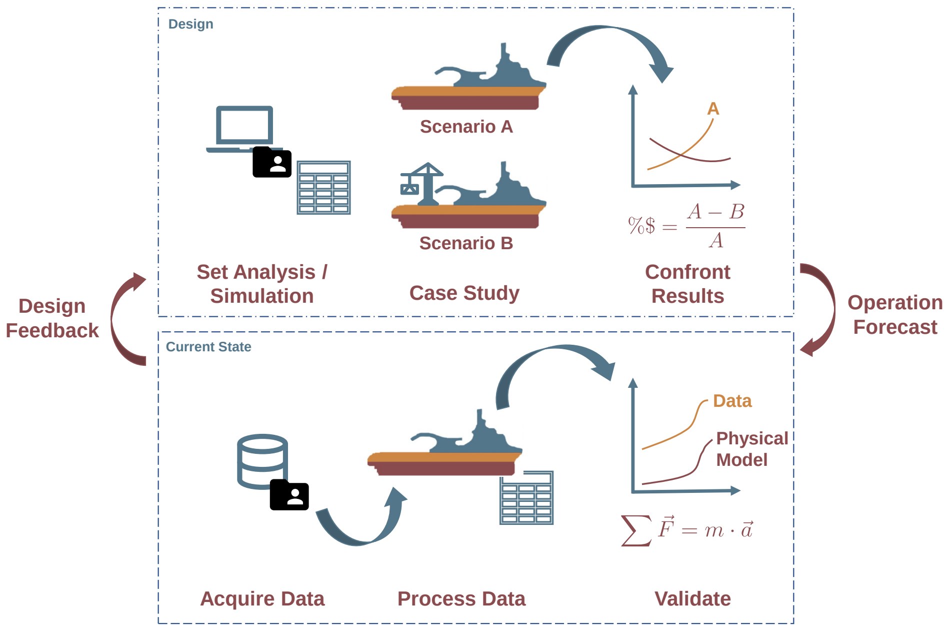


Figure 1.2: Conceivable loop for data use between the ship design and operation.

1.3 Objectives

The objective of the research is to investigate how ship analysis and simulation data can be handled effectively by employing a web-based platform approach. The focus is to link the data for the maritime system, identifying relevant roles the platform should fulfill during design and later during operation. The platform will rely heavily on analysis and simulations as a form to evaluate and validate the systems and support decisions.

In order to have a versatile data structure it is imperative that all the simulations must be designed interchangeably in a modular way, encapsulating the information and eliminating the repetitive logical operation required in frequent simulations. The common data structure must be accessible to several stakeholders and be compatible with different applications inside and outside the platform, therefore a great part of this thesis will focus on the concepts of taxonomy and how the data must be standardized to reach a high acceptance between several partners.

In summary the thesis has the mission to raise the knowledge that gives the support for web platform for simulation and analysis as part of digital thread. The objectives of this thesis are listed bellow:

- Establish the structure of the web platform in a holistic perspective by looking for the set of components that constitutes the web platform (users, views and database integration) with focus to enhance modularization and expansion;
- Propose a template for maritime objects description and data sharing to organize and give grounds for scalability, incorporating the established models from industry class;
- Apply analysis and simulations for ship evaluation. The available operational data will be used together with the simulation models as a support for a potential digital twin;
- Expose the functionalities from the proposed platform, evaluating critically its

applicability in the digital thread context;

- Discuss the improvements and possibilities in engineering field that can increase the potential of the platform applications and what are the currently main technical concerns that limit it.

1.4 Thesis Structure

The thesis is structured under seven chapters, including the present introduction which argues about thesis scope. The following chapters are presented bellow with a brief explanation about its content:

- **Chapter 2:** A broad literature review about the application of digital twins, web based simulations and data management applied to maritime systems;
- **Chapter 3:** Introduces the platform, its components, and structures;
- **Chapter 4:** Present practical terms in the application of the components;
- **Chapter 5:** Applies the concepts discussed under the previews chapters in case study examples;
- **Chapter 6:** Analyse through KPIs the relevance of the thesis and show some limitations and improvements that could be made;
- **Chapter 7:** Summarises all the contents and suggest possible future projects based on the knowledge raised in this report.

Chapter 2

Literature Review

The literature review aims to draw a landscape in the best practices that support a platform application as a data hub tool for an ideal digital thread context. Based on the principles presented in the last chapter and condensed in the Figure 1.1, the literature field can be separated in three categories that encourage the digital thread: Digital twins, Simulation in maritime industry, and data management applied to maritime systems.

2.1 Digital Twins

2.1.1 Defining a Digital Twin

To quote from El Saddik (2018) "digital twins are being redefined as digital replications of living as well as nonliving entities that enable data to be seamlessly transmitted between the physical and virtual worlds". The virtual world in this concept is a representation, usually a CAD model of a physical entity such as a device, system or even a human beam together with the data acquired through sensors reproduced interactively via tables and graphics or in the 3D model layers. Figure 2.1 gives an example of the physical and virtual world in a maritime application for a supply boat in heavy sea.

Even those digital twin is not a novel technology, at least in its concept and small

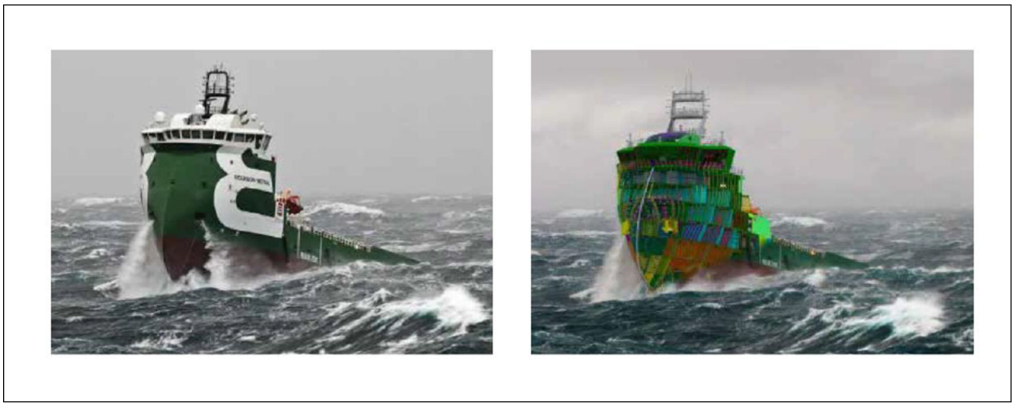


Figure 2.1: Example of digital twin. The left hand side shows a real ship in a storm environment, the right hand picture shows the virtual representation of the structural elements subjected to external environmental forces. The color scales in the virtual portrayal can classify the objects according to its physical division or simulation data results (i.e. forces, pressures).Source: `https://irsweb.it/`, accessed in 2020-12-11.

applications such as explained in Section 1.1, its importance for the industry was boosted up thanks to increasing processing capacity, expansion of sensors applications and network connectivity improvements - like 5G for example - that allows a almost simultaneous connection between the physical and virtual entities. With the facilitation of digital twin application comes into play the dialogue about what are the type of data which worth representation and how to contextualize those data in an accessible environment in a meaningful way for the decision makers.

Boschert and Rosen (2016) correctly advocate for the usage of digital twin focusing on the improvement of the systems during their operation, bridging the gap between the development and usage phases, with different granularity models and needs together with a general definition of its principal architecture (structure, content and purposes). The author argues Digital Twin is not what he calls figuratively as "data monster", which includes everything from all lifecycle phases dissociated from its application. This report itself aims to construct the web platform with this concept in mind, ensuring the simulations and features are not away from the purpose of a ship in operation.

The digital twin also has a important role in extending the testing period of the project through the operation as Glaessgen and Stargel (2012) points out. In the traditional analysis the simulations in the design phase are made according to the possible failures faced by observed experience and then incorporated in the models. In real case, however, different environments or different degrees of environments can compromise the integrity of the system by putting the components in a non planned situation. A digital twin which acquire the data in operation can be used to predict failure or improve current models by assimilating the non imagined situation in the design phase as well.

The effectiveness of digital twins can also be enriched by artificial intelligence analysis in order to make fast and wiser decisions. One example could be the failure prediction models, in that case a fleet can be inspected more frequently through the synchronized data according to each specific component historic information, instead of applying a generic inspection to a class of equipment.

2.1.2 Digital Twins in Maritime

Í. A. Fonseca and Gaspar (2020) indicates several applications of digital twins in maritime industry for different type of fields. On the resistance and hydrodynamic field Coraddu et al. (2019) estimates speed loss caused by marine fouling using a simulation model based on a neural network, demonstrating through this approach superior performance in comparison to the ISO standard for estimating fouling. On the reliability context, Schirmann et al. (2019) present a digital twin for ship motion and estimation of structural fatigue due to wave response according to weather forecast data for a given route. On the field of automation Danielsen-Haces (2018) apply a digital twin to autonomous vessels for condition monitoring and calibration of the propulsion system models based on operational data.

More linked to the combined application of digital thread to digital twin, the luxury constructor of Yatchs Benetau used sensors connected to their products to incorporate real word data to engineering and manufacturing process, allowing them to customize their services to each client and improve their engineering products (Landolo, 2019).

2.1.3 Digital Twin as Part of a Digital Thread

The digital thread is a concept that consistently integrate the acquired data through the life-cycle in a single source in order all stakeholders in the process can benefit from the exchange of the information and collaboration.

Even those the concepts of digital twin and digital thread are closed related they are not the same in practice. The digital thread can be interpreted as an update of the digital twin by assimilating information in several phases of the life-cycle, while the digital twin alone is a reliable representation of the physical system (Singh & Willcox, 2018). Figure 2.2 depicts a flowchart pattern of a digital threat application involving an update on the design phase according to other stages information.

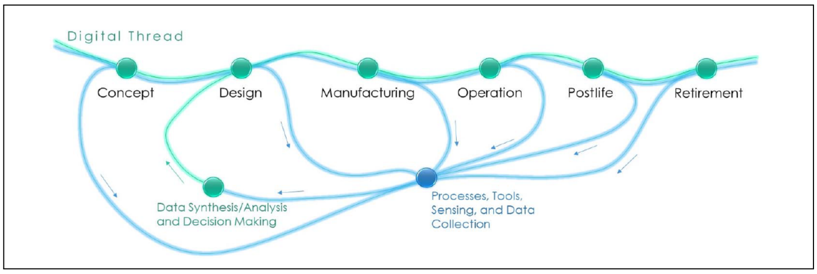

Digital Thread
Concept
Design
Manufacturing
Operation
Postlife
Retirement
Data Synthesis/Analysis and Decision Making
Processes, Tools, Sensing, and Data Collection

Figure 2.2: Illustration of engineering design with digital thread, source Singh and Willcox (2018).

For practical reasons the current platform attend analysis and simulations just in the design and operation phase. However, it is important have in mind the big picture that a improved version of this web platform can offer, by becoming an established trustworthy central source for the information exchange in the digital thread of the product. The upgraded web platform version can be used to contextualize the data and adapt the information structure to a single taxonomy, avoiding friction in the decision making and loss of relevant information in the exchange through different life cycle phases.

2.2 Simulation in Maritime Industry

2.2.1 Simulation in Design Phase and Virtual Prototyping

The simulations in ship design phase are done for predicting the system behavior in different scenarios according to previews situations faced during the operation with the purpose of getting improvements or regulatory approval. The level of details in the simulations during the design phase also permits the user to identify earlier failure and avoid the components over dimension, averting unnecessary costs with material and labour. Vossen et al. (2013) gives an broad overview related to the ship design and the simulation importance in this phase, according to the author a high degree of innovation in maritime sector require a high demanding knowledge in the integration of the several subsystem which is just achievable in a satisfactory level by relying on 3D modeling and simulations.

Morgan and Pawling (2009) analysed a compilation of studies from different simulations during preliminary ship design, ranging from evaluation of design for production proposals to simulation of fire, smoke and heat dissipation. His article advocates for simulation usage in early design phases to enhance Preliminary Ship Design (PSD) architectural approach. According to the Andrews and Dicks (1997), the PSD reconcile novel ideas with the established constrains better than traditional design approaches by applying a holistic instead of sequential methodology.

One of the approaches used in PSD is the Building Block, which consists of early design segmentation where performance will be investigated, changing the ship configuration until the design state solution is satisfactory (Andrews & Dicks, 1997). The Figure 2.3 shows a flowchart for PSD using the Bulding Block approach with a surface Vessel as study case, the drawing was modified to highlight through a red polygon the region of activities in which the simulations and analysis can help to validate the geometry, this region will be used later in the Chapter 4 to place where the proposed platform is framed in design phase. The manoeuvrability model, presented inside the red polygon, was chosen to be developed in this thesis as a study case for the platform collaborative development via the physical modules improvements.

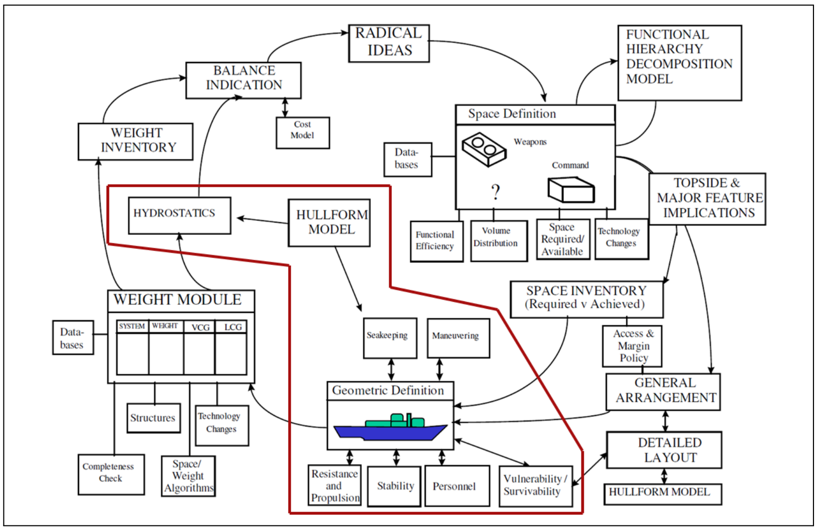


Figure 2.3: Building Block Design Methodology applied to surface Ships. The red polygon highlights the location where the simulations and analysis validate the geometry concept. Source: Andrews and Dicks (1997), modified.

In the field of open-source application such as the one cultivated in the present project as well, Chaves (2018) creates a web based application for the early structural design to automatize the hull scantlings calculation, using the open-source concept to ease sharing and collaboration. Her work reveals potential gains in automation of mechanical activities and elimination of potential errors by non proper information inputting. To achieve this goal, the author uses the Knowledge-based engineer principle to ground computational decision for lead-time reduction.

As a facilitator to the design process comes the virtual prototype (VP) concept of the models which consist, according to Wang (2003) definition, of a simulation where the aspects of the product can be presented, analysed and tested. The objective of VP is to validate the systems and processes prior the manufacturing and mock-up of the components, eliminating the employment of extra resources. He et al. (2015) suggest a resource management pattern for working with the collaborative virtual prototype design, separating it into three components: Entity Models (EM), State Model (SM) and Process Mode (PM).

The EM represents the visualization models portrayed by 2D or 3D computer-aided design (CAD) models with the purpose to define its spatial constrains, being a foundation for the other two components. SM represents the physical result information applied to an EM constrain in a certain condition, to give a practical naval example take the forces in a mooring system according to the position of the mooring points on the ship and seabed, these forces are dependent of the factors of the operation and vary to each possible configuration, therefore being classified as a state variables. Finally the PM is a expression of the continual adjacent simulation states models, to remain in the same mooring example given before, a process model for this case could represent a six degree of freedom (6DOF) simulation on time of a floating structure which will contain a collection of state results (forces for example) through time. The Figure 2.4 exemplifies the division of the virtual prototype model into these three components, showing the exchange function between the elements.

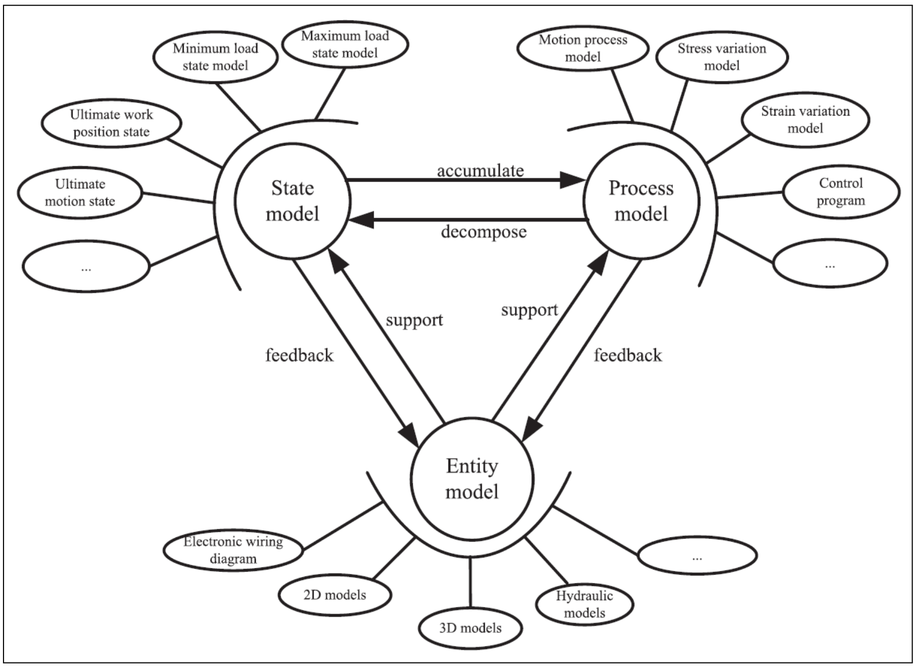


Figure 2.4: Virtual prototyping model example, source: He et al. (2015)

As an important knowledge source for the simulation in the design, the virtual prototype philosophy will be applied into the platform proposed, together with the nomenclature division for entity, states, and processes defined by Wang (2003).

2.2.2 Simulation in the ship operation context

In the operation phase the simulation are applied for human training in critical tasks, to support the decision making in face of economical and environmental circumstances, and to validate mathematical models.

On the decision field the operator can track scenarios using statistical and physical models to propose improvements and operability window. The digitization process reduce the span between the analysis and action by acquiring and processing faster real data measured by sensors. The digitization process also improve the user interface content, allowing the decision makers to have a clearer vision about the studied case. The simulations models in the operation phase have the advantage compared to the design phase in relation to cross validation opportunity since the real data allows to calibrate the model for better estimation of system behavior.

For personal training, the simulation can improve the capacitation of the employees, eliminating risks, improving service quality and granting the engineering team to have practical assessment from the in hand operators, fostering the feedback through different departments of the same company. The high risk involved and the difficult to reproduce certain conditions in controlled environments in maritime operation leaded to a particular development into simulations for training purpose in maritime area. As a role model in infrastructure provision for maritime simulation we can cite the Offshore Simulation Center (OSC) facilities located in Ålesund, Norway. The OSC training center offers rooms, Figure 2.5, that resembles bridges of ships with interactive devices such as navigation screens, localization apparatus, communication radios and control systems. The room adopt a series of projectors applied thought a domed surface to mimic the maritime environment.

Away from a high infrastructure requirement, the Vessel.js website shows a series of exemplifications (`http://vesseljs.org/`, site accessed 2021-05-30), using its library in an operational case for mooring systems together with a documentation that

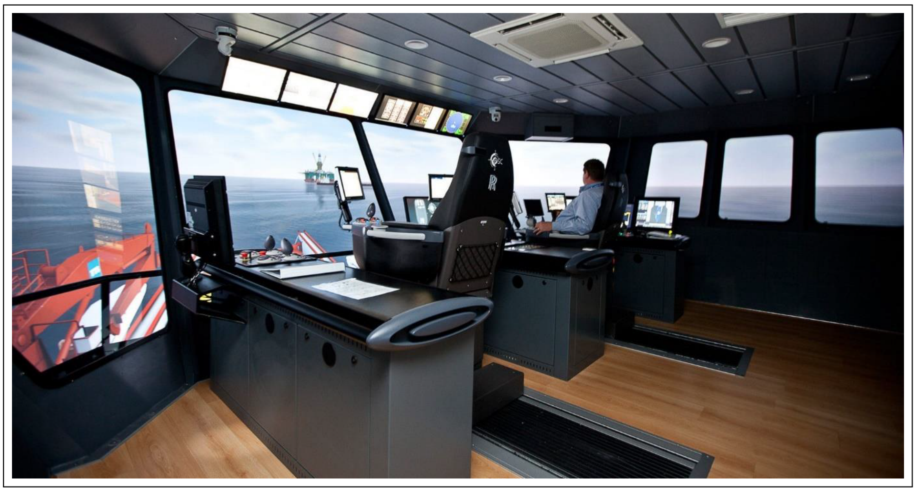


Figure 2.5: OSC bridge room infrastructure. Source: `https://osc.no/`

eases the customization of mooring lines and free body calculations with some hydrodynamic domain restrictions. In a corporation level the SESAN for maritime systems from DNV can simulate operations such as lifting analysis, tanks offloading, crane operation, transportation of floaters, the Figure 2.6 shows a preview of the software simulating a crane lifting operation.

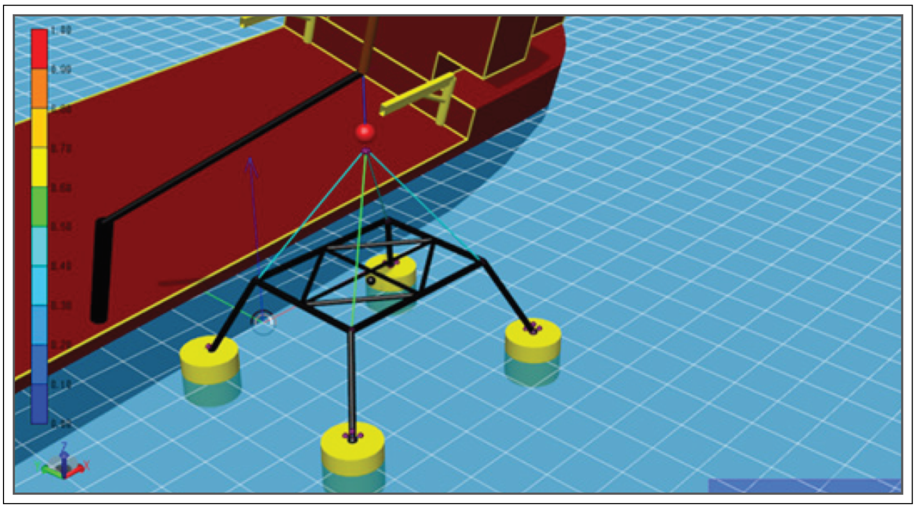


Figure 2.6: Preview of a crane lifting operation using SESAN. Source: `https://www.dnvgl.com/`, site accessed at 07/01/2021

Academically, several ideas were introduced to expand the possibilities horizon in simulation of marine operation. Varela et al. (2015) proposes a simulation architecture for rescues planning operations in damaged ships. Briano and Caballini (2012) shows a simulation for port operations used for training purpose in loading and unloading craning operations, one advantage from the simulation proposed is the capacity to set up the climate environment and impose accidental circumstances, situations that are difficult to control in a real trial. For applications using web based simulation we can cite Vieira et al. (2020) which uses the Three.js library, accessible in 'Three.js website' (2021), to introduces all the phases and components role in liquefied carbon dioxide storage operation in offshore salt caves, the visualization refereed to this work can be seen in Oliveira et al. (2021), and has the purpose to elucidate the process and the variables effects in an educative manner.

2.2.3 The Role of Simulation in Digital Twin

Digital twins performance evaluation are based on operational simulations, however they differ significantly from the previous simulation approaches that do not rely on sensors and quasi real time data for its evaluation.

Erikstad (2017) exemplifies by Figure 2.7 how the increasing in information can modify the simulation approach. On the first level, representing the engineering analysis, there is no time dependent information, this level aims to analyse the performance in general terms supported by the physical laws. On the second layer of simulation the time variable appears, in this case it is possible to assess the performance according to time series dependent variables using statistical principles anticipate load and displacement cases. On the third layer the presence of the sensor allows a quasi real time simulation, with a sensor based to acquire input data for the model, this standard is just achievable with tolerable precision by the digital twin application.

In this ambiance, digital twins are a tool to validate the analysis and simulations, revealing possible flaws in the adoptions and improve the models for better predictions. The digital twin must in this sense be an interactive process, where the information acquired through the sensors must be critically inspected by the analyse and simulation models, returning with meaningful insights in terms of models and operation.

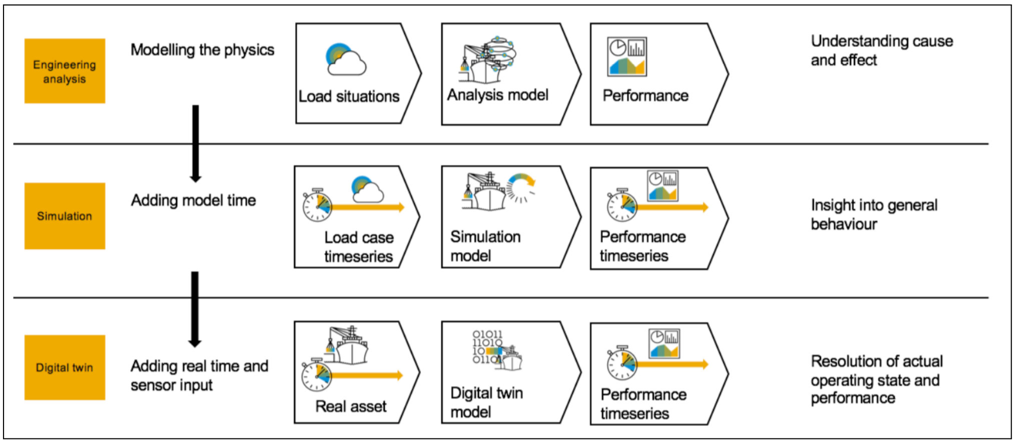


Figure 2.7: Simulation according to information level Erikstad (2017)

Beyond typical physical models consistently used in engineering problems, the digital twins intensifies the potentialities for data science application. Erikstad (2017) perceptively states in favor of the simultaneous use of machine learning methods to support the physical based simulations as a good practice for twin ship application. The author verify that high volume of data acquired through sensors in twin ship operations allows a good approximation of physical behavior if compared to the mathematical conjectures.

2.3 Data Management Applied to Maritime Systems

2.3.1 The Role of Taxonomies in Managing Ship Data

Taxonomies term is used to describe both the various perspectives of a ship and the hierarchical breakdown that organises the data under that view, also appearing in literature as hierarchy, they are central to ship lifecycle and to the digital twin specification as explained by Í. A. Fonseca and Gaspar (2020). The author argues about the importance of standards that provides a mapping between taxonomies and data content in a manner that facilitates understanding and use by humans and computer systems alike, allowing the use of heterogeneous ecosystem and connection to external services and tools.

In the following subsections we are going to introduce thee most relevant taxonomies applied to Ship data (SFI, SybsD and VIS), pointing out the relevant particularities from each model.

2.3.2 SFI Group System

The SFI (initials for the ship organization instute in norwegian) system was an early and successful taxonomy for organization of ship data and classification system, as a response to the challenges of exchanging data consistently inside and among organizations as pointed by Í. A. Fonseca and Gaspar (2020). SFI creation was possible thanks the results of research project lead by the Ship Research Institute of Norway, today named SINTEF. The goal was to help shipping and offshore companies getting control over their operations by providing a common communication language between the entities. Figure 2.8 shows a representation of the ship according to SFI taxonomy, containing the numerical indexation according to the component main group, sub group and detail code. Even those the SFI is an important applicable taxonomy it is centralized in the physical components, missing the representation of meta data, sensor information, calculation results and simulation equations, therefore making difficulty its fully in complex applications such as twin ship operations.

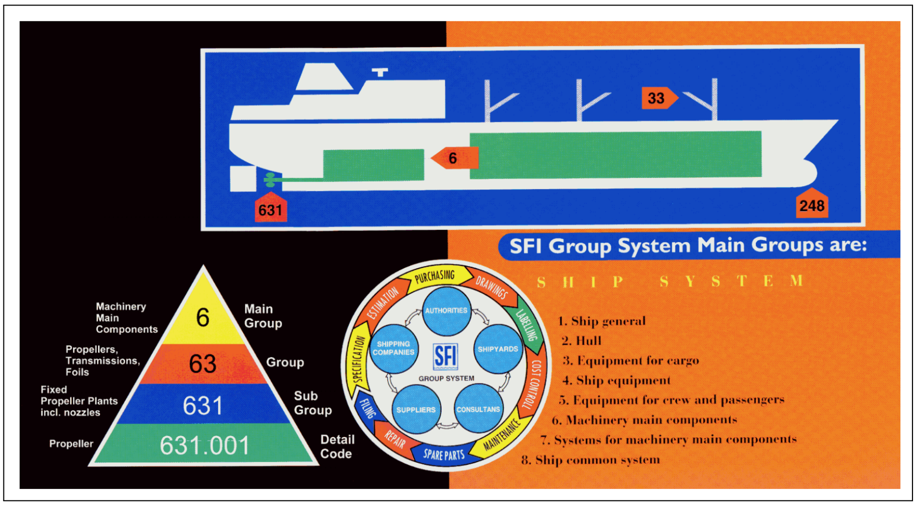


Figure 2.8: SFI structure, drawing provided by Xantic (2001)

The SFI group system is based on a hierarchical, numeric tag system that is guided by a strictly functional view of the ship that indexes components not by system, but by groups according to component function. The numeric tag system consists of three levels that are supplemented by a detail code for individual components and materials. This hierarchy is mapped to the indexed information, including drawings, specifications, and accounting registers for material and labour.

2.3.3 System Based Ship Design

To compose a more innovative ship design perspective a newer design model called System Based Ship Design (SyBSD) was proposed in contrast with the traditional project spiral model (Levander, 2012). The SyBSD has the advantages of straightening the design spiral, delaying the decision process and reducing the number of interactions to find a reasonable solution by beginning the project with the systems and the functions definition instead of locking the ship size into the early stage such as in project spiral model (Monteiro & Gaspar, 2016).

To accomplish its goals the SyBSD classifies the systems inside the ship into two main categories of functions: the ship and the payload. The ship function comprehend all systems necessary for the operation (e.g. engines, hull, fuel tanks), the task related system or payload systems include the services generation cash flow for the vessel (e.g. container space, cargo tanks, cranes). The Figure 2.9 shows a example of systems classification for a container vessel.

The ability to divide the systems into the task and ship related are an advantage from SyBSD in relation to the SFI model. Although that, there may be practical and behavioral reasons for using SFI classification together with SyBSD, in this case Levander, 2012 proposes a relational diagram into the two models shown in Figure 2.10, which translates both the models with a small inaccuracy in the process, since it is impossible to interchange perfectly the information between the two classification models.

2.3.4 Vessel Information System (VIS)

The Vessel Information System (VIS) model, for other hand, has a more functional than physical approach for ship system data structure than the SFI counterpart. As

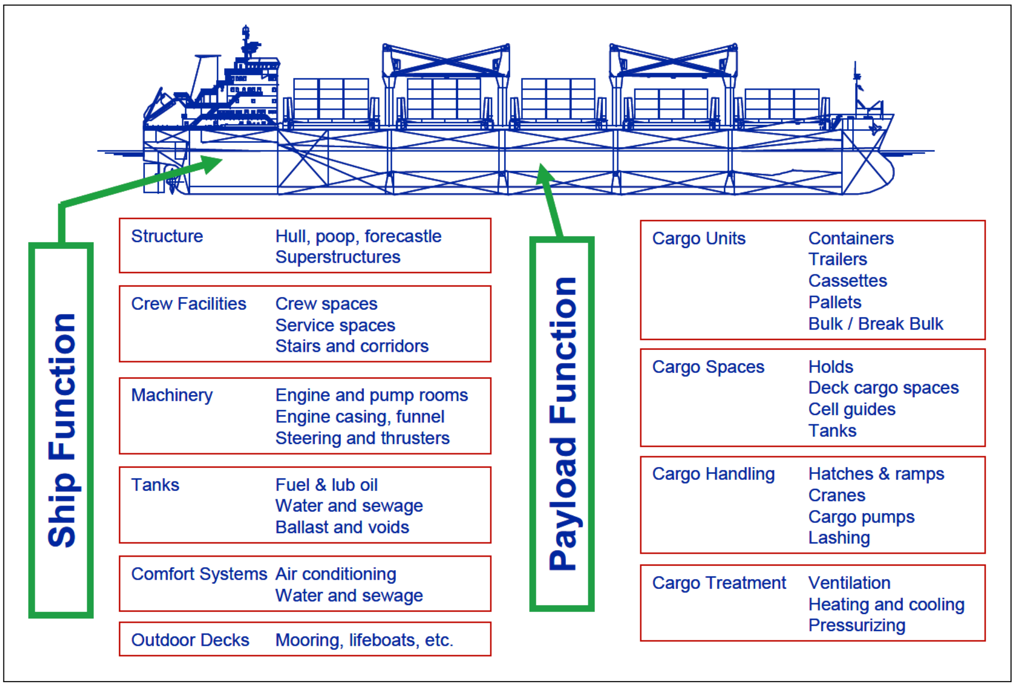


Figure 2.9: Division of the systems according to the ship function and the payload function for a container vessel (Levander, 2012).

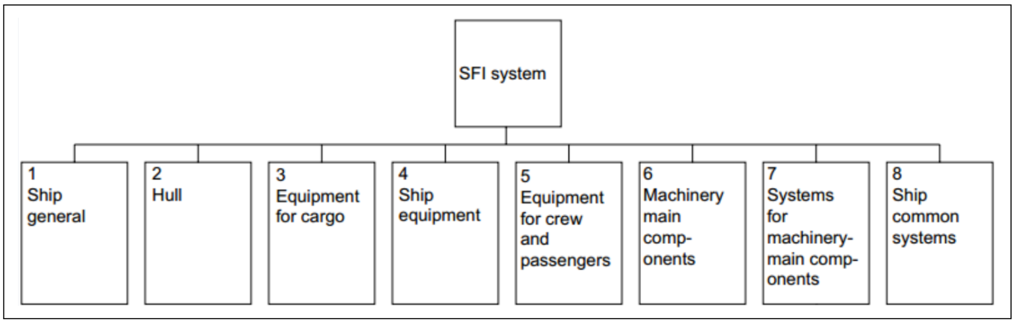
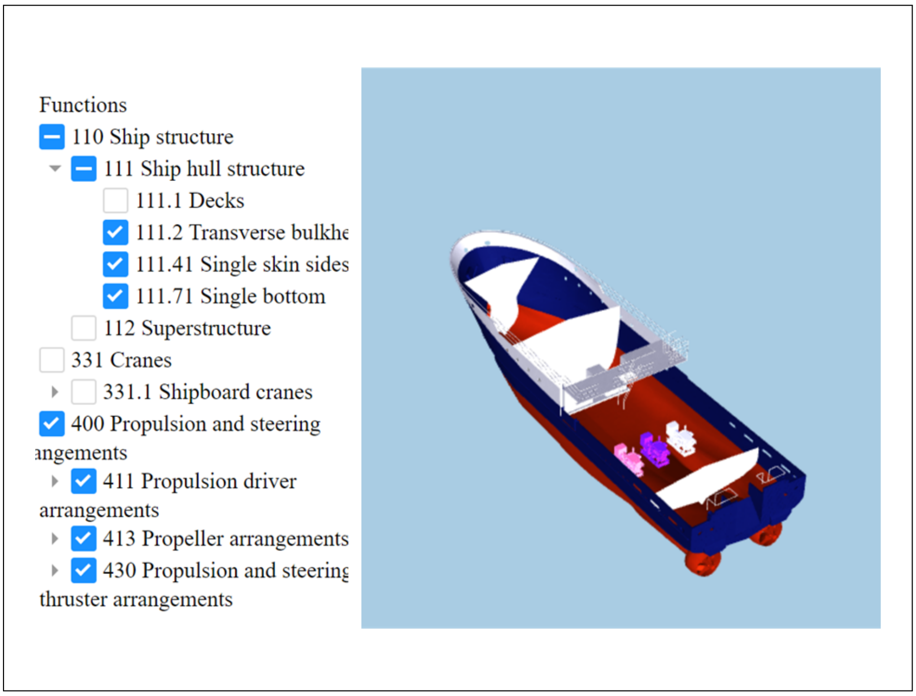

SFI system
1 Ship general
2 Hull
3 Equipment for cargo
4 Ship equipment
5 Equipment for crew and passengers
6 Machinery main comp-onents
7 Systems for machinery-main comp-onents
8 Ship common systems

Figure 2.10: Relation between SFI.

explained according to Vindøy (2020) "VIS model is based on a functionally oriented description of the vessel, comprising a hierarchical function structure, a library of components and definition of legal connections between function and components".

The functions represent the ability to perform or prevent certain actions, for example in ships is commons certain components be designed to ensure structural integrity, anchor the ship or provide propulsion thrust. Í. A. Fonseca (2020) demonstrate an application in which the Gunnerus project components is organized according to VIS perspective. Figure 2.11 shows the representation result with some functions filtered. The filtering is subject to the user activation for contextualizing process according to the considered study case.



Functions
- ▬ 110 Ship structure
  - ▬ 111 Ship hull structure
    - ☐ 111.1 Decks
    - ✓ 111.2 Transverse bulkhe
    - ✓ 111.41 Single skin sides
    - ✓ 111.71 Single bottom
  - ☐ 112 Superstructure
- ☐ 331 Cranes
  - ☐ 331.1 Shipboard cranes
- ✓ 400 Propulsion and steering ıngements
  - ✓ 411 Propulsion driver arrangements
  - ✓ 413 Propeller arrangements
  - ✓ 430 Propulsion and steerinǥ thruster arrangements

Figure 2.11: Example of structure filtering according to VIS, Í. A. Fonseca (2020).

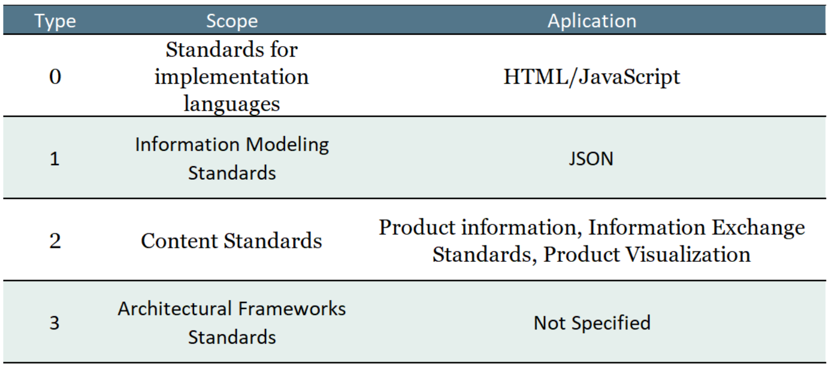
The components are the physical items in the ship, for example, diesel engines, generators and piping. Components may be assigned in the end of function hierarchy (also called function leaf). On the components group we can also have children elements, for example, the engine shaft can be a component of the diesel engine.

The final concept to be defined is the parameters. All components are subject to functional that are information related to the ship, and component parameters which are information specific for a system. For example, the functional parameters may be the Vessel name, while the component parameters may be the power, a design attribute from the engine component.

The platform will use those concepts to structure the data according to a VIS approach explained in Section 2.3.4 by the life cycle detailing design phase, since by this point there is the possibility for a more functional approach. The platform also will use this concept to modularize its components as will be explained further in Chapter 3.

2.3.5 Data Standardization

Í. A. Fonseca (2020) uses the typology of product information sharing and exchange defined by Rachuri et al. (2008) to define the information structure which best suits a web platform for a Product Data Management (PDM) system for R/V Gunnerus, using the current applications provided by Vessel.js library. Due the similarities in the tools applied, the structure used in that project will be incorporated into the current project as well. The topology of information are summarized in the Table 2.1.



Type	Scope	Aplication
0	Standards for implementation languages	HTML/JavaScript
1	Information Modeling Standards	JSON
2	Content Standards	Product information, Information Exchange Standards, Product Visualization
3	Architectural Frameworks Standards	Not Specified

Table 2.1: Typology of product information for the platform.

The type zero comprehend the standards for implementation languages, the chosen language for the platform is HTML/JavaScript based on the compatibility of those languages with the modern browsers and sharing content strengthening, even in distinct devices models like tablets or personal computers. Among several advantage of those languages is the versioning control capability, since every user can access its latest version through their own web browser. Another advantage is the popularity of those languages; JavaScript itself for examples was ranked with the most user number in 2019 (Carraz et al., 2020), reaching the mark of over 12 millions with a consistently increasing number of new users as show in Figure 2.12, meaning in practice a great ecosystem of libraries and learning content, easing the development and debugging process. For a more detailed explanation why those specific programming languages were chosen consult H. M. Gaspar (2016). For type one about the information modeling, it was chosen JavaScript Object Notation (JSON) because of its high connection with web technologies.

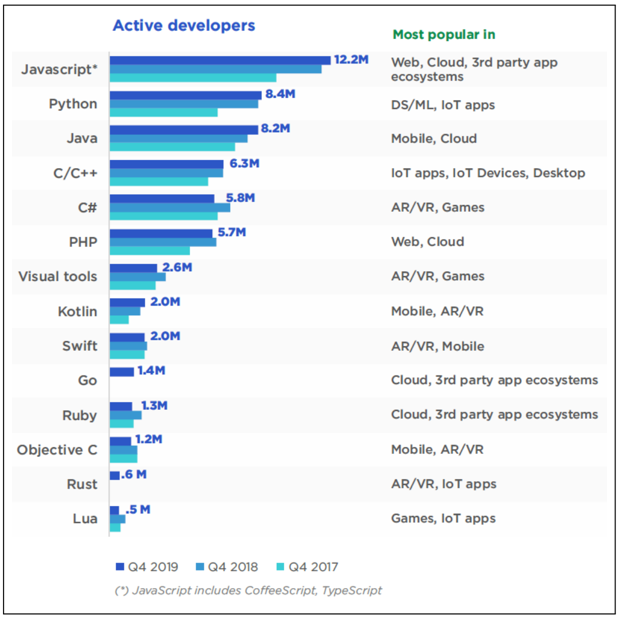


Figure 2.12: Active software developers globally, in millions (Carraz et al., 2020).

The Content Standards, designated as type two division, are subdivided into five categories but in respect to this project only the product information modeling, information exchange and product visualization will be considered, since they are closer related to the platform scope. The two other categories are standards for e-business and value chain support, and security standards. The e-business standards goes away from the business plan of an open platform and therefore will not be discussed in this report. Although security is a high concern for companies which want to protect their intellectual property, this subject will not be touched in this report by considering it a huge subject that deserves a wider and detailed explanation in a own dedicated master thesis, therefore, for this web platform it would be assumed that all the protocols of data exchange are satisfactorily safety.

Finally, the last type of information structure is the Architectural Frameworks Standards, which reconcile the different types of standard for creating integrated support systems. Here is not in the scope of the project to choose a specific model like those presented in Sowa and Zachman (1992) or DoD (2004), however, through the report several concepts will be used in Web Architecture to elucidate the integration framework and different standards used for data information.

2.4 Web-Based Platform for Simulation

2.4.1 Defining a Web Platform

First of all, it is crucial to identify what the term "web platform" means in the context of the web development to not generate a misconception with the different synonyms the word "platform" have. The term can be linked to O'Reilly (2007) which defined Web 2.0 as the web as platform, in that sense the new applications have not just the function to be a medium for publishing information as currently applied in what he defined as the web 1.0, but now they become an infrastructure to build applications on, allowing users, companies, developers to expand the platforms by adding content into a single architecture.

On the release of the Facebook platform Andreessen (2007), summarizes the platform term as "a system that can be reprogrammed and therefore customized by outside developers - users - and in that way, adapted to countless needs and niches that the platform's original developers could not have possibly contemplated, much less had time to accommodate". Therefore, in his definition what differentiates web network from web platform consists in its capacities to be programmable. To accomplish this goal the code source will be provided as open source under the MIT licence terms which are one of the most permissive options available. An open source code by definition must comply with several rules, being the most important to adhere to the platform definition the access to source code and derived work allowance (Open Source Initiative, 2007). Those two rules are guaranteed by hosting the project in GitHub where any user with a free subscription can access the code history as well as the source code. The GitHub platform also enhance the collaboration by providing management tools, communication posts, and wiki for documentation.

The attitude of licensing the project under the open source must not be interpreted however just as a simple platform attachment requirement, but as an strategy adoption that is in demand for big technology companies. Some examples of important open source tools among those used in this project are the Framework React supported by Facebook, bootstrap which was created by Twitter employees, and even GitHub that is the main open source code hosting was acquired by Microsoft. Some of the reasons why companies are becoming open source friendly is they are cheaper to develop and maintain, since proprietary code requires to sustain highly skilled employees that are not able to compete with the increasingly army of open source developers. To give a more precise estimation from the open source community dimension, Carraz et al. (2020) survey estimates that around 59% of all developers contribute to open source, and basically all developers use indirectly or directly an open source tool in their activities.

Open source projects are less susceptible to errors as well because they can be easier reported and corrected by the users/developers. Also, an open source code is a more appealing alternative to avoid industrial espionage, once the open script and the traceable development strategy eases the code function audition to ensure no undesired activity is being running behind the scenes. Such as in technology industry, the maritime sector can also adopt some initiatives in this field as well and this project aims to foster the adoption of the technology into this direction.

Evans et al. (2006) and Helmond (2007) put the general definition of software platform as its capacity of providing services to application, in web development this service is provided through Application Programming Interfaces (APIs). The API provide series of methods that can ease the development and can help third parties users to construct their projects upon. It is not in the scope of this project to provide an API, since it was chosen as objective the understanding about how digital twin data can be applied in a single architecture through ship life cycle, however, the author understand the importance of APIs in enriching the platform and incentive further work developed in this area.

2.4.2 Web-based Simulations

Once it was argued in last subsection about the importance of simulations on digital twin applications and the present project aims to construct a web platform that enhance the digital thread via simulation of a ship for project and operation phases, it is going to be discussed about the web based simulations in browser interfaces, one of the main elements that composes the proposed platform.

The web platform that aims to contain calculated and bind data from both, design and operation phases, must use graphical tools to facilitate the user interpretation. In the study held by H. M. Gaspar et al. (2014) is shown a data driven exemplification to ease designer evaluation, discussing the leverage of data visualisation with D3 library in ship projects and how the web based approach can ease the access of information between owners and project departments. An visualization exemplification using the ship motion theory was developed by Andrade and Gaspar (2015), in this application a GUI interface in a web browser is used to acquire a fast estimation of the ship response according to the article Jensen et al. (2004), the data is easily depicted through a collection of graphics that helps the designers perceive ship response.

On the speed processing capacity field, Í. A. Fonseca and Gaspar (2019) shows physical time domain equations cases, including maritime applications like six degree of freedom mooring systems solved by numerical methods using web based libraries with a response speed similar to traditional off-line software. Among the several conclusions, the author argues in defence of web based approach because its compatibility between several machines, sharing capacity and open source cultivation capability. The simulations used Vessel.js library to organize the states of the components, and many of the concepts depicted by his work will be integrated in this report together with the vessel.js library itself for the analysis and simulations of the models.

An evolution case in terms of complexity of the mathematical operations is the simulations produced by Escamilla i Miquel (2019) that involves hydrodynamic calculation of multi-bodies floating structures. The differential equations of movement were solved in the browser and were supported by the hydrodynamic coefficients calculated using the software Wamit. The coefficients estimation required a high processing capacity, discarding the possibility for real time calculations, to manage this drawback a partnership was made with Numerical Offshore Tank (Tanque de Provas Numérico, TPN) for processing support and knowledge exchange. The author also foster the idea for future works involving API usage to exchange data between the lab and the operator. This API would allow the calculation of those coefficients in different machines, returning back the coefficients results into database for the web simulation and visualization. A web platform could be a perfect place to contain such API and could manage to accomplish the integration task, the construction of a web platform as proposed in this project could be faced as an initial step to achieve this stage of exchange in simulations, having the database as a channel for information exchange.

2.4.3 Modularization in a Digital Twin Platform

The modularization design concept has as principle the construction of the product in detached subsystem with a corresponding interface in order to ease customization. This approach is a response from a competitive market, rapidly changing in technology, and demanding willing from customers to receive fast and effective personalized products since modular models can incorporate or modify semi components without disturbing the whole system function. An clear perception from this trend is the twenty one century automobile industry, where one non specialist can effortlessly outline several products varying from different classes (SUV, compact, sedan and etc), colors, fuel types, and brands that are only achievable by reusing components from different models through a modular perspective. This industry trend is also expressive in maritime sector, even those applied in a smaller scale if compared with automotive area, and it is more prominent for ships with high added value like supply vessels and cruise ships.

Hildre et al. (2010) insert the modular application as crucial element for innovation, giving successful examples from maritime industry. The author affirms the vertical chain approach is reasonable for integrated products with a low degree in innovation with the aim to accomplish a low price product. But once innovation and competition comes into place a horizontal chain with modular products that links the knowledge thought different stakeholders such as companies, universities and research centers is more effective. Far from a simplistic opposition between the modularization and integration concepts, the usage of each concept depends on the maturity degree of technology and companies smartly migrate between the two approaches depending on the development level. Nowadays, the maritime sector face the digitization process, the full or partial automation of operations and the usage of non pollutant fuels, pushing the sector into innovative ambient, consequently increasing the importance of modular application.

A software architecture that aims to follow holistically along with the advancements of maritime sector must be adaptable in order to incorporate the new subsystem assessment. Therefore, the code script ought be written in a modular form in order to interconnect with the function of the modular systems in the ship. The figure 2.2 is a simplified scenario case for the relation of modular battery inclusion and the insertion of new code module to evaluate this new system, the table pays attention to the purpose of the systems involved and the user analysis required, distinguishing it in three categories: the component before modification (A), the introduced modular component (B), the resulting system (A + B). The new incorporated subsystem must not affect the whole function of the ship as much as the new simulation code module must not affect the main function of the web platform. In summary the purpose of the systems in the three phases (A, B, and A+B) must not be effected by the integration of the components both in physical and digital spheres.

		Pre Modification (A)	Introduced component (B)	Resulting System (A + B)
Physical System	Representation			
Physical System	Purpose	Transport goods and people through ocean.	Store electrical energy.	Non pollutant form to transport goods and people.
Code Script	Representation	`<MainShip>` `<MainShip />`	`<Battery />`	`<MainShip>` `<Battery />` `<MainShip />`
Code Script	Purpose	Database interaction and calculation of ship information.	Calculation of the energy storage and consumption.	Relation among the several systems and electrical energy consumption.

Table 2.2: Component purpose in the three phases for the example of modular battery insertion together with its respective code script.

2.4.4 Object-Oriented Approach Applied to Ship Data

One of the responses to modularization in software development is the object-oriented approach. Object-oriented programming allows storing both data (information) and methods (logical and mathematical operations) inside an object, providing a cleaner way to execute, and easing the code reuse. The object oriented approach is particularly useful for ship industry because the object, in this case represented physically by the ship, is composed of several subsystems that as extension can be stored in a database with its methods representing the several physical modeling for shipping analysis. Í. A. Fonseca (2016) and H. M. Gaspar (2016) shows with more details how the object oriented approach was used to structure the Vessel.js and why this approach fits well for maritime industry.

To organize the data according to an object oriented approach the platform will be written using a modularized standard in accordance with the 'Class ES6 description' (n.d.) revision for JavaScript language, using the concept of classes. Following code is a general example about how the concept of classes can be used in an object-oriented paradigm under a naval architecture context, the constructor is a function that will run every time the class is created, in this example it will assign the length, beam and draft to the object. The functions assigned to a class are called methods and they enable logical and mathematical operations that are related to this class, such as `calculateVolume()` in the Ship class example.

```
 class Ship {
    constructor(Cb,L,B,T) {
        this.L = L
        this.B = B
        this.T = T
        this.Cb = Cb
    }
    calculateVolume() {
        return this.Cb*this.L*this.B*this.T
    }
}
```

In practice after the definition of the class it can be used to create several distinct objects, the listing bellow gives an generic example in how to call a class and how the variables are structured under a object oriented approach.

```
let ship1 = new Ship(0.7;100;20;5)
let ship2 = new Ship(0.85;200;40;10)

console.log(ship1.L) // 100
console.log(ship2.L) // 200
console.log(ship1.volume()) // 7000
```

The class can be extended to accommodate a specific subclass with its methods. bellow we have an example about an extension in the ship class. The term `super` calls the constructor function of the parent function, in the exemplification it calls the constructor of the class `Ship`.

```
class Titanic extends Ship {
    constructor(T) {
        super(0.66, 269.1, 28.2, T)
    }
    whenWasSinked () {
      return 'Titanic was sinked in 1912.'
    }
}

let famousShip = new Titanic(10.5)

console.log(famousShip.L) //269.1
console.log(famousShip.T) //10.5
console.log(famousShip.calculateVolume()) //52589.1366
console.log(famousShip.whenWasSinked()) // 'Titanic was sinked in
    1912.'
```

Chapter 3

Web Platform

During this chapter, we are going to extend through the series of tools and structures that compose the web platform, using the concepts defined in the literature reference. As explained before, a web platform must have the ability to be expandable and the current project uses a series of modules for this purpose. For this reason, this chapter will explain the web platform composition, showing their relationship with other elements.

The section will present each module, starting from the web platform architecture modules until the Vessel.js structure.

3.1 Web-Based Development

3.1.1 Web Platform Architecture

Even those the system architecture is not the scope of the this project, it is interesting to have knowledge about the structure used to depict the main tools applied for this application. It will not be on the ambition of this project to establish the best possible web platform architecture for digital twin purpose, restricting the project purpose to this application itself. Different technologies could have been used instead, that in practice would enhance the web platform potential, increasing even more the collaboration between stakeholders such as the one proposed in Hatledal et al. (2019).

The web platform will be constructed using the stack popularly called MERN, that is an acronym for the initials from the tools that compose this architecture (**M**ongo DB, **E**xpress, **R**eact, and **N**ode.js). Those tools are a set of open source components that together provide an end-to-end framework for building dynamic web applications; starting from the top (code running in the browser) to the bottom (database) as defined Morgan (2020). Among several advantages, this stack applies only Javascript language with JSON notation for front end (user side) and back end (server side), making easier the developer adaptation once it is not required several programming language skill. The architecture uses a three tier architecture as shown in the 3.1, giving the advantage for architecture upgrade or changed in one of its tier without putting the whole system at risk.

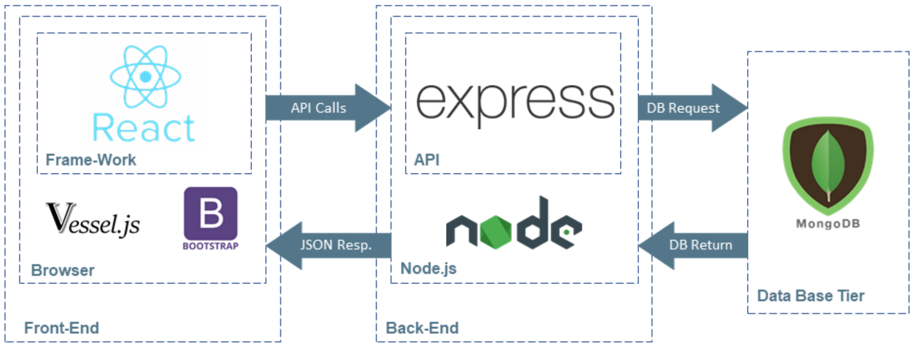


Figure 3.1: Three-tier MERN web platform architecture scheme, categorizing the most important tools for this project according to the respective domain.

On the following paragraphs will be briefly explained each of the tools used in the web platform development and its location according to the three tears architecture with a short explanation about each level. All the tools source code can be accessed through GitHub which are licensed under open source paradigm.

1. Front-End: related to the user interface layer, here the programming logic is passed to the end user in a user-friendly way.

   (a) React: A declarative and flexible JavaScript library for building user interfaces. React modifies the front end according to the state, rendering just the right components for each change. The building process is encapsulated in components enhancing modularization. `https://reactjs.org/`;

   (b) Vessel.js: is a JavaScript library for conceptual ship design with an object-oriented pattern. Vessel.js represents the vessel as an object, which is used to simulate different functionalities and behaviors. Currently, the library includes methods for resistance, seakeeping, hydrostatic and stability calculations. `http://vesseljs.org/`;

   (c) Bootstrap: Open source front-end toolkit, featuring responsible grid system and prebuilt components. `https://getbootstrap.com/`;

2. Back-End: Server side layer to handle data storage and business logic.

   (a) Express.js: Node.js web application framework that provides a robust set of features. Provide methods and middleware for creating quickly and easily robust APIs. `https://expressjs.com/`;

   (b) Node.js: Cross-platform back-end that executes JavaScript code outside the browser. `https://nodejs.org/en/about/`;

3. Data-base tier: a logical organization of information collection.

   (a) MongoDB: A document database that stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time

Since the development of web platform itself is not in the scope of this project, the web platform will be based upon the open provided framework in Schiff (2020). The project will extend and adapt the tools to accommodate to the maritime context and handle three dimension modeling.

3.1.2 Structure of Web Platform Repository

The web platform repository divides its files according to its features and routes. The Figure 3.2 shows a schema with the web platform repository structure, grouped by color according to its purpose. The repository can be conferred for the version presented in this report at `https://github.com/ferrari212/Platform-Twin-Ship/tree/v1.2.1`.

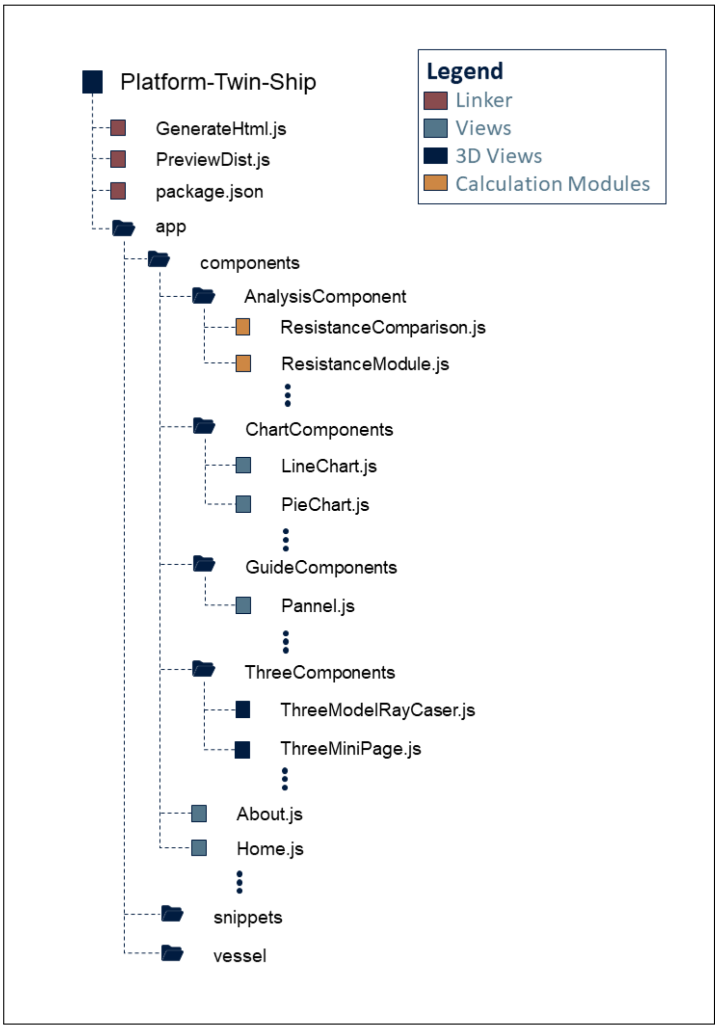


Figure 3.2: Web platform structure according to its folders. Some of the files were hided for simplification

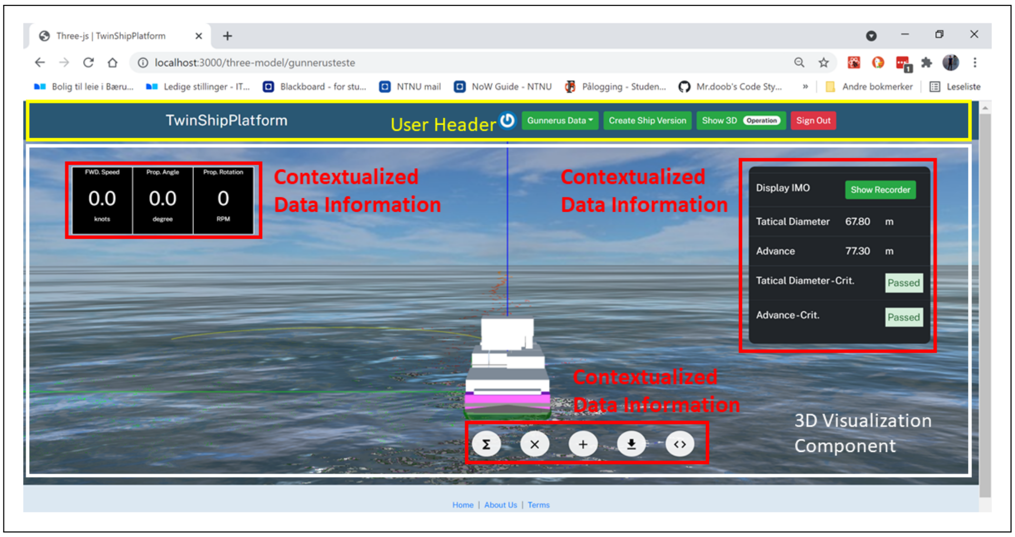
Links are the files which are part of the process for integrating the several dependencies and modules together. The Views corresponds to the modules that are responsible to display the information to the user, both in textual and graphic format represented generically by the Views group or by the three dimensional interactive drawings represented by the 3D views group. Finally the Calculation Modules are the ones responsible for the heavy calculation functions involving the Vessel.js library.

The components modules were divided in four folders: AnalysiComponents, Chart-Components, GuideCompronents, and ThreeComponents. The analysis component are the modules responsible for the calculation of the analysis, as its name suggests, of each ship in terms of resistance, response operation and hydrostatic. The chart components are the ones created to handle the graphics creation using the Chart.js library in its react form. The components inside GuideComponents folder are the ones which contains the displayed interface for the user, such as the panels with simulation information. Last folder is the ThreeComponents, that as its name suggests, is the folder made to contain all the 3D visualizations involving the use of Three.js library. The remaining modules which contain the visual user interfaces are inserted generically in the components folder.

3.2 Graphical Interface and Visualizations

3.2.1 Modularization of the Web Platform Components with React.js Framework

One advantage of using React.js framework is its capability for modularized code application, where each part of the script is separated in components and assembled together to form the desired user view. This encapsulation also allows the code to be more efficient, once the React framework manage the loading of the components in second plane making the update of the modules according to the user state view instead of changing the complete DOM in the traditional approach. The Figure 3.3 shows a interface case of the web platform highlighting each of its components. The white container contains the module responsible for the visualisation of the ship with the extracted information from the data base. The user header component presented in yellow can be changed according to the user and context, contextualizing the web platform according preset permissions. The red containers shows several components that present contextualized data, in this case the modules show the results from a maneuvering simulation with the ship states, a contextualized "lifecycle" bar that is the series of buttons controllers in the bottom part of the web platform, and a result from the maneuvering assessment. A further explanation about graphics modularization will be given in Section 3.2.2.


Three-js | TwinShipPlatform
localhost:3000/three-model/gunnerusteste
TwinShipPlatform
User Header
Gunnerus Data
Create Ship Version
Show 3D Operation
Sign Out
Contextualized Data Information
Contextualized Data Information
Contextualized Data Information
3D Visualization Component
Home | About Us | Terms

FWD. Speed 0.0 knots | Prop. Angle 0.0 degree | Prop. Rotation 0 RPM

Display IMO — Show Recorder
Tatical Diameter 67.80 m
Advance 77.30 m
Tatical Diameter-Crit. Passed
Advance-Crit. Passed

Figure 3.3: Example of modules for the web platform. Each colorful block is layered according to its function and represents one component with detached function.

Each component module can be programmed and modified separately, this functions allows a freedom for different layouts application, enhance cooperative work and eases the insertion of new modules. The Listing provided in Annex A is a template of the modules for generating the main 3D visual objects using Three.js such as the one presented inside the white box by Figure 3.3, using as structure model the script provided in Vorontsova (2019). The module is handled by the browser in four phases:

1. `constructor(props)`: triggered right before the module enters in the browser. Pass all the properties from the parents components into the module and calculate the main information to generate the current states. The `props` is a variable defined in the parents components which for the ship visualization will be the `user` data, which contains among other user information a string for the JSON object of the current ship version;

2. `componentDidMount()`: triggered right after the module enters in the browser. Insert all the elements referent to the module itself, for example user events listeners;

3. `componentDidUpdate(prevVersion)`: method to be triggered each time the component data is updated, for example with the forced exchange with database or change in the ship version analysed. The `prevVersion` is the preview state of the web platform and can be used for comparison with the current state;

4. `componentWillUnmount()`: triggered after the component is unmounted. This method serves to clear out the event handler functions and pending requests. For this project, the main functionality is to clear the routine related with the three dimension scene rendering, therefore, when the user change the state or the path the new animation function will not be running simultaneously with the previous in the background.

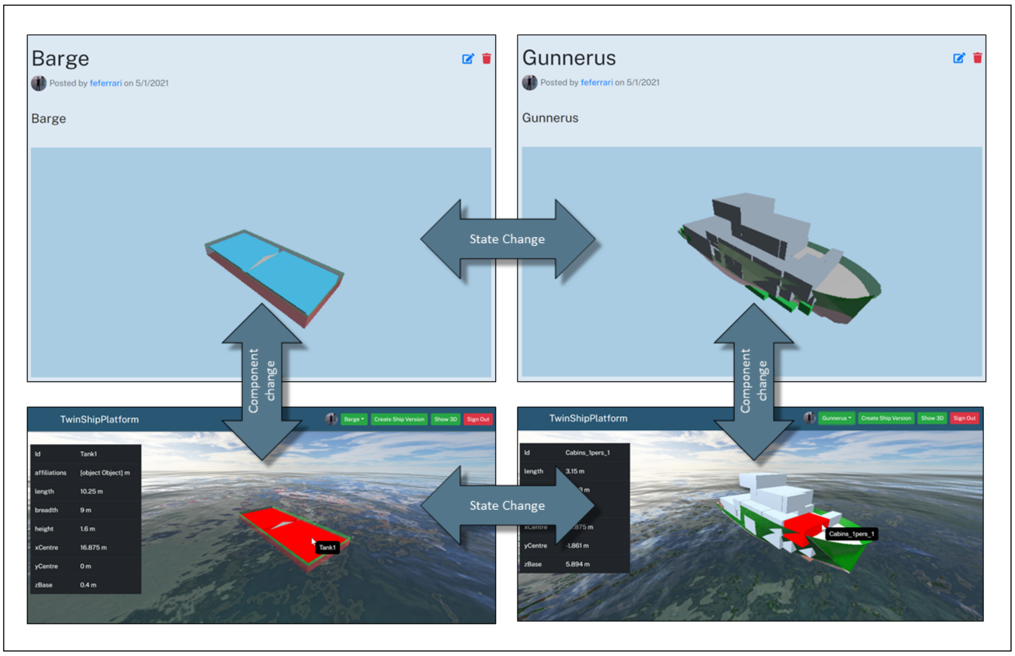
Inside the phases the code triggers different methods that aims to achieve a specific purpose. The table 3.1 shows the minimum required methods for creating the ship visualization, using the methodology described in Vorontsova (2019) which instructs how to integrate Three.js library with React.js framework. The current methods can be modified and new methods can be inserted to accomplish other non established purposes, preserving the structure of the previews code via the modularized strategy.

Basically the modification of the methods allows the freedom to develop whatever type of simulation or visualization. As an example case, Figure 3.4 shows in the vertical direction the page change by modifying the methods inputted inside the component. The modification presented in horizontal direction shows the effect of ship state change triggered in `componentDidUpdate(prevVersion)`.

On the top component in Figure 3.4, the same module is presented for two different ship for version verification purpose, highlighting the author, title and the description while suppress the features linkage to simulation such as the sea environment. The bottom figures show the same ship versions in the simulation context with a sea en-

	Method Name	Purpose
Constructor	`Super`	Instantiate the parents props into the component states.
ComponentDidMount	`SceneSetup`	Scetup camera and controls.
	`handleWindowResize`	Maintain resizing screen ratio.
ComponentDidUpdate	`addShip`	Insert defined ship model
	`removeShip`	Remove chosen ship mode
	`startAnimationLoop`	Start the Three.js render
	`addScenario`	Insert environment scenario

Table 3.1: Minimum required methods for ship presentation and its triggering positions according to the mounting process.



Id	Tank1
affiliations	[object Object] m
length	10.25 m
breadth	9 m
height	1.6 m
xCentre	16.875 m
yCentre	0 m
zBase	0.4 m

Id	Cabins_1pers_1
length	3.15 m
[illegible]	[illegible] m
xCentre	[illegible].875 m
yCentre	-1.861 m
zBase	5.894 m

Figure 3.4: Variation of the page view by change in state and component.

vironment surrounding it, this new context allows the interaction of the user with the 3D objects, presenting in form of tables the information about pointed compartments of the ship.

3.2.2 Graphics Modules

Humans fells more comfortable in understanding relations by visual sense, for that reason there was always the willing to represent data in a visual form. Charts are a visual manner to organize data in intelligible way, giving a broad overview about the phenom studied, highlighting for specialized eyes possible flaws in the theories and presenting a way for contrasting information. The proposed web platform has the objective to be a reliable source for data management, using charts to translate complex information in a way that are meaningful for ship evaluation.

Similar to the previous features for three dimensional visualization, the graphics are constructed with a modular perspective, in this particular case relying in Chart.js (version for react) open source library, accessible in `https://reactchartjs.github.io/react-chartjs-2/`. The library is in a certain extension malleable, which means they can be fitted for several visualizations purposes, in that sense the methods in the libraries are not ready to be used in a modular project and must be refactored to fit the particular format this project requires. The modules for graphic generation and integration are presented in a full form in Appendix B.

In this subsection we will attain to show how the modules are used in practice. First, it must be created a variable for hold the `DataChartStructure`, which will contain the information about the titles and axis. Then the title and the rgba color must be inserted in the method to `pushDataSet()` returning the dataset variable which will contain the numeric values of the charts. Finally the data must be parsed to the `data` and `dataSet` to represent the x and y axis values:

```
var data = new DataChartStructure()

var dataset = data.pushDataSet("Title", "rgba(1, 28, 64, 0.6)")

var xValues = [1, 2, 3, 4] // x-axis values
var yValues = [1, 4, 9, 16] // y-axis values

for (i = 0; i < xValues.length; i++) {
    dataset.push(yValues[i]) // Insert the y number in the dataset
}

data.setLabels(drafts) // Insert the x number in the data
```

After the data is parsed to the structure it is possible to return it inside a module graphic, the example bellow returns the four type of graphics possible for Chart.js:

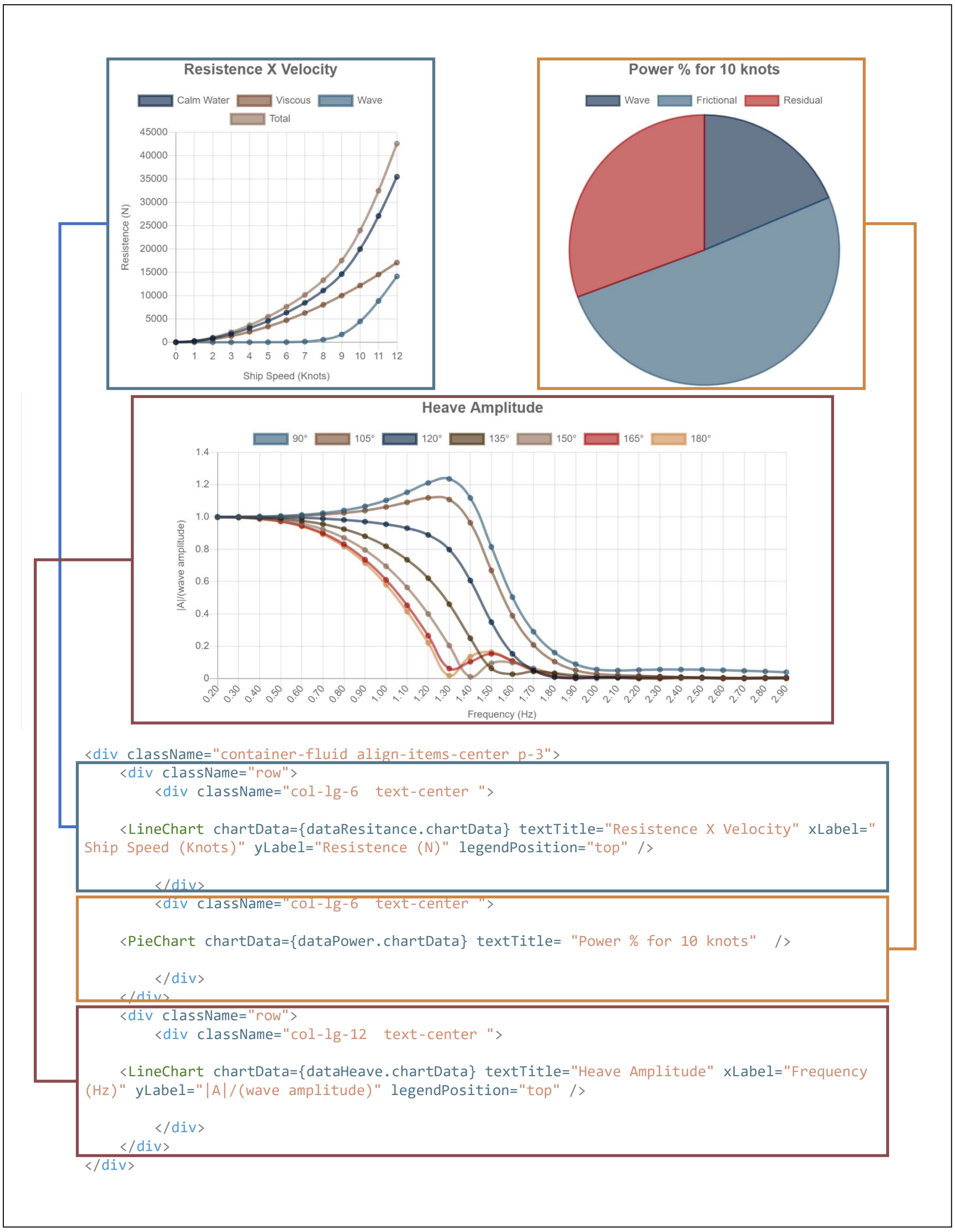
```
return (
    <LineChart chartData={data.chartData} textTitle="Text Title"
       xLabel="x-Axil title" yLabel="y-axis title" displayLegend="
       true" />

    <BarChart chartData={data.chartData} textTitle="Text Title"
       xLabel="x-Axil title" yLabel="y-axis title" displayLegend="
       true" />

    <PieChart chartData={data.chartData} textTitle="Text Title" />

    <RadarChart chartData={data.chartData} textTitle="Text Title" />
    )
```

Figure 3.5 shows the relation between the code module return using the Chart.js and Bootstrap libraries together with the resulting interface. To create a new evaluation chart it is just necessary the insertion of a new module in the return code without changing specifically the rest of the script since all the graphics are created in a modular perspective.



```
<div className="container-fluid align-items-center p-3">
    <div className="row">
        <div className="col-lg-6  text-center ">

    <LineChart chartData={dataResitance.chartData} textTitle="Resistence X Velocity" xLabel="
Ship Speed (Knots)" yLabel="Resistence (N)" legendPosition="top" />

        </div>
        <div className="col-lg-6  text-center ">

    <PieChart chartData={dataPower.chartData} textTitle= "Power % for 10 knots"  />

        </div>
    </div>
    <div className="row">
        <div className="col-lg-12  text-center ">

    <LineChart chartData={dataHeave.chartData} textTitle="Heave Amplitude" xLabel="Frequency
(Hz)" yLabel="|A|/(wave amplitude)" legendPosition="top" />

        </div>
    </div>
</div>
```

Figure 3.5: Relation between graphics interface and script modules.

3.3 Data Storage and Information Structure

On this section will be discussed the organizational pattern used for the data storage in MongoDB database and the repository structure for the web platform and Vessel.js library. MongoDB is based on the principle of relational data base, which mean a series of table can be related by one another according to a respective id. The importance behind the organization awareness is maintaining a pattern to keep the maintainability of the components.

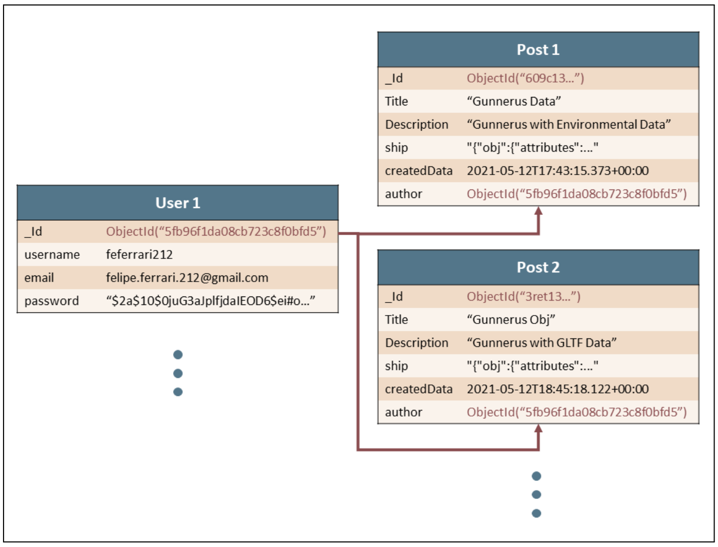
3.3.1 Mongo DB information structure

The database structure is divided into two collections: the `users` collection containing information about the user such as the username, the e-mail and password (protected automatically by Mongo cryptography) and the `posts` collection which stores the information about the ship versions.

Users are created by the subscription in the web platform on the main page. The versions are created in the new versions post page, and every new version is marked with the user id as a way to relate the versions with the users. The Figure 3.6 shows the relation between the two examples.

Each post contains the version data that belong to the ship stored in the data base as a string. The object contains all the four elements objects that constitute the ship physical behavior (obj, propeller, powerPlant, and the man). The mounted attribute object is show, with some of their structures hidden for simplification, as following:

```
ship : {
    obj: {
        attributes:{...},
        designState:{...},
        structure:{...},
        baseObjects:{...},
        derivedObjects:{...}
    },
    propeller: {...},
    powerPlant: {
```



User 1	
_Id	ObjectId("5fb96f1da08cb723c8f0bfd5")
username	feferrari212
email	felipe.ferrari.212@gmail.com
password	"$2a$10$0juG3aJplfjdaIEOD6$ei#o..."

Post 1	
_Id	ObjectId("609c13...")
Title	"Gunnerus Data"
Description	"Gunnerus with Environmental Data"
ship	"{"obj":{"attributes":..."
createdData	2021-05-12T17:43:15.373+00:00
author	ObjectId("5fb96f1da08cb723c8f0bfd5")

Post 2	
_Id	ObjectId("3ret13...")
Title	"Gunnerus Obj"
Description	"Gunnerus with GLTF Data"
ship	"{"obj":{"attributes":..."
createdData	2021-05-12T18:45:18.122+00:00
author	ObjectId("5fb96f1da08cb723c8f0bfd5")

Figure 3.6: Collections of the web platform. The objectId of the author is a link to assign the posts element to the author in a context known as relational databases.

```
        main: {...},
        auxiliary: {...}
    },
    man: {...},
    data: {...}
}
```

The `obj` that represents the information value for the ship compartments, the propeller object that contains the data for to describe the `propulsor` in the B-troost series and the `powerPlant` that describes the power engines inside the ship were extensively described in Í. A. Fonseca (2016) and Oliveira (2021a). The new objects introduced for this purpose are the `man` and the `data`.

The `man` object stands for the maneuvering model values. An example of the information used in this object can be depicted bellow:

```
man: {
    distHel: 15,
    m: null,
    I: null,
    initial_yaw: 0,
    initial_angle: 0,
    M: null,
    N: [
        [ 0, 0, 0 ],
        [ 0, 5.5e4, 6.4e4 ],
        [ 0, 6.4e4, 1.2e7 ]
        ],
    helRate: 0.5,
    rudderRate: 5,
    maxPropRot: 210,
    maxTorque: 400
}
```

Each of the key values are explained with a brief description, an accepted type of value and the unit for the variable by the Table 3.2 bellow:

Key Name	Descritption	Accepted Types	Unit
distHel	Longitudinal distance between propeller and Center of Gravity	Number	m
m	Mass of the ship	Number or Null	kg
I	Maintain resizing screen ratio.	Number or Null	kg.m^3
initial_yaw	Initial angle of yaw	Number	deg
initial_angle	Initial rudder angle	Number	deg
M	3 X 3 Matrix of Inertia	3X3 Array or null	-
N	3 X 3 Matrix of Damping	3X3 Array or null	-
helRate	Helix rotation variation rate	Number	Hz/s
rudderRate	Rudder variation rate	Number	degree/s
maxPropRot	Maximun rotation rate	Number	RPM
MaxTorque	Maximun torque	Number	N.m

Table 3.2: Maneuvering model keys description.

The `data` object will contain an array with the information acquired by the sensors required for the twin ship operation. In this project the values of data will be limited to the maneuvering experiments as explained in details in Section 5.6. The data structure is shown bellow:

```
data: [{
        'ṭ': 1540291957409,
        'x': 0.0,
        'y': 0.0,
        'u': 6.97,
        'v': 0.43,y,
        'yaw': 312,
        'RPMO': 55.406166000000006,
        'Azimuth': 1.616903
    },
    ...
]
```

Each element in the array contains the time `t` the data it was acquired, its relative position from the initial point in meters `x` and `y`, the `yaw` angle, the velocities in longitudial `u` and transversal `v` directions, the propeller rotation order `RPMO` is a percentage of the of rotation according to the maximun allowed RPM (in the case of Gunnerus this value corresponds to 203 RPM) and finally the `Azimuth` which represents the propeller degree in the situation.

3.3.2 Vessel.js structure

Figure 3.7 shows the repository structure of Vessel.js, that is divided in three types of fundamental elements: Classes, FileIO, and Math (H. M. Gaspar, 2016). The Classes folder contains the scripts that are responsible for data ship handling for different entities and states. The FileIO contains functions for file exchange such as the download and upload of ship data. Finally the Math folder contains methods for the mathematical calculation that are fundamental for the analysis performed by Vessel.js classes such as vector basic operations, geometry calculation and numerical solution.

The Classes modules, presented in Figure 3.7, were labeled according to its publication. The First Release achieved by H. M. Gaspar (2016) in the version v0.13 includes the primary modules necessary to manage ship data, hydrostaic calculation, and weights and centers definition. The second big expansion was held by Í. A. Fonseca (2016) which integrated the analysis/simulation response amplitude model of the Jensen et al. (2004) in `WaveMotion.js` file, with the possibility to extend to irregular ocean patterns with the handling of `WaveCreator.js` script, and using the main ship machinery and the resistance knowledge developed by Holtrop (1984) to assess the consumption and resistance calculation `FuelConsumption.js` and `hullResistance.js` . Finally comes the `Manoeuvring.js` model contrasted in the figure with a golden color that will be implemented in this report.

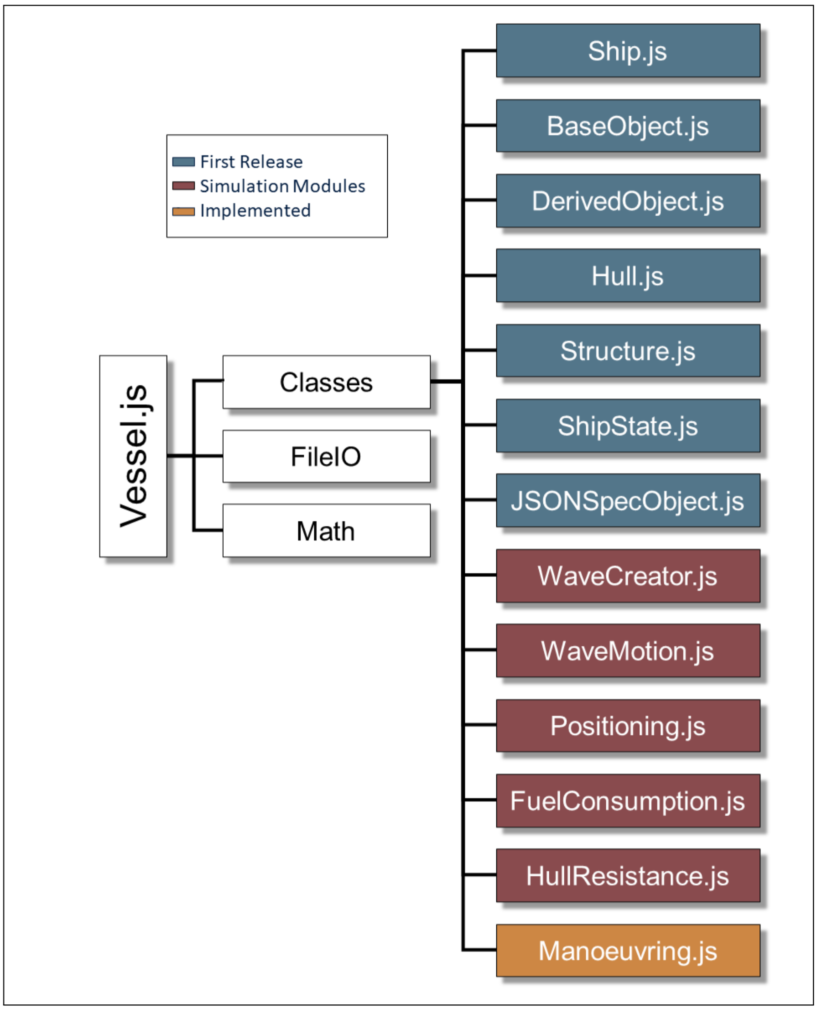


Figure 3.7: Vessel.js structure compatible with the version simultaneous to this report.

Chapter 4

Application of the Web Platform in Ship Design and Operation

4.1 Context of the web platform in ship design

The web platform must contain three phases: preliminary design, detailing design and operation (digital twin) as a plan to turn the web platform in a tool for strengthening the digital thread. In a the design phases (preliminary and design) the web platform is planned to accommodate the PSD methodology as discussed before during Section 2.2. The idea is to use the analysis and simulations on the web platform as a tool to ease the information loop in the Block Design method, such as the one proposed in the Figure 2.3. The web platform has as input the space inventory and information presented in the general arrangement, for example, the water line length. Those inputs are also required for the Vessel.js calculations and are a minimum requirement for the visualization of the geometric definition.

The Figure 4.1 is a detailed workflow web platform application for the simulation and visualization phase in the Building Block Design located similarly inside the region represented by the red polygon in the Figure 2.3. On light blue rectangles is shown the modules for the design evaluation which are in most case already imple-

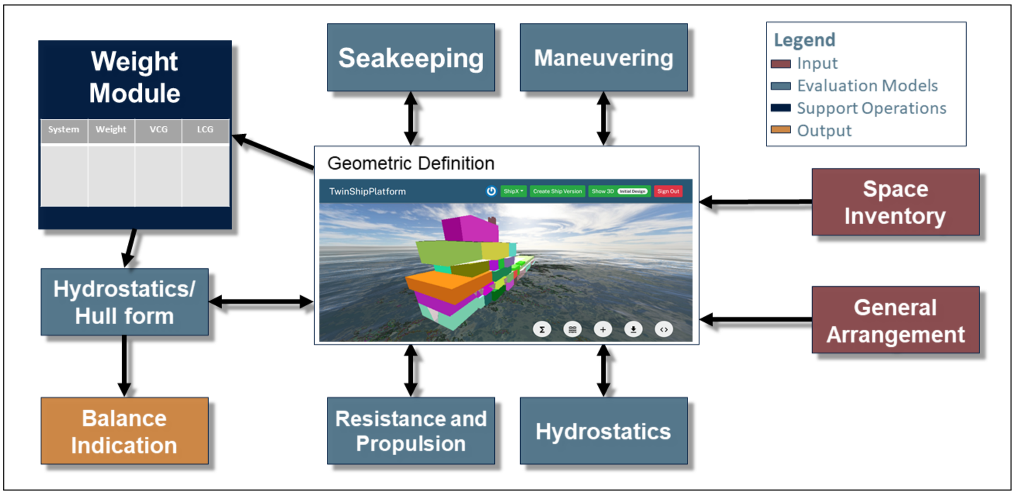


Figure 4.1: Application of the web platform in the Building Block methodology process. Inspired in the model for surface vessels of Andrews and Dicks (1997).

mented in the Vessel.js librarys (the exception is for the Maneuvering method which is going to be coded along with this project as a collaboration example), together with the support operations represented by the weight module in dark blue in the Figure 4.1. As output from the hydrostatics hull form is the balance indication that can be used in the next phases of the project.

The manoeuvring model will be partly assimilated from my previews work, involving the guiding creation to use the Vessel.js library (Oliveira, 2021a). While the cited project is more concerned about the usage of the manoeuvring model and the user commands interaction, here we are going to explore the mathematical equations, discussing about the simplifications and the tools backing the implementation.

4.2 Web Platform Collaboration Approach

Although in the current version the web platform contains the most important parts of the ship analysis models, they must always be subject to improvements by the insertion of new modules or the current models revision.

New mathematical modules can be done by the insertion via Vessel.js in the web platform back end, such as the one succeeded with the Maneuvering model. This type of expansion is classified as one of the three forms for collaboration that the web platform was designed to foster. The two other ways for expansion are via the insertion of new content in the front end by the upload of new ship models and interfaces, or by data exchange with the data base. Those three types of modification are correlated with the three tears of the architect depicted in Figure 3.1 and are classified as forms for collaboration with the web platform according to this project definition.

The Figure 4.2 elucidates those three directions the user can take to implement the web platform in respect with the tier in the architecture the User/Developer interacts with. In this example, the User A improves the web platform by simply inputting new ship version or by modifying the front end open source code stored in the GitHub repository, the own system of verification and approval from GitHub helps to guarantee that the implemented new functions are reviewed by its peers, reducing the chances of bugs and out of context contents. The second form to improve indirectly the web platform is by upgrading the physics and mathematical models in Vessel.js library, represented in the Figure 4.2 by the User B, as matter of fact this improvement is not strictly restrict by the web platform itself, since the library which feeds the web platform with the mathematical models is the one being improved, therefore this form of collaboration is a more indirectly approach. Finally, the last way to improve the web platform is by exchange data with the database, such as shown for User C, this data could be acquired through sensors or heavily expensive computation that are impossible to be calculated inside the web platform.

4.3 Web platform User Interface

On this section we are going to introduce the tools and interfaces the web platform provides, it is highly recommended for the reader to access the web platform hosted in https://twinship.netlify.app/ (Accessed in 2021-04-04) to have a real experience with the features. The interface has the function to be a bridge between the user and the internal logical commands, ideally the user interface must not hamper the functionalities access by inputting enormous decisions possibilities. For this purpose, the appearance of the web platform was designed to suppress tools depending on the

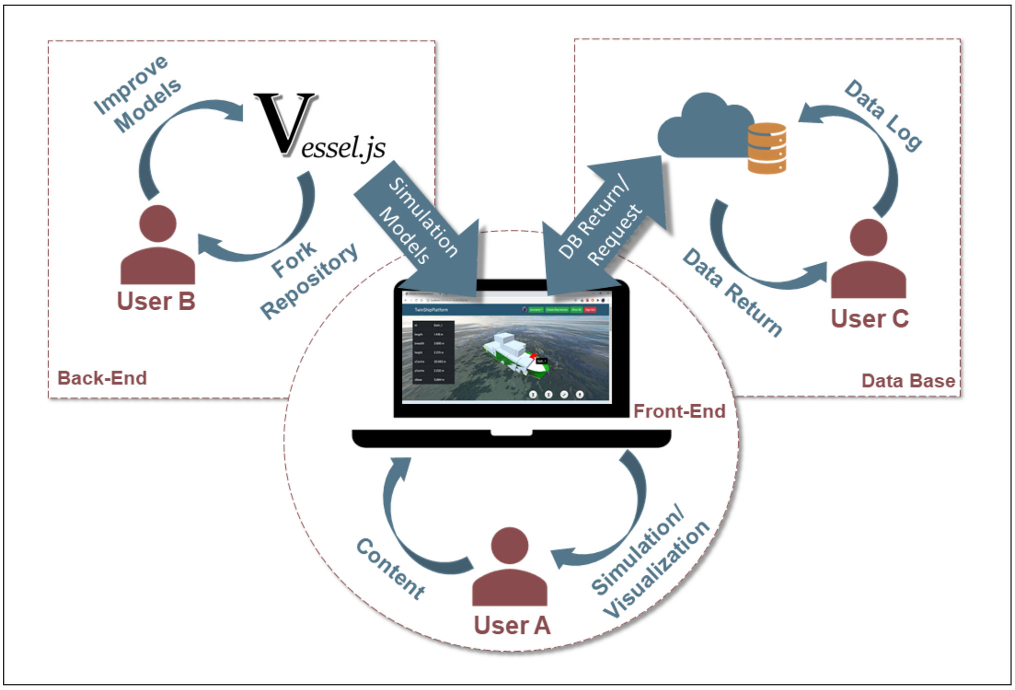


Figure 4.2: Three Possible interactions for collaboration. The User/Developer can increment the web platform content in three ways in respect with the three tears the web platform was designed: modifying the ship content in the front end of the web platform (User A), improving the mathematical and mathematical models present in Vessel.js lybrary (User B), and finally by inserting acquired data through sensors or simulations in the database (User C).

project phase to contextualize the web platform.

A total workflow is depicted in Figure 4.3 to synthesize and visualize the decision process, representing all possible user interactions. A practical rule indicates that the more straightforward the work flow is the more efficient and easy to maintain is the code, consequently better is the user experience and the adaptability for further applications. In addition from the main work flow represented in the Figure 4.3, this chapter will provide a series of workflows diagrams from different parts of the process, linking the elements with the main view features to ease the reader understanding about the process.

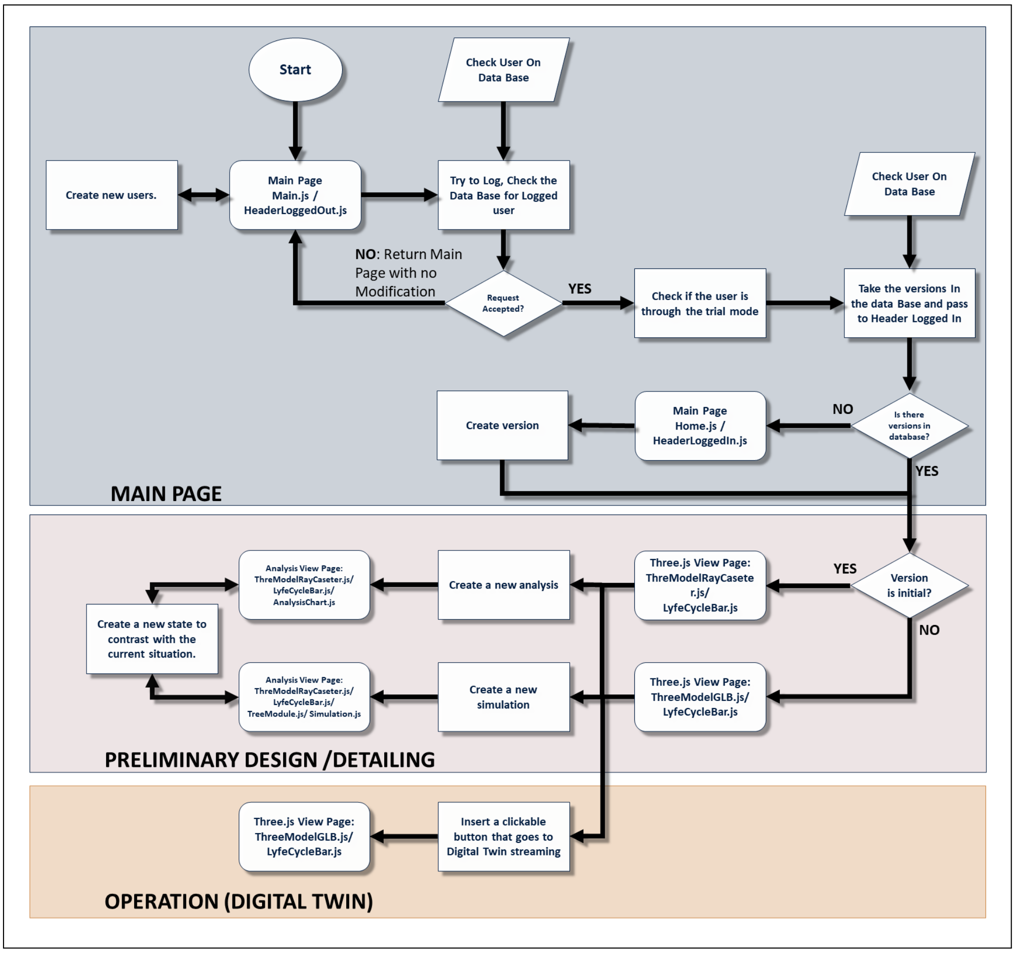

Start
Check User On Data Base
Create new users.
Main Page Main.js / HeaderLoggedOut.js
Try to Log, Check the Data Base for Logged user
Check User On Data Base
NO: Return Main Page with no Modification
Request Accepted?
YES
Check if the user is through the trial mode
Take the versions In the data Base and pass to Header Logged In
Create version
Main Page Home.js / HeaderLoggedIn.js
NO
Is there versions in database?
YES
MAIN PAGE
Analysis View Page: ThreModelRayCaseter.js/ LyfeCycleBar.js/ AnalysisChart.js
Create a new analysis
Three.js View Page: ThreModelRayCaseter.js/ LyfeCycleBar.js
YES
Version is initial?
Create a new state to contrast with the current situation.
NO
Analysis View Page: ThreModelRayCaseter.js/ LyfeCycleBar.js/ TreeModule.js/ Simulation.js
Create a new simulation
Three.js View Page: ThreeModelGLB.js/ LyfeCycleBar.js
PRELIMINARY DESIGN /DETAILING
Three.js View Page: ThreeModelGLB.js/ LyfeCycleBar.js
Insert a clickable button that goes to Digital Twin streaming
OPERATION (DIGITAL TWIN)

Figure 4.3: Total flow chart of the web platform.

4.3.1 Header and life cycle bar

The header in the top portion and life cycle bar in the bottom of the web platform are the command interfaces the user can interact that will change the features in the window, providing ways to change the analyses and ship types displayed. Each one of the possible command buttons appear in the Figure 4.4 with a brief explanation about their functionalities.

The Figure 4.5 depicts all possible styles the header can take, the appearance is linked with user phase of the project analysed. The first version displayed on the top is the one from the login page, in that case the header will wait for a username and a password to sign in or a command from the user to the open area of the web platform. Once the user is logged into the web platform the header will be changed in one of the three following possibilities. The "No ship version" style as suggested appears when the user has no types to show, in this case the drop down button and the show 3D drawing buttons are suppressed since it is impossible to apply those functions without any ship version in the database. The headers for detailing and initial design are similar, containing basically the same buttons set, however, there is a small notation inside the Show 3D button indicating the phase the analysed version is currently in.

In additional to the header page, the life cycle bar highlighted in Figure 4.6 is another source for commands. The bar has more relation with the analyse/simulations displayed than the header bar and have their buttons suppressed if they are not suitable for the study case. The Table 4.1 shows all the figures that can be displayed for the life cycle bar, showing the occasions in which they are suppressed.

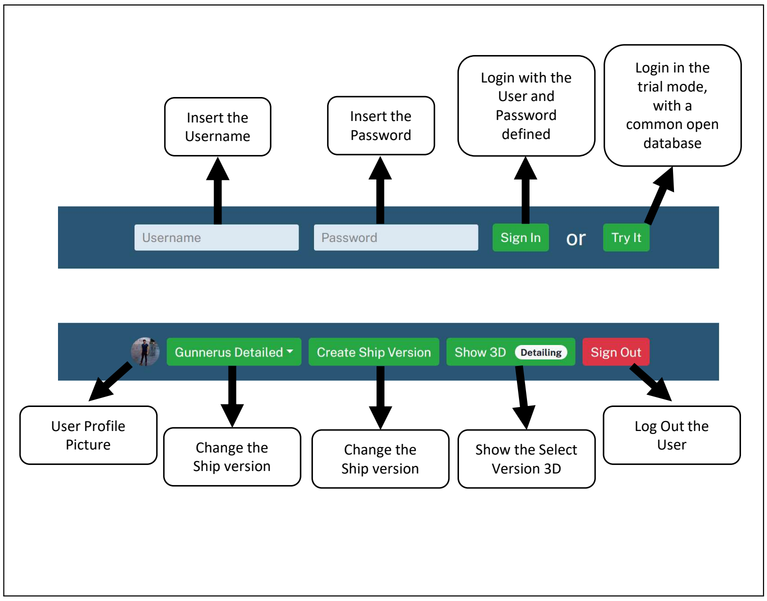


Figure 4.4: Header explanation features. Top header is for logged out users and bottom for logged in users.

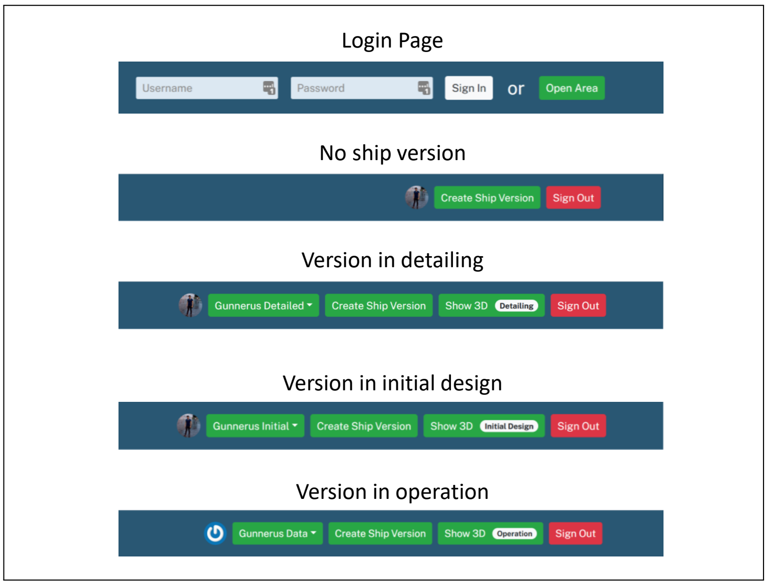


Figure 4.5: Header possible styles. The visualization format change according to the user and ship version cycle phase.

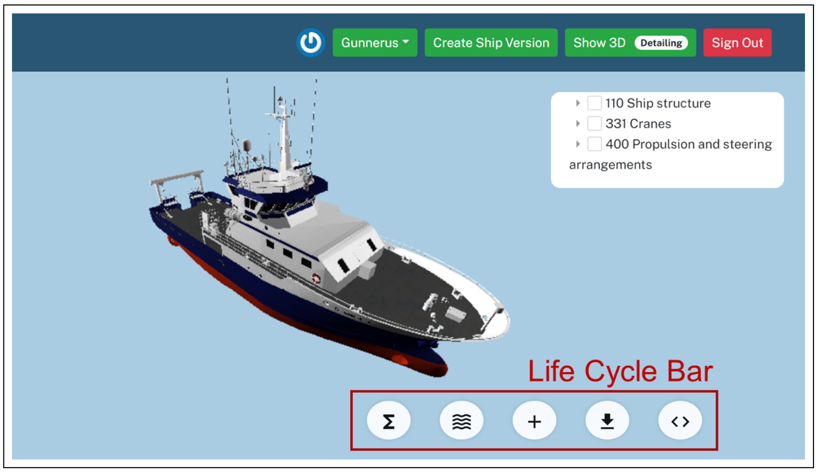
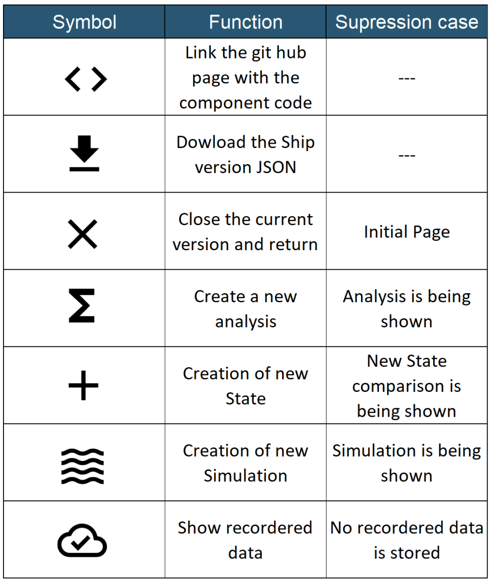


Figure 4.6: Life cycle bar style for the analysed. The figures are licensed according to

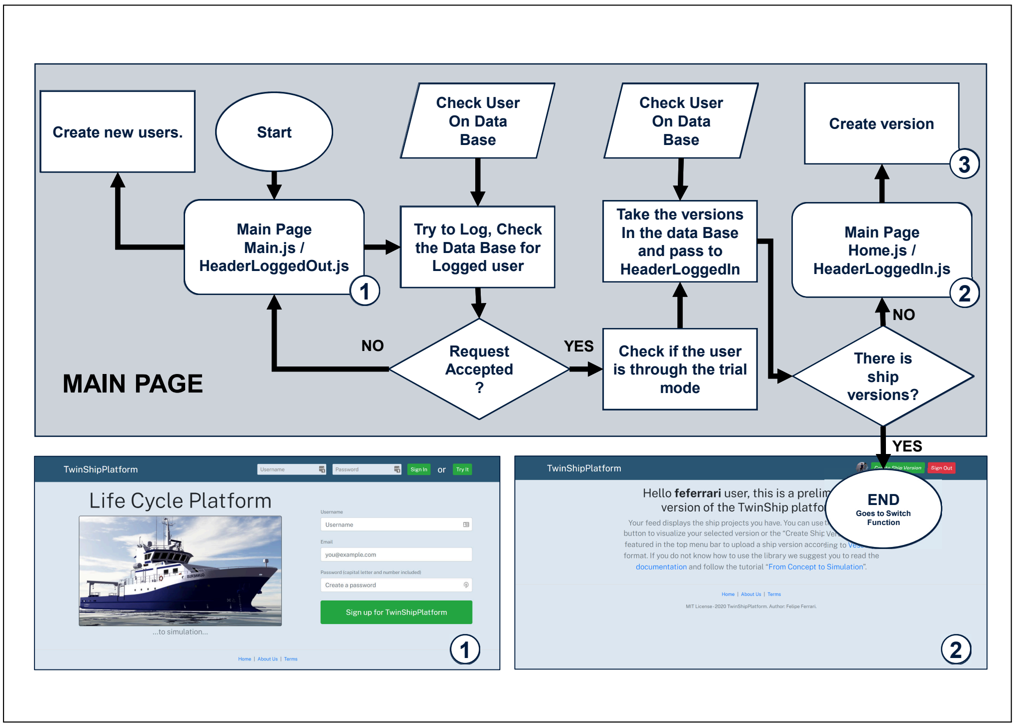


Symbol	Function	Supression case
< >	Link the git hub page with the component code	---
⬇	Dowload the Ship version JSON	---
✕	Close the current version and return	Initial Page
Σ	Create a new analysis	Analysis is being shown
+	Creation of new State	New State comparison is being shown
≋	Creation of new Simulation	Simulation is being shown
☁✓	Show recordered data	No recordered data is stored

Table 4.1: Icons used in life cycle bar. The Table provides a brief explanation about each icon command and the occasion it is suppressed. All the icons are licensed in a creative common version and are accessible in `https://fonts.google.com/icons`

4.3.2 Main Page

The main page workflow diagram, shown in Figure 4.7 with the number localization of both screenshots, is in most cases the first interaction the user has with the web platform. Basically in the main page is where the logic related to the login commands is handled, with user authentication and ship models creation and callback. On the left side represented by the icon number 1 is the feature of the home page for a non logged user with the purpose to show the services possibilities and allow the integration of new users with the web platform. The page presents an image transition slide showing the different phases the web platform can manage as a way to advertise for the broad usage application and a form which accepts the new users information to be stored in the database. The interface represented by the number 2 shows a home page with a logged interface for a user without ship models stored, containing a broad description about the learning tools and the directions where to start.



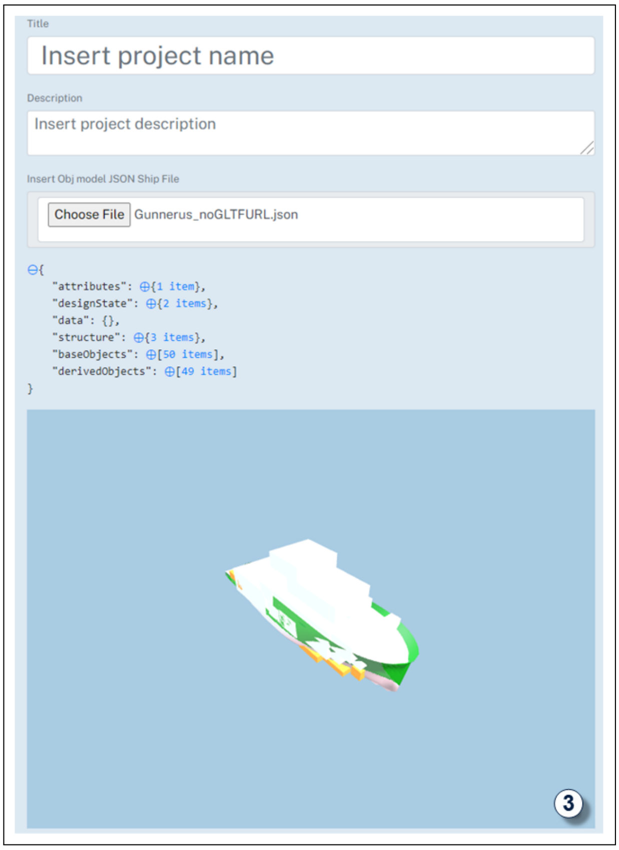
Hello **feferrari** user, this is a prelim
version of the TwinShip platfo

Your feed displays the ship projects you have. You can use t
button to visualize your selected version or the "Create Ship Ve
featured in the top menu bar to upload a ship version according to Ves
format. If you do not know how to use the library we suggest you to read the
documentation and follow the tutorial "From Concept to Simulation".

MIT License-2020 TwinShipPlatform. Author: Felipe Ferrari.

Figure 4.7: Idealization from the main page flow chart tasks.

On the Figure 4.8 is shows the create version page to insert a new ship version in the database with a correspondent title and description, this module is located by the index number three on the work-flow present in Figure 4.7. This page allows a verification of the project with a ship spaces preview and a JSON structure format according to the structure established by the vessel.js library. When there is no version in the database this page is the only one allowed to be accessed by the user as show in the header context in Figure 4.5



Title

Insert project name

Description

Insert project description

Insert Obj model JSON Ship File

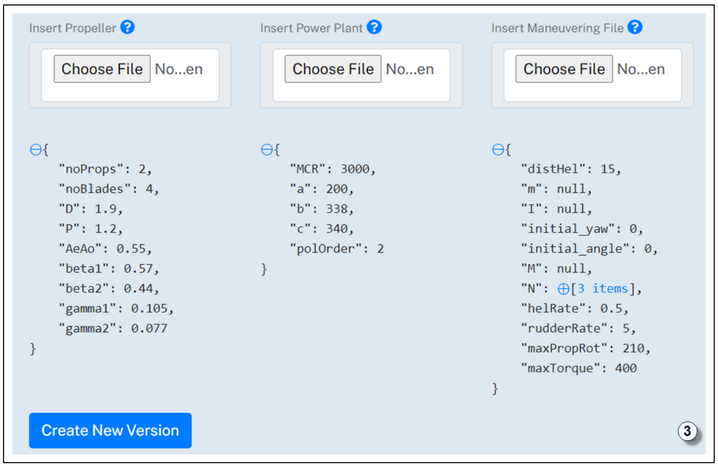
Choose File Gunnerus_noGLTFURL.json

```
⊖{
    "attributes": ⊕{1 item},
    "designState": ⊕{2 items},
    "data": {},
    "structure": ⊕{3 items},
    "baseObjects": ⊕[50 items],
    "derivedObjects": ⊕[49 items]
}
```

Figure 4.8: Creating version page. The number three in the bottom right corner represents the location of the page according to Figure 4.8.

On the same page, the JSON objects that compose the ship as the propeller, the power plant and the maneuvering related information can be inserted. In case any of the material are not informed a generic object is stored in the data base in its place to keep the compatibility with the other modules. The Figure 4.9 shows a example of the propeller, power plant and maneuvering information insertion in the page.

As can be noticed the credential verification is a part of the main page logic, on this stage one may wonder if any authentication process is relevant, since it is in the principles of the web platform to integrate new users as much as possible. In that ar-



```
{
    "noProps": 2,
    "noBlades": 4,
    "D": 1.9,
    "P": 1.2,
    "AeAo": 0.55,
    "beta1": 0.57,
    "beta2": 0.44,
    "gamma1": 0.105,
    "gamma2": 0.077
}
```

```
{
    "MCR": 3000,
    "a": 200,
    "b": 338,
    "c": 340,
    "polOrder": 2
}
```

```
{
    "distHel": 15,
    "m": null,
    "I": null,
    "initial_yaw": 0,
    "initial_angle": 0,
    "M": null,
    "N": [3 items],
    "helRate": 0.5,
    "rudderRate": 5,
    "maxPropRot": 210,
    "maxTorque": 400
}
```

Figure 4.9: Insertion of remaining objects. The number three in the bottom right corner represents the location of the page according to Figure 4.9.

gument, whatever type of authentication can limit the access to the whole potential of the web platform by turning private part of the database, potentially effecting the open source principles the project is based upon. To argue against this misconception it is crucial to have in mind the difference between privacy and privation, the possibility to have its own project and being the one with the permission to change the ship models is a privacy exercise that are embraced by the web platform fundamentals and in fact can incorporate more users by extending the web platform for the segment that have no intention to let their projects totally public. The authentication process must not be faced as a privation intention since all the features remain accessible in the open area, working as a public interface to share the projects. In addition the feature does not hurt the open source principles according to MIT license, since all the script that sustain the provided features remains available in the repository and are independent from the usage model.

4.3.3 Preliminary Design/Detailing

After the validation of the user and the ship version settled, the web platform will parse the information to the specific ship phase. The preliminary and detailing design follows a similar order containing first a visualization and then an analysis or simulation according to the user's command. The Figure 4.10 shows the work flow for a preliminary design and detailing project.

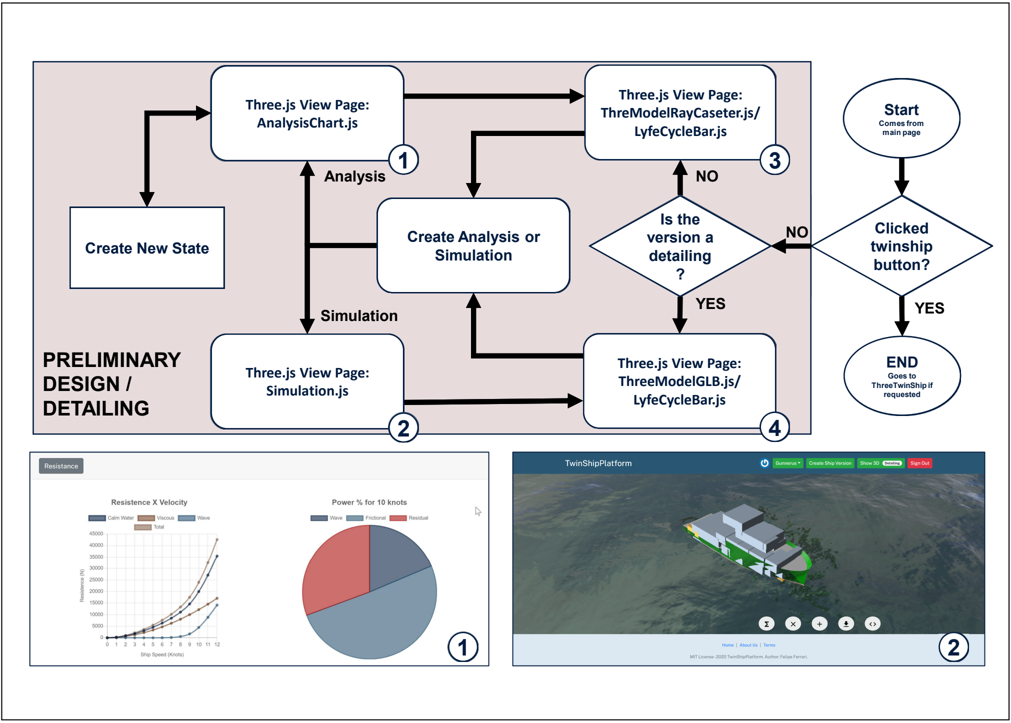
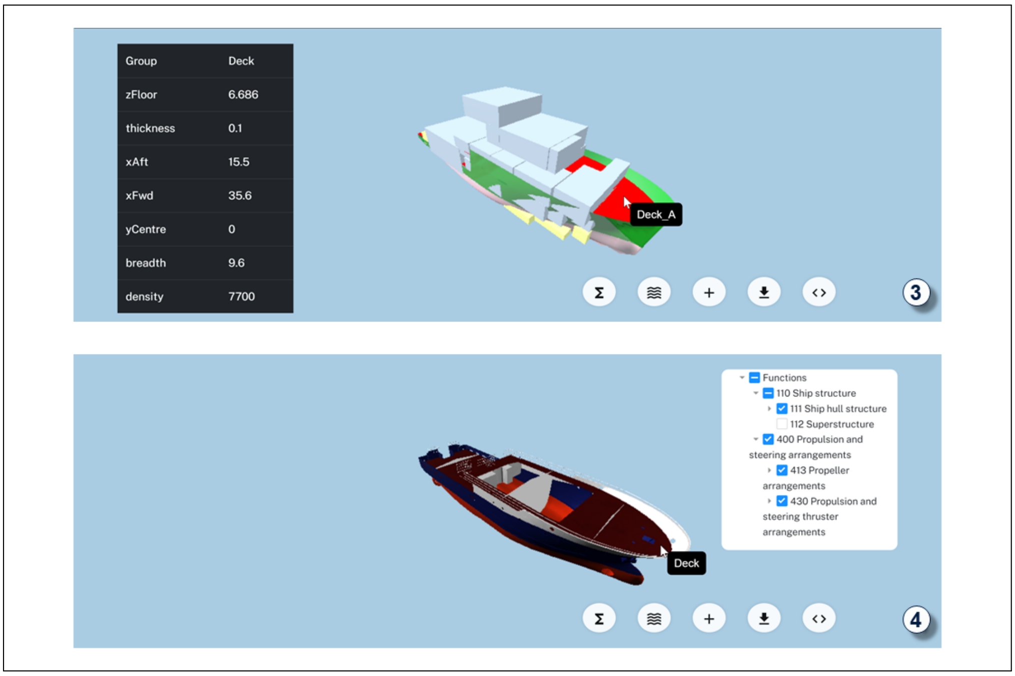

Three.js View Page: AnalysisChart.js
1
Three.js View Page: ThreModelRayCaseter.js/ LyfeCycleBar.js
3
Start
Comes from main page
Create New State
Analysis
NO
Create Analysis or Simulation
Is the version a detailing ?
NO
Clicked twinship button?
YES
Simulation
YES
PRELIMINARY DESIGN / DETAILING
Three.js View Page: Simulation.js
2
Three.js View Page: ThreeModelGLB.js/ LyfeCycleBar.js
4
END
Goes to ThreeTwinShip if requested
Resistance
Resistence X Velocity
Calm Water
Viscous
Wave
Total
Ship Speed (Knots)
Resistance (N)
Power % for 10 knots
Wave
Frictional
Residual
1
TwinShipPlatform
Create Ship Version
Show 3D
Sign Out
2

Figure 4.10: Flow chart tasks from the design and detailing project page.

Indexed by the number 1 in the Figure 4.10 is the the analysis feature, containing the line and pie graphics for the Resistance analysis. On the right side is shown a simulation window represented by the index number 2. Both capability potentials will be further explored during the Chapter 5 when the studies cases will be presented.

The visualization page, represented by the index 3 and 4 in the Figure 4.10 have same purpose: to show the facilities presented in the ship. However, in the preliminary analysis the ship follows a simplified visualization such as the one proposed in Oliveira (2021a), closer to the PSD philosophy that aims to chart the involved spaces in the early phases of the project. In the detailing phase the visualization page shows a more accurate ship format, containing a defined shape and colors through a gLTF (GL transmission format) file. The detailed object is ordered based on the Vessel Information System (VIS) principle such as explained in Section 2.3.4, separating the elements hierarchically according to its function that can be toggled to show or hide elements, similar to the analyses presented and developed in (I. A. Fonseca et al., 2021). A contrast by the two visualization types is shown in the Figure 4.11.



Group	Deck
zFloor	6.686
thickness	0.1
xAft	15.5
xFwd	35.6
yCentre	0
breadth	9.6
density	7700

Deck_A
3
Functions
110 Ship structure
111 Ship hull structure
112 Superstructure
400 Propulsion and steering arrangements
413 Propeller arrangements
430 Propulsion and steering thruster arrangements
Deck
4

Figure 4.11: Comparison between the spaces visualization in initial design and detailing stage. The top image shows a space drafting for the spaces, with a initial definition for the location. The bottom image shows a detailing project containing a defined form ship segmented according to the VIS taxonomy.

4.3.4 Operation (Digital twin) flow chart

Finally, the last flow chart represents the tasks with operational data, this condition is the closer the web platform reaches in the representation of a digital twin that can only be achievable by improving the response rate to a level that is imperceptible for humans.

Basically when the command for stream the operational data is triggered, the three.js module with the information is loaded. The data is parsed to each time step and simulated according to the maneuvering model inside the Vessel.js library. Both the field data and the calculated value are streamed side by side to show the contrast between the real case and the estimation. The flowchart for the tasks are presented in the Figure 4.12.

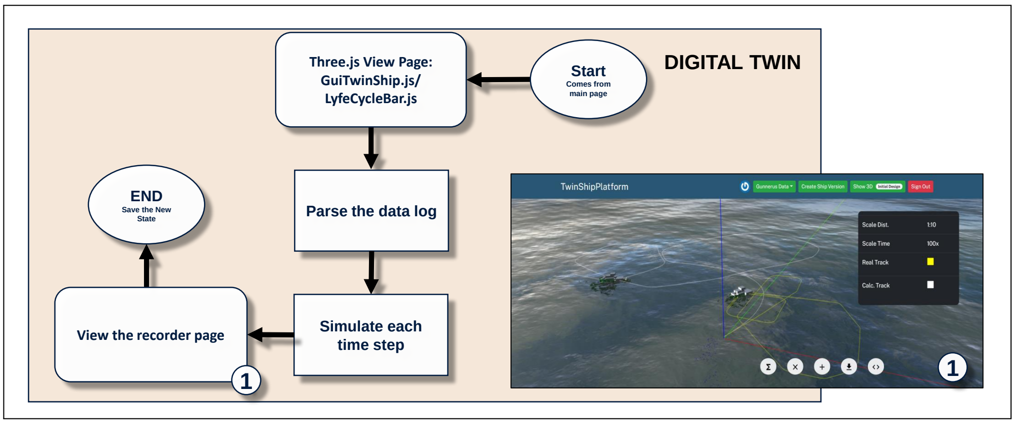


Figure 4.12: Flow chart in operation phase.

4.4 Implementation of mathematical models in Vessel.js

The current report expands the potential of the web platform by developing the maneuvering model with the concepts raised in Appendix C, this operation will make the Vessel.js library more suitable for a PSD approach and in this report will work also as a practical exemplification for library expansion and how to collaborate with the web platform as an User B actor represented in Figure 4.2. In this section we will dive deeper in the manoeuvring model integration and what are the main conventions defined.

4.4.1 Dynamic Methods

By definition the systems that change its states through time, for example the position and velocities, are classified as dynamical in opposition to the ones that remains with constant state values. To handle dynamical systems the Vessel.js uses the webGl renderer method (https://threejs.org/docs, accessed in 2021-04-04) provided by the Three.js library, which iterates the functions inside a loop with a variable frequency dependent of the available computational power in face of the simulation complexity. In general the simulation returns a satisfactory user experience when the rendering frequency is between $50 \sim 60\ Hz$.

The manoeuvring simulation developed in this report uses several methods ordered sequentially to solve numerically the ship displacement. The Figure 6.1 represents those different tasks, the purpose and the output for each function. In summary the script is settled to read the user command for propulsion and helix rotation, then it calculates the resistance at the present velocity that will be balanced with the propeller thrust to give the resulting force for approximating the displacement of the ship.

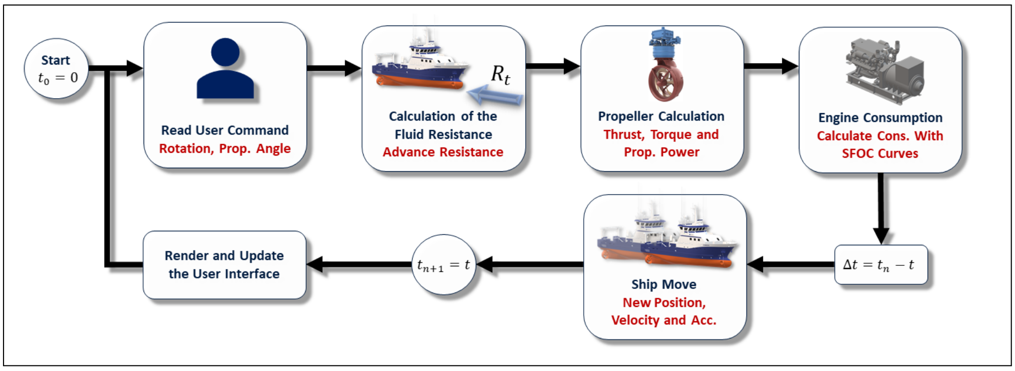

Start
$t_0 = 0$
Read User Command
Rotation, Prop. Angle
R_t
Calculation of the Fluid Resistance
Advance Resistance
Propeller Calculation
Thrust, Torque and Prop. Power
Engine Consumption
Calculate Cons. With SFOC Curves
$\Delta t = t_n - t$
Ship Move
New Position, Velocity and Acc.
$t_{n+1} = t$
Render and Update the User Interface

Figure 4.13: Render sequential methodology. The blue text is a description of the method purpose and the red text is the output received from the task. Ship drawing offered by NTNU, propeller drawing from Kongsberg Website (https://www.kongsberg.com/maritime/products/, accessed in 2021-04-04).

The three mathematical model used inside the loop; the propeller, the hull resistance, and the dynamical movement are dependent on each other. In that sense each change in one system effects the other method output, suggesting that an interactive process could be applied to find a more reliable solution than the sequential form proposed. This alternative, however, has as drawback the increment of required computational power to process the information to keep the simulation time step imperceptible for the user, therefore, the sequential option was chosen as alternative for this project.

4.4.2 Integration of new models into Vessel.js

To integrate new modules inside the Vessel.js it is indispensable to understand the structure of the library. First, the library uses an object oriented approach concept similar to the used in this web platform and explained in Subsection 2.4.4. However, in the current version the library uses a concept previews to ES6 notation of `prototype` instead of the syntactic sugar form of `class` to structure the inheritance.

We will present generically the steps to insert new models in the vessel.js library using our `Manoeuvring` model script as background, explaining the function of each step. For simplification, some of the contents of the code were suppressed, but the full code can be accessed in the main repository (`https://github.com/ferrari212/Platform-Twin-Ship`, accessed at 2021/06/14).The first step is to create a new class function which will inherit from the `StateModule` main class, this first part will guarantee all the states are linked and the speed and draft values are set according to the user definition or calculated parameters. The code inside the function works similar to the `constructor()` part explained in Subsection 2.4.4 and the remaining inputs can be chained to the class using the notation `this`. The `output` and `cacheDependence` mark the dependencies and the return module has, for whatever modification in the `cacheDependence` the memoized functions explained further will be triggered and store new values.

```
function Manoeuvring(ship, states, hullResistance,
   propellerInteraction, fuelConsumption, manoeuvring, rho = 1025) {

   StateModule.call(this, ship, states);

  if (typeof this.states.discrete.FloatingCondition === "undefined")
     {
    this.setDraft();
  }

  if (typeof this.states.discrete.Speed === "undefined") {
    this.setSpeed();
  }

   // Store the previews models inside the Manoeuvring model
  this.hullRes = hullResistance
  this.propellerInteraction = propellerInteraction;
  this.fuelConsumption = fuelConsumption
  this.powerPlant = fuelConsumption.powerPlant
  this.manoeuvring = manoeuvring;

  // Handle here the constructor calculations

  // Write down the output and caches
  this.output = ["hydroCoeff", "dn"];

  this.cacheDependence = ["PropellerInteraction", "FloatingCondition"
      ];
  this.cache = {};

}

Manoeuvring.prototype = Object.create(StateModule.prototype);
```

The next steps consists of creating the functions to be used in the module. Here the use of `Object.assign()` method will match the existing functions to our new model class. The assign of new functions is optional, and depends of the context the class is used for, the assign of constructor however is mandatory to attach the created function:

```
Object.assign(Manoeuvring.prototype, {
  constructor: Manoeuvring,
  getPropResult: function (n) {
    if (n === 0) return {Fp: 0, Pp: 0, cons: 0};

    // Calculate the total force from the propeller (Fp), the total
        power used Pp and the consumption rate cons here.

    return {Fp, Pp, cons};
  }
  // The other calculations are set here
});
```

The final steps is to set the heavy parameters calculation functions into the memoized function. The memoization is a technique for improving calculation performance by storing values that are processing expensive and used several times in the routine. The Figure 4.14 shows a flowchart containing the decision route for the memoizing function. In the maneuvering example, the values to be stored are the heavy processing values from the hydrodynamic calculations. The following listing gives an exemplification for those values:

```
Object.defineProperties(Manoeuvring.prototype, {
    hydroCoeff: StateModule.prototype.memoized(function() {

        // The heavy demanding calculation is done here

        return { Yvacc, Yracc, Nvacc, Nracc, Yvdn, Yrdn, Nvdn, Nrdn }

    }, "hydroCoeff"),
    // Other state modules are inserted here
});
```

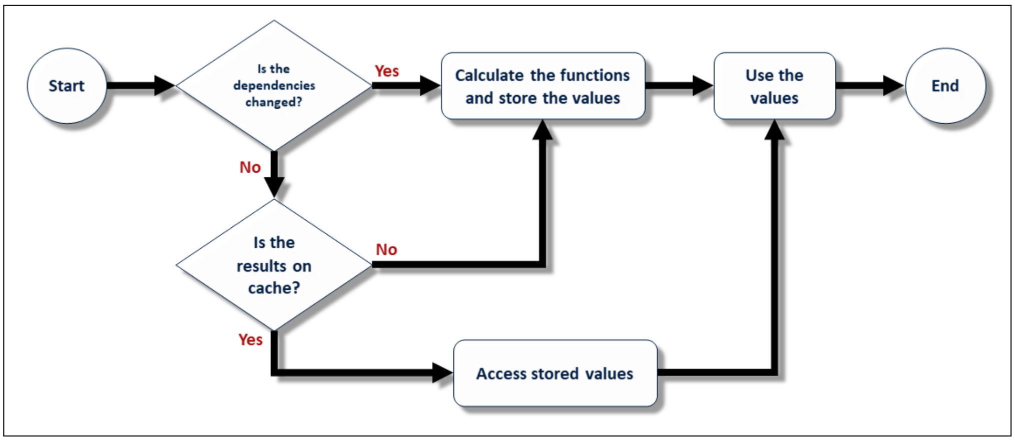


Figure 4.14: Memoizing function flow chart.

Those three steps in Figure 4.14 describes the guidelines for Vessel.js library expansion. In addition to the Maneuvering model which handle the assignment of the components and calculate the main information, a complementary class `ManoeuvringMovement` written in the ES6 notation for classes was constructed to handle derivative calculations solutions using the `numeric.js` open source library for matricial calculation and ordinary differential equation approximation. The code shown bellow is a simplification of `ManoeuvringMovement` class highlighting the main methods and noting the purpose behind each function. The `ManoeuvringMovement` class full content is accessible in (`https://github.com/shiplab/vesseljs/blob/dev/examples/snippets/ManoeuvringMovement.js`, accessed at 2021/06/19).

```
class ManoeuvringMovement {
  constructor(manoeuvring) {
  // Check for the numeric function existence
   if (typeof numeric !== "function") {
     console.error("Manoeuvring requires the numeric.js library.")
    }

    // sign up the main information
    this.mvr = manoeuvring;
    this.manoeuvring = manoeuvring.manoeuvring;
    this.states = manoeuvring.states;
    this.getPropResult = manoeuvring.getPropResult;
    this.dt = 0;

    }

   setMatrixes (F = [0, 0, 0], yaw = 0) {
   // Set the A and B matrices inside the function
    }

   parseA (R, M) {
     // Calculate the matrices A
     // Receives the matrices R for rotation of coordinates
     // and M for the massess
     return A
    }

   parseB (INVMF) {
     // Calculate the matrices B
     return [0, 0, 0, INVMF[0], INVMF[1], INVMF[2]]
    }

   parseR (yaw) {
     // Calculate the coordinate changing
     // matrix R
  }

  getDerivatives (V = {u: 0, v:0, yaw_dot: 0}) {

    // Calculates the derivative function X_dot

    return X_dot
  }

  getDisplacements (dt) {
   let self = this;

   // Parse matrix V

    let sol = numeric.dopri(0, dt, X, function (t,V) { return self.
        getDerivatives({u: X[3], v:X[4], yaw_dot: X[5]}) }, 1e-8,
        100).at(dt);

    // Get global coordinates variation (dx, dy, dyaw)
    // Get local velocity (du, dv, dyaw_dot)
    this.states.DX = {x: sol[0], y: sol[1], yaw: sol[2]}
    this.states.V = {u: sol[3], v: sol[4], yaw_dot: sol[5]}
    this.states.yaw += this.states.DX.yaw
  }
 }
});
```

4.5 Ship models for the study case

For validation and contextualization purpose three ship models were chosen as study case to apply the capabilities of the web platform: a squared barge from Í. A. Fonseca (2016), a PSV designs adapted from H. M. Gaspar (2016) and the Gunnerus university ship with the detailed drawings provided by NTNU.

The barge vessel is used as a test case for geometric calculations. Thanks to its rectangular geometry the dimensional calculations are relatively easy to predict, highlighting non expected results. The Figure 4.15 shows the lateral and top view of the squared geometry with its correspondents tanks volume.

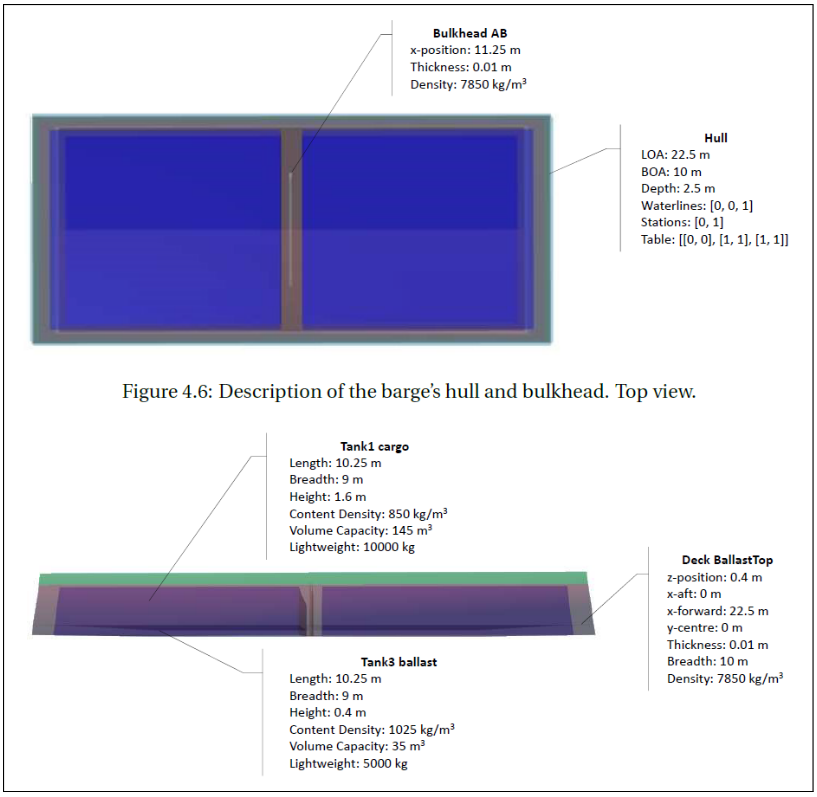


Figure 4.15: Geometries of the barge geometry (Í. A. Fonseca, 2016).

The second ship adopted is the PSV version created by H. M. Gaspar (2016), which contains 106 objects representing compartments and tanks, being the most extent type used before in Vessel.js library in terms of number of elements. This version was chosen due the relevance PSV has for the Norwegian economy and the better adherence to the estimation analysis restrictions used for vessel.js such as Holtrop (1984), Jensen et al. (2004), and Lee et al. (2003). The version of the PSV is shown in the Figure 4.16.

The last ship used as a study case is the Gunnerus ship that is owned by the NTNU university. The advantage about using this ship is the possibility to acquire real data

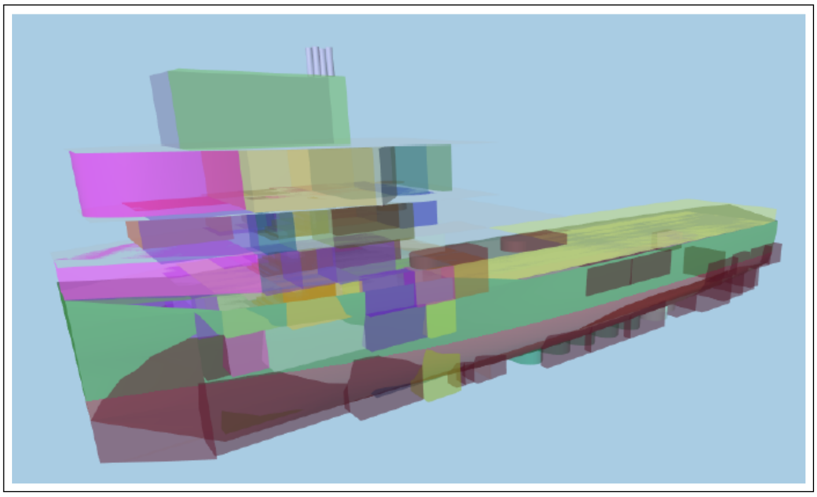


Figure 4.16: PSV geometry adopted (H. M. Gaspar, 2016).

from sensors installed in the ship and offered by the NTNU University, making feasible an initial application of digital twin. The NTNU university is also capable to provide a series of detailed drawings such as the three dimensional one shown in Figure 4.17 which improves the project quality and information.

Figure 4.17: Gunnerus ship three dimensional drawing offered by NTNU university.

The Table 4.2 summaries the relevant information for the analysis modes from each version case. As stated before, the usage of different geometries is also advised due the different restrictions the models implemented in Vessel.js library. In summary, the barge geometry was used for hydrostatic analysis validation, the other geometries are used to show the other resistance and seakeeping application, without attaining meticulously in the implications of rules detachment. The Table 4.3 shows the attachment of Gunnerus and PSV geometries according to the different models adopted.

	Barge	Gunnerus	PSV
LOA (m)	22.50	36.25	82.00
BOA (m)	10.00	9.60	18.00
Depth (m)	2.50	6.60	8.00
Draft (m)	2.00	2.79	6.50
LWL (m)	22.50	34.90	80.00
Cb	1.00	0.63	0.68

Table 4.2: Version main dimension summary.

	Lee Restrictions - Maneuvering		
	Gunnerus	PSV	Status Legend
Cb: 0.55 - 0.87	**0.626**	**0.680**	**Passed**
Draft/LWL: 0.022 - 0.071	**0.080**	**0.081**	**Fail**
LWL/B: 5.0 - 8.8	**3.635**	**4.444**	
Cb(BOA/LWL): 0.075 - 0.166	**0.172**	**0.153**	

	Holltrop - Advance Resistance		
	Gunnerus	PSV	Status Legend
LWL/B: 3.9 - 15	**3.635**	**4.444**	**Passed**
B/Draft: 2.1 - 4	**3.445**	**2.769**	**Fail**
Cp: 0.55 - 0.85	**0.626**	**0.600**	

	Jensen - Response Operator		
	Gunnerus	PSV	Status Legend
B/Draft: 1 - 6	**3.445**	**2.769**	**Passed**
			Fail

Table 4.3: Version information summary. The red items are the ones that does not pass in the criteria and blue are the ones fitted to the criteria

On this stage is evident the advantage of using a web platform to enhance productivity. In fact, the creation of modules require a significant effort to be programmed, but once the platform functionalities are in place the evaluation of the ship models is easy to implement, manipulate and compared in a high scale without the extra overload for drawing models adapting, graphics creation, storing and sharing the results in a common location for all the team mates involved. This advantage is what made possible the evaluation of the three geometry types for different analysis, simulations and operations in a relatively short period.

Chapter 5

Case studies

On this chapter will be shown the web platform capabilities via the three interaction forms schematized in Figure 4.2. Here the objectives are to show in practical terms meaningful results the ship designers can have with the application of the web platform as a instrument for ship evaluation, hence the chapter does not focus in providing a guide on how to use the web platform. When necessary, the geometries defined in Section 4.5 will be used to expound the web platform potential.

The terms analysis, simulation, and digital twin are used in this chapter according to the definition of Erikstad (2017) discussed in details during the Subsection 2.2.3 ordered according to information degree. The front end domain will be used most to exemplify the analysis build in Vessel.js and assimilated in the web platform as PSD application, while the back end domain will be used to show the simulations integration with the accommodation of Vessel.js mathematical models for maneuvering purpose and feed through the node.js library. In additional, the database will be used to show a simplified version for digital twin.

For clarifying purposes, the conceivable integration flow chart is reproduced over again in Figure 5.1 with a legend modification that helps the reader to placing itself in the conceivable operation of the web platform, linking them with the subsections of this chapter.

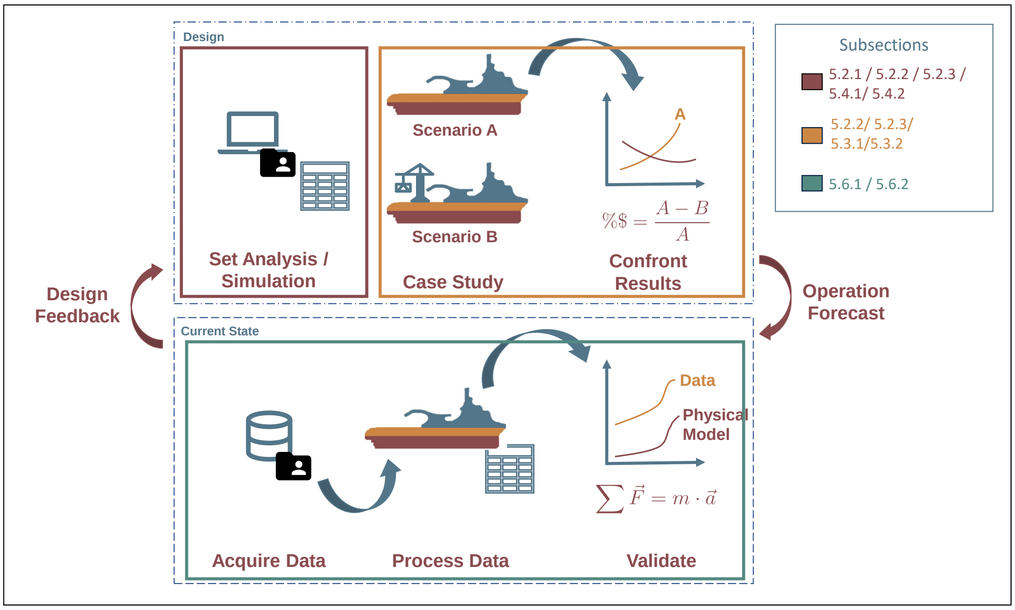


Figure 5.1: Conceivable mapping. Each color matches with a subsection of this chapter for clarification purpose

5.1 Full Stack Collaboration

As mentioned before in Subsection 2.4.1, the collaboration is an essential part in the open web platform function, hence this section is dedicated to discuss about the different collaboration ways the user can take.

The term Full Stack refers to the whole components used in modern web platforms: the front-end user interfaces, the back-end server and the database. In that sense, we name the term full stack collaboration as the capability to interact in the three domains presented, following the Figure 4.2 concept.

Starting from the front end collaboration which consists to features the user can exchange in the main URL view. In most cases, this form of participation comprehend the information the user can post into the web platform to receive in return the results from the simulations/visualizations, not requiring necessarily any specific coding skills, making this type of collaboration the most accessible and common way. For this specific web platform, as an open source project, it also permits the joint effort for interface improvement via GitHub repository request.

Notice the term collaboration comprehends the simple upload information in the front-end domain, since under the definition for web platform adopted, the data available is interpreted as a way for community strengthening and consequently the web platform extension.

The back end deals with the integration of several external packages and business logic, usually taken place in third parties devices as servers. In that sense, a collaboration via this path can be interpreted as a modification of the several API the web platform is fed. One of the key applications that the web platform is relied upon is the Vessel.js library code, which even those is processed in the front end domain, it is in the back end which the library is requested. A version modification can be easily updated through NPM command line and then integrated into the web platform avoiding outdated script usage.

NPM is an acronym for Node Package Manager, being the largest software registering in the world. The NPM consists on three components: a website, where users can check for packages, releases and documentation; a command line interface which permits the update of any package using Node.js with a single command; and a registry data base where most of the free software used in the world are.

The NPM registry can be found by node application with the version information in a `package-lock.json` accessing file, registries and requirement packages. The `package-lock.json` is generated automatically with the install of the npm and updated automatically according to the installation of new packages. To install new packages through npm it is just necessary to type the following command line in the Command Prompt, with the NPM registry name in place of `<package>` command:

```
npm install <package>
```

For more information regarding npm usage, please access the `https://www.npmjs.com/` web page. The Vessel.js version which is used under this project can be accessed in Oliveira (2021b) and it will be integrated in Vessel.js main repository after this project review.

The last type of collaboration is the direct database input of information from an external source. This collaboration comprehend the twin ship application since the sensors can act in behalf of the mentioned external source that input information into the web platform.

At the current stage, an advance type of digital twin which is able to acquire real time information, process and turn it back for decision making is not supported by the web platform for several technical reasons. For instance the web platform relies on several free services to not introduce new expenses to those involved in this project, this decision comes with the downside of having a slow response speed between the web platform and the database. Although this down side, the web platform itself could in theory be adapted to a new technology which uses a fast response, since the web platform was constructed in a modularized way, and in the current stage it can be considered as a initial form of twin ship as a digital representation for a real system.

5.2 Graphical Analysis

On the next subsection will be shown the graphical features the web platform provide in the front end, using different case studies in ship design as a background for displaying the interaction with the web platform results. This type of analysis is handled in most cases in the front end level of the web platform, with the user uploading content to receive back meaningful results.

The analysis process is the first level of information, which evaluates the physical phenoms by making a cause effect relation without computing the dynamic effects. This section will start by the hydrostatic features evaluation of the barge geometry, since her simple geometric shape makes it easier to validate the information. After that the two other geometries, the Gunnerus and PSV, will be used in the case studies for the resistance and response calculation.

Even those the analyses presented in the fallowing section are relatively simple in a ship design perspective they are easily adaptable to newer projects and expanded due the advantages the web platform usage gives. It is important to have in mind this project main goal is not specifically to improve the Vessel.js library and its models, but instead, to show how a web platform can be integrated into design and operation phases for improvement and validation, having this section to explicit the front end interfaces the user can rely on.

5.2.1 Hydrostatic Features Validation Case

For ship designers is always crucial to have a critical perspective with computational results, being extremely important the understanding about the magnitude degree and the physical phenoms involved in the solutions. In that context, this section will start from the hydrostatic assessment of Barge geometry as study case for confronting the results provided by the web platform using the Vessel.js library as well as presenting the hydrostatic analysis interfaces. As explained before in Subsection 3.2.2, the graphic modules were implemented to provide a summarized overview of the phenom studied, highlighting possible flaws for specialized professionals. Once any unpredictable behavior is found in the graphics, the root cause can be tracked back to the Vessel.js or the web platform code, giving the possibility for issue signalization in the open source repositories.

The barge, with dimensions described in the Table 4.2, has a relatively simple geometry, permitting a fast check of the solutions by confronting the numerical results with analytical formulas, so becoming an easy way for validating the front end tools. The web platform is conceived to calculate the hydrostatic values in ranges between $0.25\ m$ and the ship depth in steps of $0.25\ m$. The result can be conferred in Figure 5.2 with the values for the design draft noted in the graphic.

The Figure 5.2 shows expected graphic shapes for the barge geometry; in the left hand side the graphic shows a linear variation according to the draft, that is an expected result due the squared horizontal section of the vessel, the analytical solution in function of the draft h is shown in Equation 5.1. On the right side is shown the longitudinal center of floating and buoyancy, both presenting the expected result in the middle section of the ship $LCF = LCB = LWL/2 = 11.25\, m$.

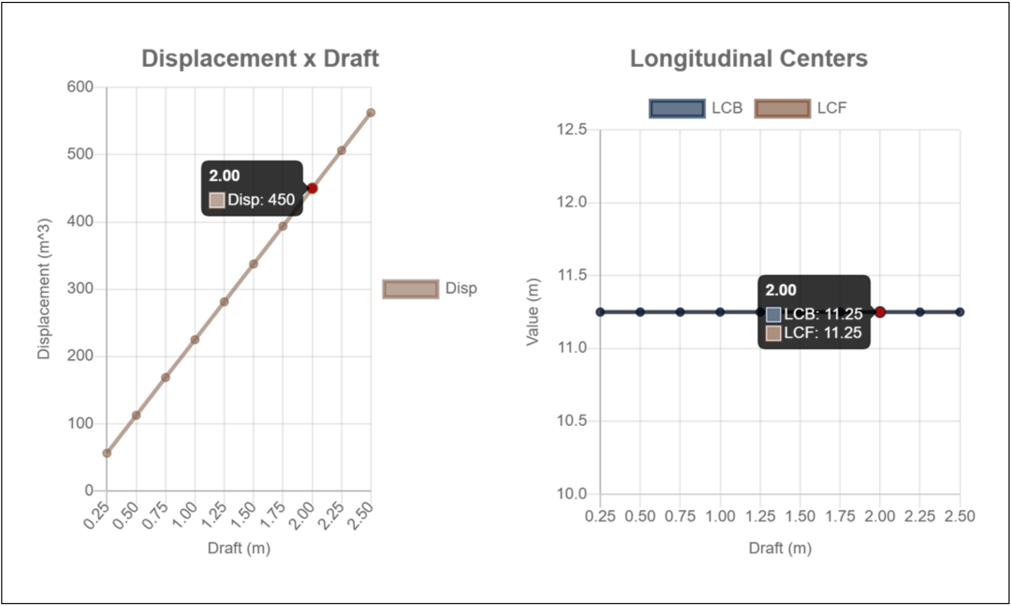


Figure 5.2: Displacement (left side) and Longitudinal Centers (right side) according to barge draft.

$$\Delta(h) = LWL \cdot BOA \cdot h \Rightarrow \Delta(2.00) = 450\ m^3 \tag{5.1}$$

The third graphic presented in the hydrostatic analysis are the vertical centers of buoyancy and metacentric shown in Figure 5.3. Over again the shapes of the curves behave as expected, following linear function for the KB and an inverse function for GM in relation to the draft h, according to the Equation 5.2:

$$\begin{cases} KB(h) = h/2 \\ GM_t(h) = \frac{I_t}{\Delta} = \frac{B^2}{12 \cdot h} \end{cases} \tag{5.2}$$

Finally, the web platform return the table summary of all information for the design draft. The Table 5.1 shows the example of the Barge in the two meters project draft.

Note that in the hydrostatic barge case there was an expected outcome, with the graphics providing shapes that are reasonable for the geometry. However, it does not mean we can trust whatever type of calculations result, specially for Vessel.js case

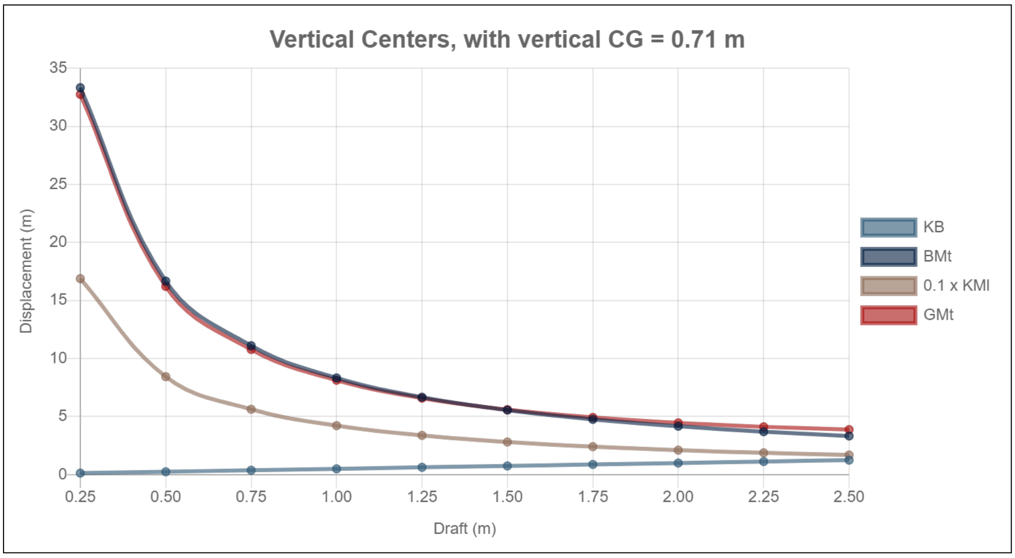
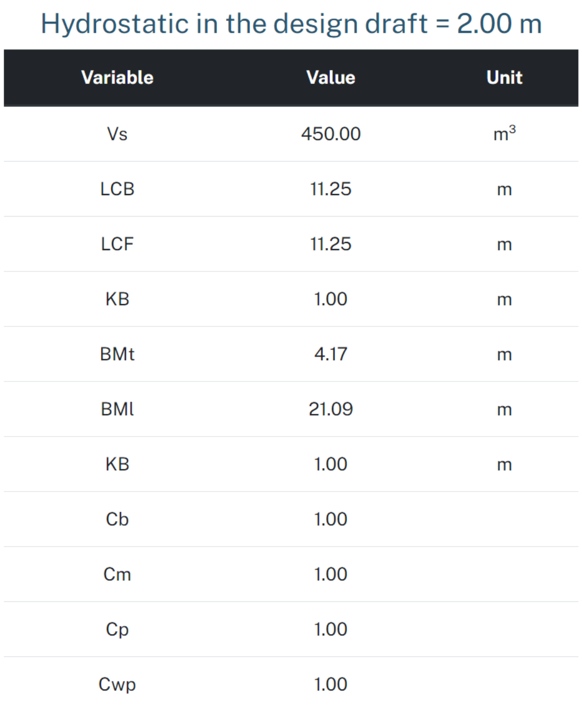


Figure 5.3: Buoyancy variables according to barge draft.



Hydrostatic in the design draft = 2.00 m

Variable	Value	Unit
Vs	450.00	m^3
LCB	11.25	m
LCF	11.25	m
KB	1.00	m
BMt	4.17	m
BMl	21.09	m
KB	1.00	m
Cb	1.00	
Cm	1.00	
Cp	1.00	
Cwp	1.00	

Table 5.1: Hydrostatic summary for the Barge vessel.

which is a library still in the beginning stages of development and is under constant evolution with high probability for numerical method errors. This must be a special concern for non regular shapes like bulbous bows vessels that are harder to be fitted under general solutions.

To give a practical example of mismatching results and how the front end collaboration can highlight methodological errors in the web platform and the supporting libraries, take the following case happened during the Gunnerus evaluation. The previews graphics were showing unrealistic values of coefficients of form and water line length for Gunnerus geometry. The root cause was tracked back and was verified the water line length calculation method considered a maximum global instead of the length in the draft studied, this error was passed without being noticed before because the previews projects using the Vessel.js library did not ever accounted for a bulbous bow as Gunnerus has. The graphic presented in the Figure 5.4 shows a contrast between the current version, corrected after the described study case, and the previews v0.0.1 for waterline length values in function of the draft applied for the Gunnerus geometry. The full calculation and plotting accessible in details by `https://observablehq.com/@ferrari212/comparison`, accessed in 2021-04-16.

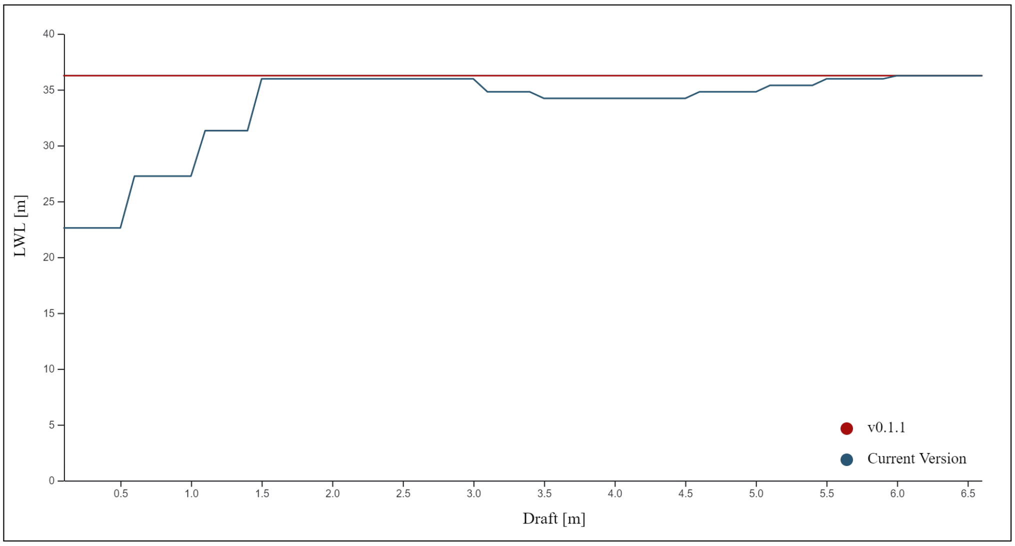


Figure 5.4: Comparison of LWL calculation result between the v0.1.1 and the current improved version.

This example shows how the use of graphics to organize complex data can have a positive impact in errors identification by linking established technical knowledge with concise information transparency charts provide. The waterline length has still an error in the order of the offset distance of the input table. A reduction of this value would be possible by applying an interpolation method to represent the longitudinal section of the ship. Although the method for water line length still presents an numerical error, the current version has close enough results for the current expectation stage. The accuracy of the methods and other types of improvements will be suggested later in the Section 6.3.

5.2.2 Resistance

Different from the previews Subsection 5.2.1 regarding the hydrostatic calculation, in which the searched information is dependent only in geometric parameters, the ship resistance is dependent of several and sometimes difficult to measure variables.

The most important factor in ship resistance is the the fluid resistance, which is hard to be assessed analytically for most scenarios due the non linearity of the Navier Stokes Equations that describes the phenom. The fluid forces can be divided in two main sources: viscosity and potential forces. The terms of viscosity, encompass the resistance due the friction in the flow layers and the variation of pressure due flow separation also known as residual resistance. The term of potential forces are linked to the wave making and relates to the energy spend in free surface perturbance.

This Subsection has as objective to show the features provided by the web platform in the resistance field using the Holtrop (1984) integrated to Vessel.js library. A visual graphics analysis will be held without attaining exhaustively to the applicability of Holtrop (1984) for the geometries exemplified or the previews implementation in Vessel.js library. We can start by exposing the Figure 5.5 with the resistance in function of ship velocity for Gunnerus and PSV ship.

The Figure 5.5 shows how the resistance behave, and what are the nature from each source. It is possible to notice, for example, a more demanding energy in wave

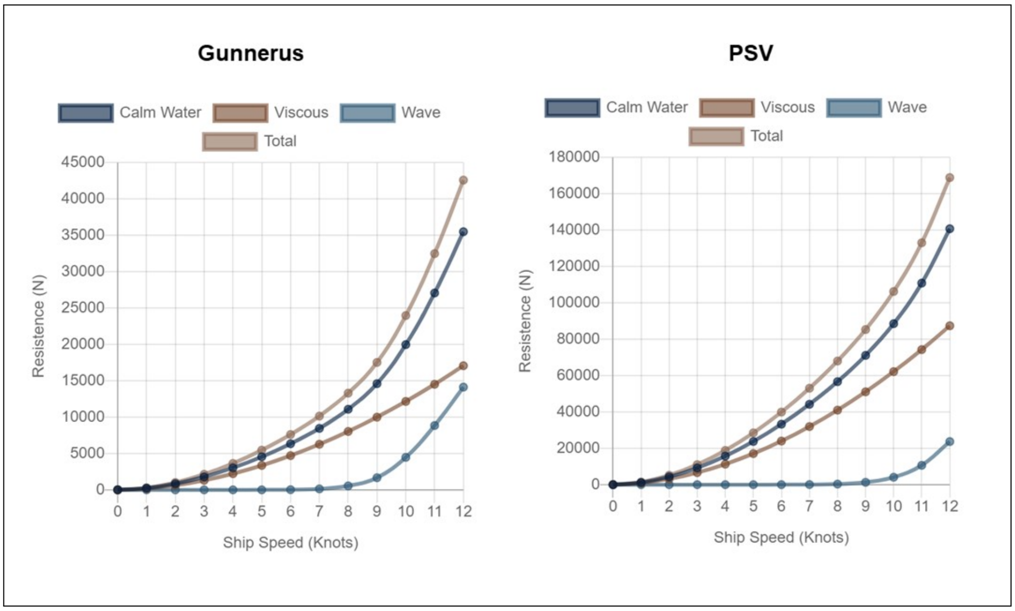


Figure 5.5: Resistance values in function of speed for the Gunnerus version and PSV.

generation for higher speeds in Gunnerus, while the PSV presents a more demand in the viscous source. This information could be used in ship designing phase to evaluate solutions that are better suitable for resistance reduction. To let this information more explicit, the resistance percentage were rearranged for the design speed, the pie chart shown in Figure 5.6 shows each type of source for the defined operation speed.

Finally, similar to the presented case in hydrostatic design, the resistance module presents the summary for the resistance according to the 5.2.

5.2.3 Response Operation

The third analysis that is possible to be held in the web platform is the response operation using the Jensen et al. (2004) approximations, already implemented in the Vessel.js library. The web platform plot the non dimensional responses and accelerations for different heading types from 90 to 180 degrees (where 90 degree represent side waves and 180 degrees represent head seas). To give an graphical exemplification, an example from the resulting for the PSV, are shown in the Figure 5.7.

As in the other examples, a table is plotted to show the significant results. For this

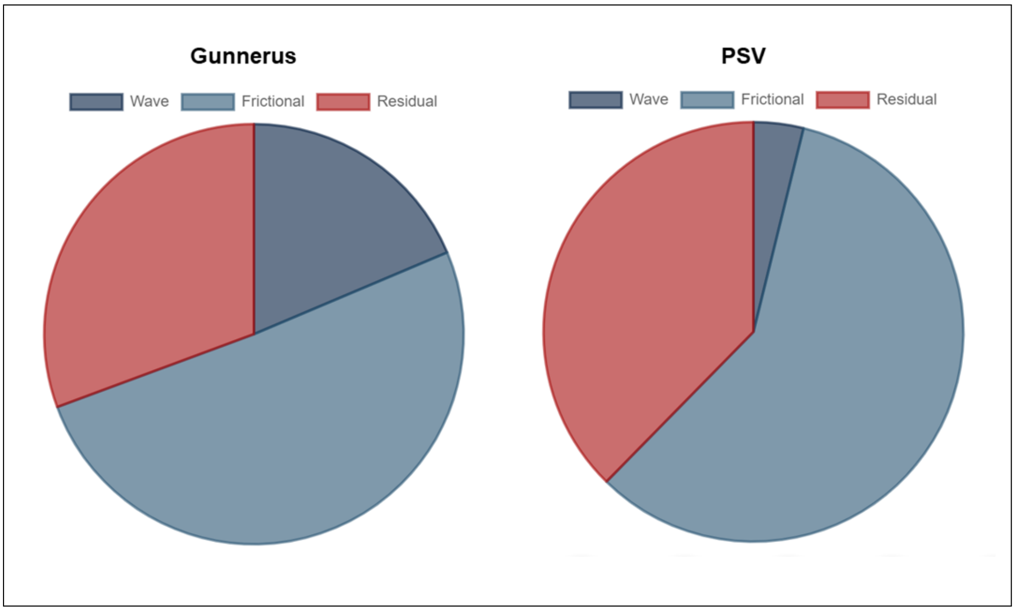
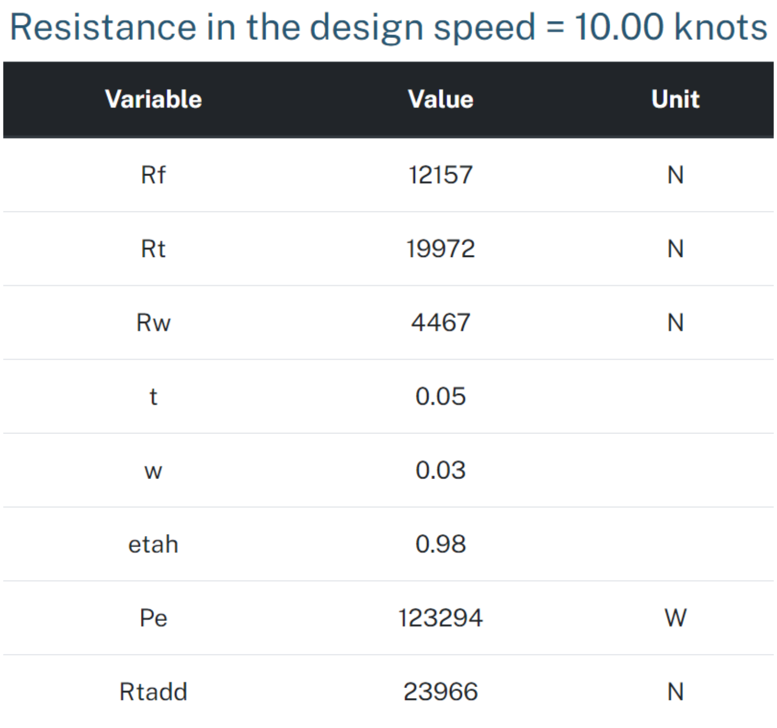


Figure 5.6: Pie graphic for the gunnerus version and PSV.



Resistance in the design speed = 10.00 knots

Variable	Value	Unit
Rf	12157	N
Rt	19972	N
Rw	4467	N
t	0.05	
w	0.03	
etah	0.98	
Pe	123294	W
Rtadd	23966	N

Table 5.2: Resistance for the Gunnerus web platform.

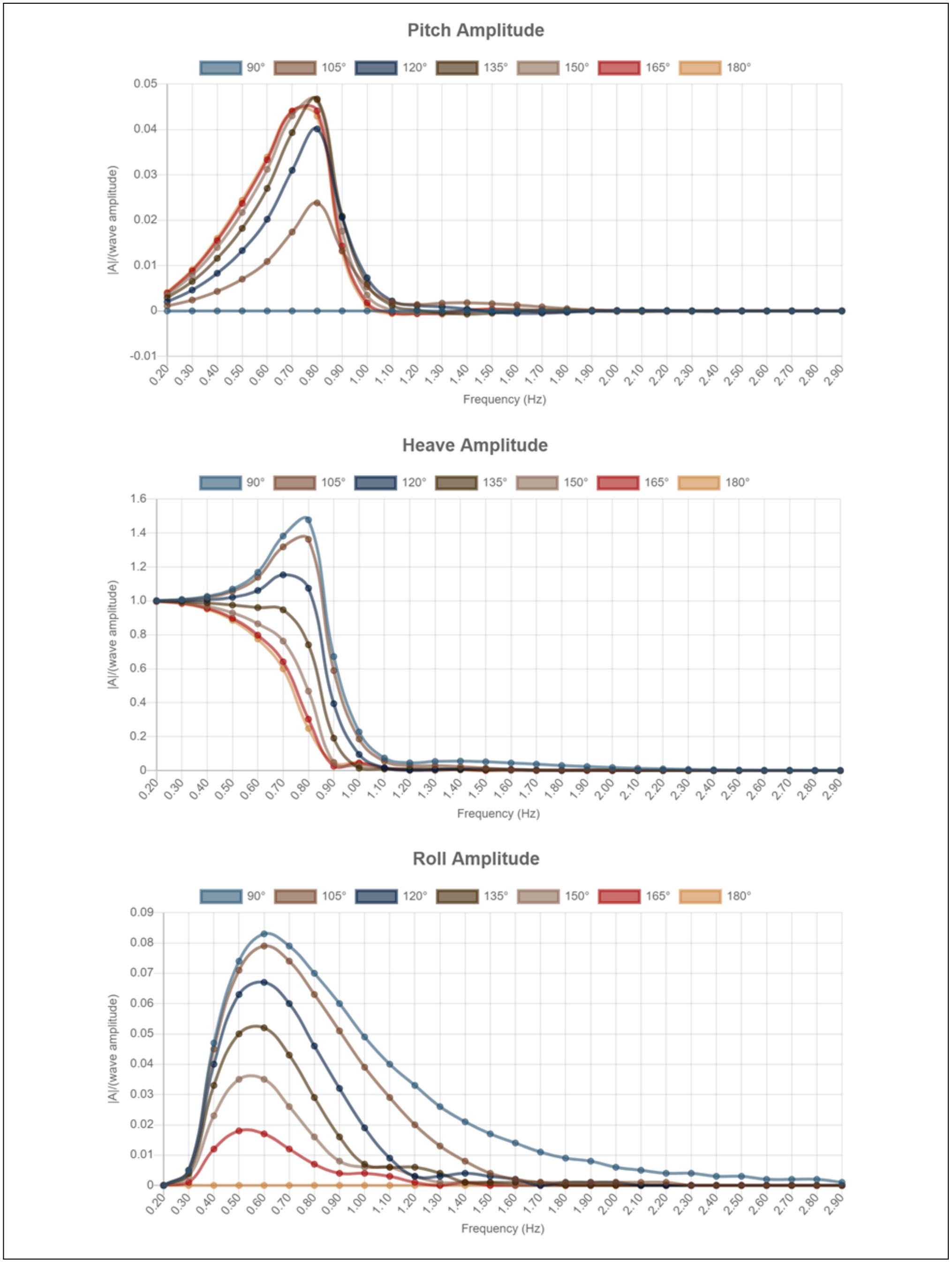
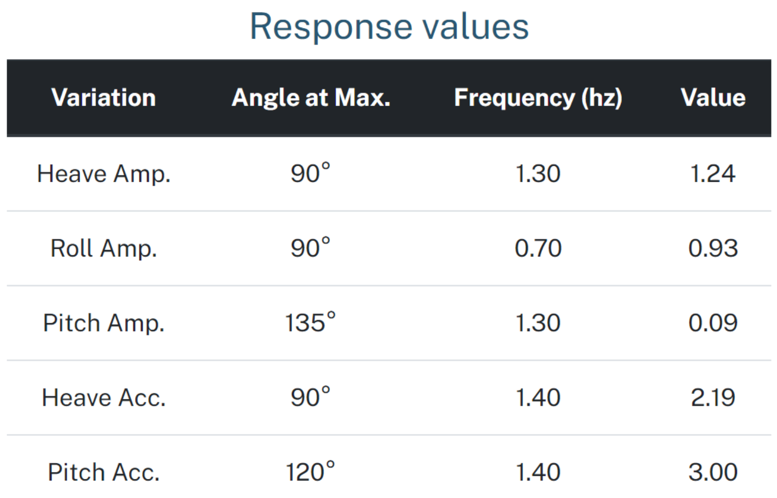


Figure 5.7: Response operations graphics.

case the maximum for each response are shown in Table 5.3.



Response values

Variation	Angle at Max.	Frequency (hz)	Value
Heave Amp.	90°	1.30	1.24
Roll Amp.	90°	0.70	0.93
Pitch Amp.	135°	1.30	0.09
Heave Acc.	90°	1.40	2.19
Pitch Acc.	120°	1.40	3.00

Table 5.3: Maximum amplitude for each degree of freedom angles according to the angle at maximum value, the frequency and the value for Gunnerus vessel.

5.3 Gunnerus Jumboization Study Case

Jumboisation is a procedure which involves the ship elongation by introduction of a middle body section to attend an increasing demand for the ship suited operation. Recently, Gunnerus university ship passed through this procedure responsible to increase the length overall of the ship in five meters. The modification involves before hand calculation of several factors, for example, the increasing in resistance, modification of the response operator curves and hydrostatic factors variation, having a special complexity degree by managing the trade off from distinct areas of the ship.

This section use this past scenario to bring a motivation for the web platform usage as a tool for efficient design and to give a context for comparison graphics situation, putting in contrast two ship versions. As discussed before in Subsection 2.2.1, in PSD it is ideal to run through the project loop with lowest friction as possible, with this concept in mind, the web platform was designed to give the ability for new users to create easily new scenarios for the specific ship versions, allowing a fast pretesting of different types of ship versions. Since the web platform is a single source of information linked to a specific database, chances of version mismatch are reduced in the process of evaluation different Jumboisation extension values.

Using the life cycle bar to press the "create new state" button identified in the Table 4.1, a slider form shown in Figure 5.8 appears to create a new circumstance for the current version. In the current web platform version the new state creation allows only main dimensions modification, however, nothing prevent that in newer version the web platform would be able to modify external variables as well, such as the wave pattern. Through the following Subsection 5.3.1 and Subsection 5.3.2 it will be presented the comparison interfaces for a variation over five meters increasing in the total length for Gunnerus as if a new Junbolisation would be under evaluation.

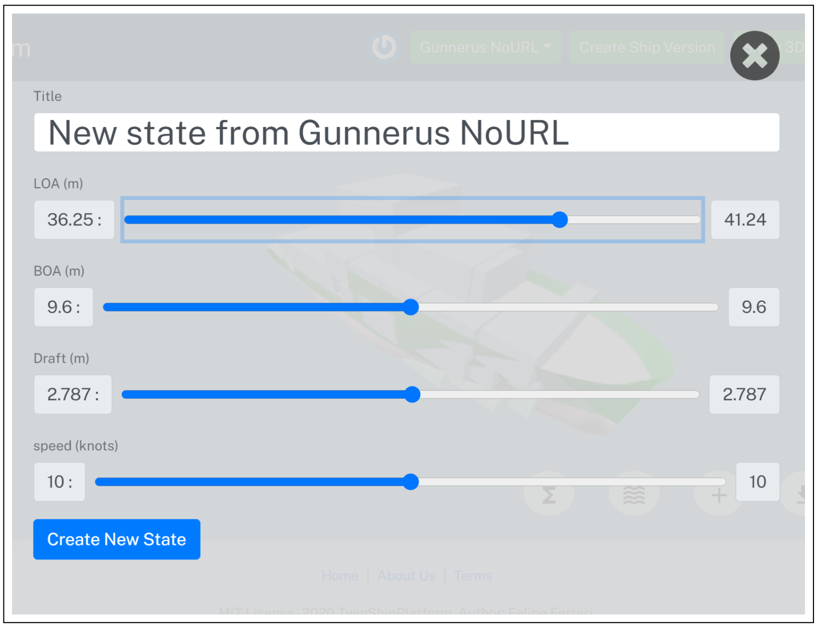

Title
New state from Gunnerus NoURL
LOA (m)
36.25 : 41.24
BOA (m)
9.6 : 9.6
Draft (m)
2.787 : 2.787
speed (knots)
10 : 10

Figure 5.8: New state page example with the Jumboisation value for the length over all.

5.3.1 Resistance testing - Jumboization study case for Gunnerus

The graphics following to be analysed were generated in a hypothetical scenario which the ship would increase its length over all again by five meters, explaining possible conclusion ship designer could rely on by graphic visualization. We can start first with the resistance analysis, the graphics in Figure 5.14 presented in the web platform shows the resistance behavior for the current and new state. On the figure left hand side is located the resistance variation according to ship speed, it can be notice it is not shown any expressive variation, at least in the perspective of the models assumed in vessel.js library, which favor the adoption of the longer ship due the capacity to transport more weight at same speed.

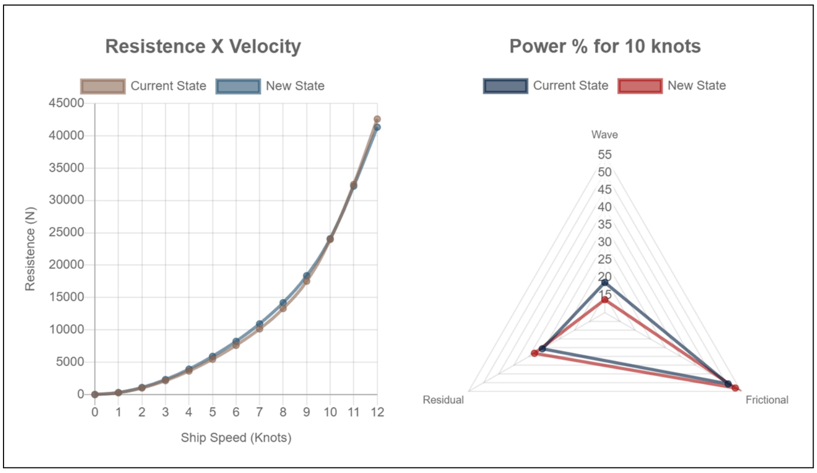
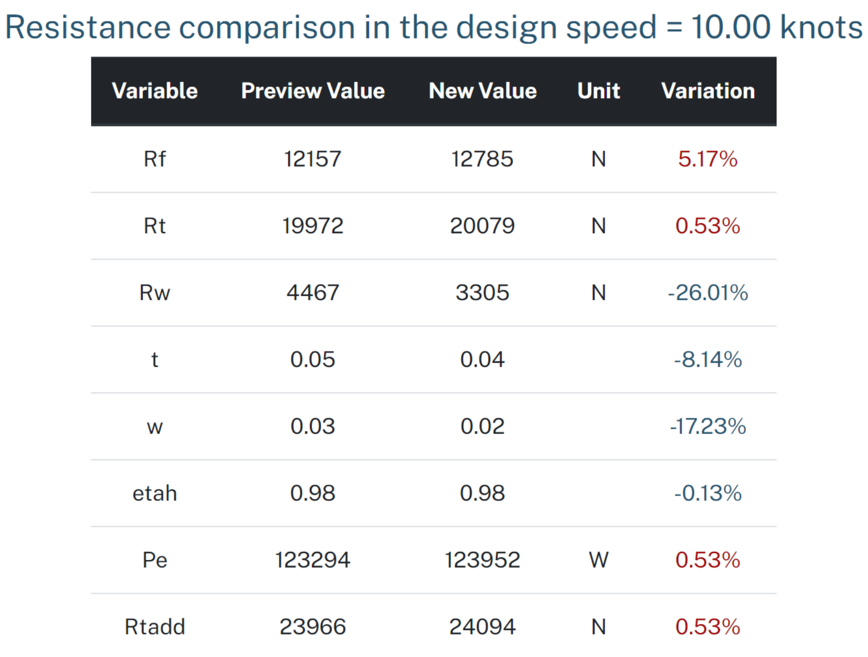


Figure 5.9: Resistance comparison between the length extended and current version ship.

It is possible to noticed that higher velocities tend to favor the adoption of the new state by presenting a slightly smaller resistance while for lower speeds the resistance favor the current state status, this phenomenon happens partially due the nature from each force. Despising the difference in displacements, at higher velocities the wave forces are predominant, benefiting longer ships with smaller Freud number. For other hand, in smaller speeds the viscous factors are stronger, benefiting the shortest ships which contain smaller Reynolds number. This can be evident in the right hand part of the Figure 5.14, where the new state presents an almost 5% less wave resistance participation in the total resistance for a project speed of 10 knots (a speed relatively high) if compared with the current state version.

A summary similar to the simple analysis calculation is present at the end of the analyse. The Table 5.4 presents the contrast between the current state and new state, notice that there is just a small variation in total resistance despite of the displacement increasing thanks to a expressive decreasing in 26% in wave resistance giving.



Resistance comparison in the design speed = 10.00 knots

Variable	Preview Value	New Value	Unit	Variation
Rf	12157	12785	N	5.17%
Rt	19972	20079	N	0.53%
Rw	4467	3305	N	-26.01%
t	0.05	0.04		-8.14%
w	0.03	0.02		-17.23%
etah	0.98	0.98		-0.13%
Pe	123294	123952	W	0.53%
Rtadd	23966	24094	N	0.53%

Table 5.4: Resistance for the Gunnerus web platform.

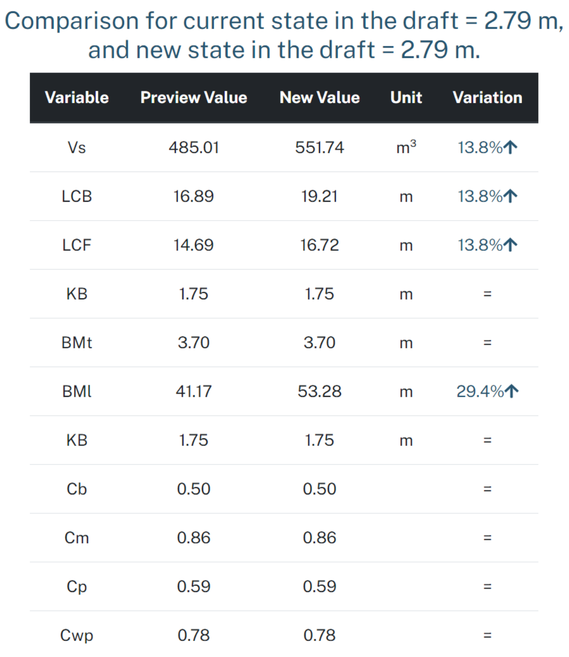
5.3.2 Response operation and hydrostatics comparison - Jumboization study case for Gunnerus

On the last subsection it was proved that at least for the perspective of the ship resistance using the Holtrop (1984) method there is an advantage in increasing the ship length by five meters. Although this information is positive, it is not sufficient to define if the change really would cause a global improvement, that is why it is necessary to take in consideration the different aspects of the ship as proposed by PSD procedure. In this section we are going to use the hydrostatics and seakeeping parameters to evaluate the Jumboization scenario as well, like conceived in Figure 4.1.

Table 5.5 summarizes the comparison outcome from the two ship versions. In the terms of hydrostatic, the most clear outcome is the increasing in the volume displacement with the increasing of the length, meaning in this new configuration the ship is able to carry more weight than before without a significant increasing in the energy demand as demonstrated in Subsection 5.3.1. The LCB and LCF are changed proportional to the increment in length which may effect the weight distribution and consequently the trim. The major difference is the increasing of the longitudinal metacentric distance, which in this case represented an almost 30% in gains that can be translated into a more stable ship trim angle.

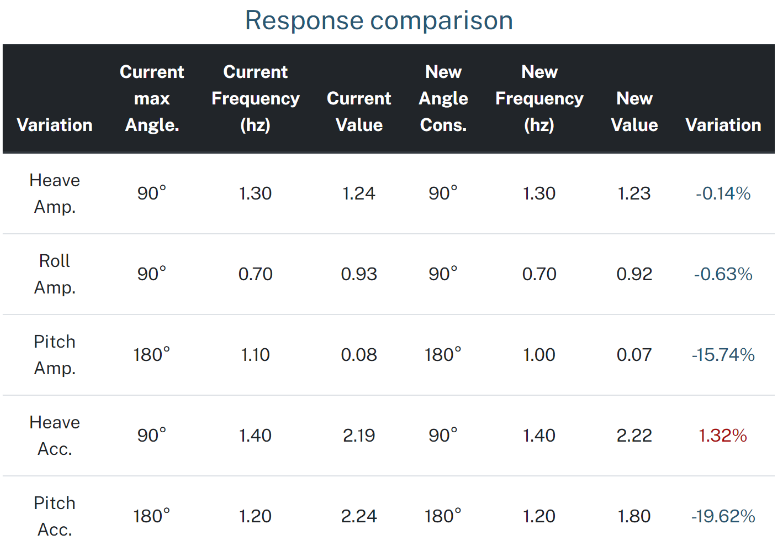
Under the seakeeping perspective, the biggest consequence in the response area for the metacentric variation is that the pitch response and acceleration become smaller as shown in the Figure 5.10, without effecting the roll behavior and with a small variation in terms of resonance frequencies in heave as shown in Figure 5.11. The values of maximum results can be compared in the Table 5.6, showing as expected a predominant response reduction with emphasis in the pitch variation.

Taking the studies raised in this Subsection and Subsection 5.3.1 we can conclude a further Jumboization of gunnerus could be beneficial for the ship performance, at least in terms of the resistance, weight carriage, and seakeeping according to the models adopted in Vessel.js library. Here is important to point out once again the advantage of the web platform usage in applying several methodologies in a single architecture, making this type of comparison, which involves the evaluation of two geometries in different perspectives, easier to be made.


Comparison for current state in the draft = 2.79 m, and new state in the draft = 2.79 m.

Variable	Preview Value	New Value	Unit	Variation
Vs	485.01	551.74	m^3	13.8%↑
LCB	16.89	19.21	m	13.8%↑
LCF	14.69	16.72	m	13.8%↑
KB	1.75	1.75	m	=
BMt	3.70	3.70	m	=
BMl	41.17	53.28	m	29.4%↑
KB	1.75	1.75	m	=
Cb	0.50	0.50		=
Cm	0.86	0.86		=
Cp	0.59	0.59		=
Cwp	0.78	0.78		=

Table 5.5: Hydrostatic comparison for the current and new version.


Response comparison

Variation	Current max Angle.	Current Frequency (hz)	Current Value	New Angle Cons.	New Frequency (hz)	New Value	Variation
Heave Amp.	90°	1.30	1.24	90°	1.30	1.23	-0.14%
Roll Amp.	90°	0.70	0.93	90°	0.70	0.92	-0.63%
Pitch Amp.	180°	1.10	0.08	180°	1.00	0.07	-15.74%
Heave Acc.	90°	1.40	2.19	90°	1.40	2.22	1.32%
Pitch Acc.	180°	1.20	2.24	180°	1.20	1.80	-19.62%

Table 5.6: Comparison for the current and new response operation.

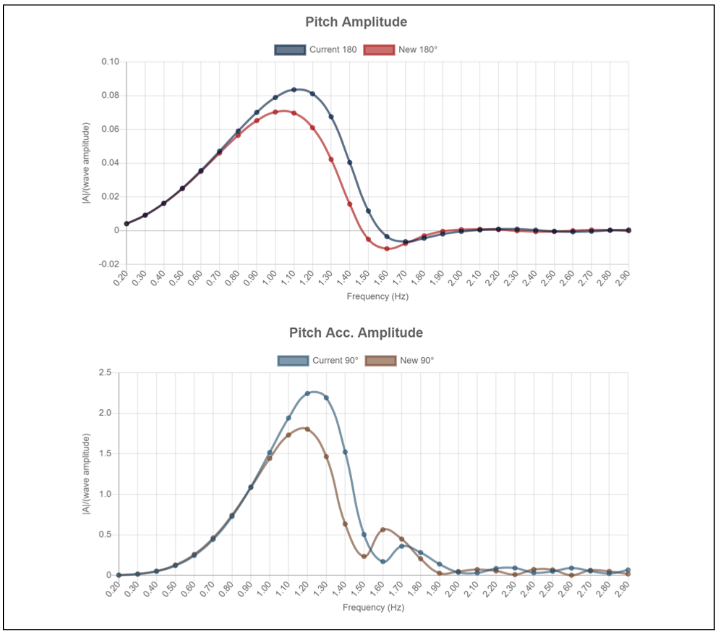


Figure 5.10: Response operation amplitudes for the current and Jumboisation versions.

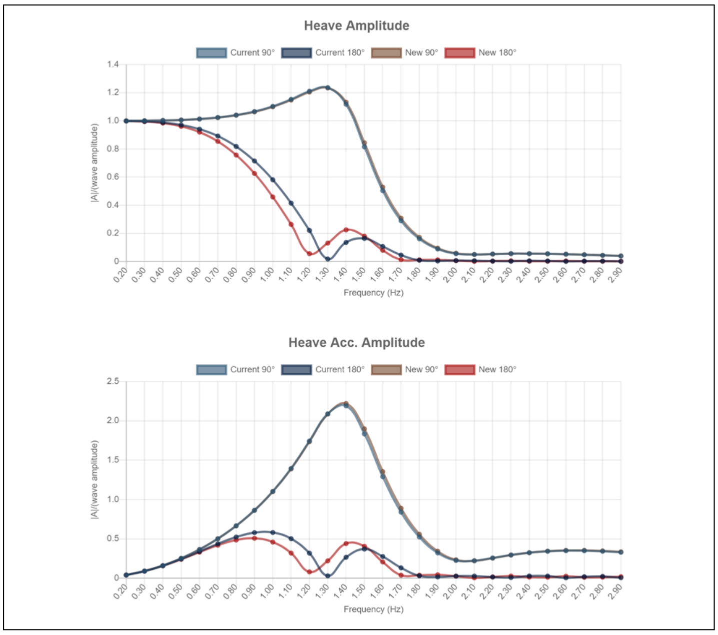


Figure 5.11: Response operation acceleration amplitudes for the current and Jumboisation versions.

5.4 The Maneuvering Model Usage

This study case will present the application of the recent build maneuvering model in the web platform. This model is fed through NPM by the back end and can be classified and a back end collaboration form. After the installation of the package via node.js the Vessel.js file can be imported using the following command in the top of the module:

```
import { Vessel } from  @ferrari212/ntnu-vessel-dopri
```

The maneuvering model uses five of the models implemented in Vessel.js library: `ship`, `shipState`, `hullRes`, `propellerInteraction`, and `fuelCons`. Basically to set up the maneuvering model the script should be written in the following order:

```
var ship = new Vessel.Ship(obj)

var shipState = new Vessel.ShipState(ship.designState.
    getSpecification())

const WAVE = new Vessel.WaveCreator()
var hullRes = new Vessel.HullResistance(ship, shipState, propeller,
    WAVE)
hullRes.writeOutput()

var propellerInteraction = new Vessel.PropellerInteraction(ship,
    shipState, propeller)
propellerInteraction.writeOutput()

var fuelCons = new Vessel.FuelConsumption(ship, shipState, plant)
fuelCons.writeOutput()

var manoeuvring = new Vessel.Manoeuvring(ship, shipState, hullRes,
    propellerInteraction, fuelCons, man)
manoeuvringMovement.writeOutput()

var manoeuvringMovement = new ManoeuvringMovement(manoeuvring)
```

```

this.clock = new THREE.Clock()
this.time = this.clock.getElapsedTime()
```

The `obj`, `propeller`, `powerPlant` and `man` corresponds to the JSON data description of ship components, as explained in Subsection 3.3.1 and Í. A. Fonseca (2016). The method `writeOutput()` is used to synchronize all values in different models ensuring all have the same states of draft and speed. The object `this.clock` is a three.js method to track the time, inside this object the method `getElapsedTime()` returns the milliseconds since the previews call was instantiated and the object `this.time` is responsible for storing the elapsed time value.

After the creation of the `manoeuvring` and `manoeuvringMovement` variables we can insert then into the loop that will process the ship displacement for each time step, as explained in the Figure 6.1. The script is described by the function `startAnimationLoop()` placed according to the template provided Appendix A. The loop function is written with comments bellow:

```
startAnimationLoop = () => {

    // Update ocean views
  if (this.ocean.name) {
    this.ocean.water.material.uniforms.time.value += 1 / 60
  }

    // Set speed and propeller rotation state
  this.manoeuvring.setSpeed(this.manoeuvring.states.V.u * 1.96)
  var propellerAngle = (this.manoeuvring.states.rudderAngle * Math.PI
      ) / 180
  var cos = Math.cos(propellerAngle)
  var sin = Math.sin(propellerAngle)

    // Calculate resistance values according to the defined speed
  var Rt = this.manoeuvring.getRes(this.manoeuvring.states.V.u)

    // Get propeller forces
```

```
var rotationStates = this.manoeuvring.getPropResult(this.
    manoeuvring.states.n)
  const distHel = manoeuvringMovement.manoeuvring.distHel

var forceVector = [
        rotationStates.Fp * cos - Rt,
        rotationStates.Fp * sin,
        rotationStates.Fp * sin * distHel
    ]

  // Stores the time step
this.manoeuvringMovement.dt = this.clock.getElapsedTime() - this.
    time
const dt = this.manoeuvringMovement.dt

  // Calculate and set the displacements
this.manoeuvringMovement.setMatrixes(forceVector, this.ship3D.
    rotation.z)
this.manoeuvringMovement.getDisplacements(dt)

this.ship3D.position.x += this.manoeuvringMovement.states.DX.x
this.ship3D.position.y += this.manoeuvringMovement.states.DX.y
this.ship3D.rotation.z = - this.manoeuvringMovement.states.yaw

  // Changes the third person camera view
this.thrirdPersonCamera.Update(dt)

// Store the time
this.time = this.clock.getElapsedTime()
```

5.4.1 User command controller testing

The `ThreeSimulation.js` file contains the script for the web platform under the simulation mode allowing two types of maneuvering experience. The first one is a free user command controller testing which will be explained in details in this subsection. The other is the IMO turning ability simulation summarized in American

Bureau of Shipping (ABS, 2017) and explained in details in the Subsection 5.5.

To allow a user controller it is necessary to make a function which will read the user command and will modify the propeller rotation and angle in real time according to the components rates capabilities. The `onDocumentKeyDown` is the function responsible for this interaction and is described as follow:

```
onDocumentKeyDown = event => {

  var keyCode = event.which
  var n = this.manoeuvringMovement.states.n
  const DT = this.manoeuvringMovement.dt
  const MAN = this.manoeuvringMovement.manoeuvring
  const F = MAN.maxPropRot / 60
  const T = MAN.maxTorque
  const L = this.manoeuvringMovement.states.load
  const LT = Math.abs(n * T)

  switch (keyCode) {
    case 87:
      if (n <= F) {
        if (L > 1 || LT < L) {
          break
        }

        this.manoeuvringMovement.states.n += MAN.helRate * DT
        // rotationText.innerText = (60 * this.manoeuvringMovement.
            states.n).toFixed(0)
        }

      break
    case 83:
      if (n >= -F) {
        if (L > 1 || LT < L) {
          break
        }

        this.manoeuvringMovement.states.n -= MAN.helRate * DT
      }

      break
    case 65:
      this.manoeuvringMovement.states.rudderAngle -= MAN.rudderRate *
           DT
      break
    case 68:
      this.manoeuvringMovement.states.rudderAngle += MAN.rudderRate *
           DT
      break
    default:
      break
  }
}
```

The numbers on the `onDocumentKeyDown` element represents the - w, s, d, a - button event, with the pair w and s to increase and reduce the propeller rotation respectively and d and a to increase and decrease the propeller angle. The function is mounted inside the document by the following script using the predefined functions for the JavaScript language:

```
document.addEventListener("keydown", e => {
    this.onDocumentKeyDown(e)
})
```

In addition to the key down listener, the simulation sets the camera in the third person aspect, right behind the ship pointing towards the ship direction. Figure 5.12 shows the free maneuvering screenshot pointing its main components: a screen with a panel in the top left side corner which shows the states of the ship and a top right side corner table with a button that changes the mode for the IMO test operation and shows IMO parameter results and status.

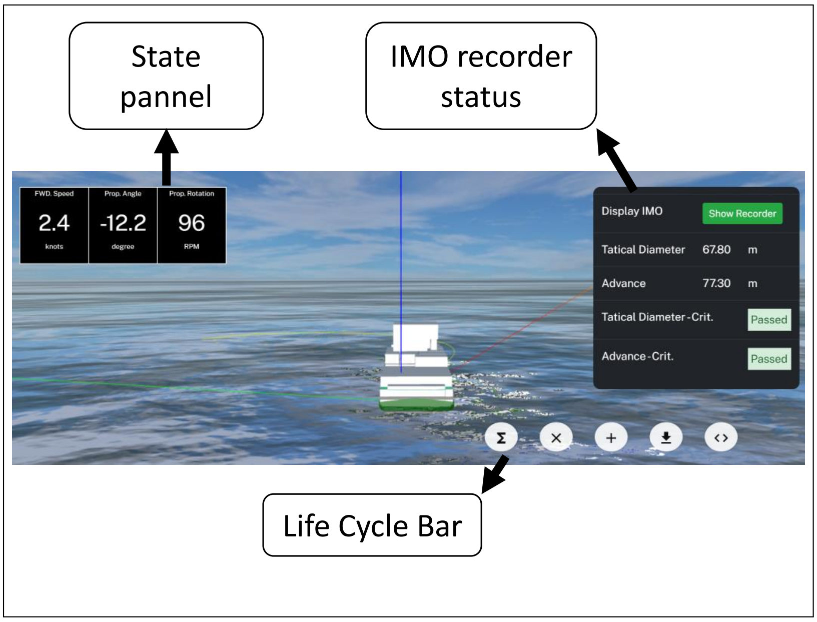


Figure 5.12: Maneuvering free control main elements.

5.4.2 Practical case on maneuvering stability model

Take as practical case the two models for Gunnerus tested in the web platform, both accessible in the https://twinship.netlify.app/ (Accessed in 2021-05-24). One of the model corresponds to the Gunnerus ship with constant damping matrix `N` and another without this information, meaning the `manoeuvringMovement` function will instantiate the matrix according to Lee et al. (2003). Both ship versions are stored in the web platform with the names of "Gunnerus Data" and "Gunnerus NoDam" respectively.

It can be noticed that the model calculated using Lee et al. (2003) has a high instability and tends to the infinite rotation displacement for whatever turning rate value. This result indicates an inconsistency in the model used that can be derived from several factors which deserve a further investigation in a own project. On this project we will go so much further in the improvement of the model since we are using the models for maneuvering as background for the back end collaboration, being limited to raise some of the possibilities for this deviation.

One of the reasons may explain the inconsistent result is the Gunnerus size detachment from the restrictions imposed by the maneuvering model as argued in Section 4.5. It is advisable to test a ship version that serves integrally the restriction from the model to find if the problem is related with the ship versions or the models implemented.

Another possible explanation can reside in the numerical approximations used. The models linearize several factors to maintain small processing time to achieve a frame rate that do not disturb the user real time experience and do not compromise the simulation time, those approximations in certain conditions can propagate to an inconsistent solution. The models chosen for example, have as restriction a smaller turning radius with a small variation in the advance speed, conditions that may not be obeyed integrally in all simulations.

For the other hand, the linear damping model presents an suitable solution, with a behavior that resembles the expected in reality. This factor helps to sustain the thesis that there is a integration problem between the Vessel.js library script and Lee et al. (2003) models. Although this deviation, the main objective of this project are kept intact since it is not in this project scope to provide a perfect model for maneuvering application but to construct a reliable web platform that are capable of handling different information in design and operation cycle stages, using the maneuvering model as an background for the collaboration between the Vessel.js library and the web platform. A further investigation of the divergence of the results may be done in specific projects using as guide lines the discussions raised above.

5.5 Turning Ability Study Case

The second maneuvering study case consists in the turning radius criteria defined in the resolution from IMO and summarized in ABS (2017). The resolution defines the advance and tactical diameter of the ship to validate the turning ability, those values are visually represented in Figure 5.13.

The IMO establishes as criteria that the tactical diameter must be smaller than five

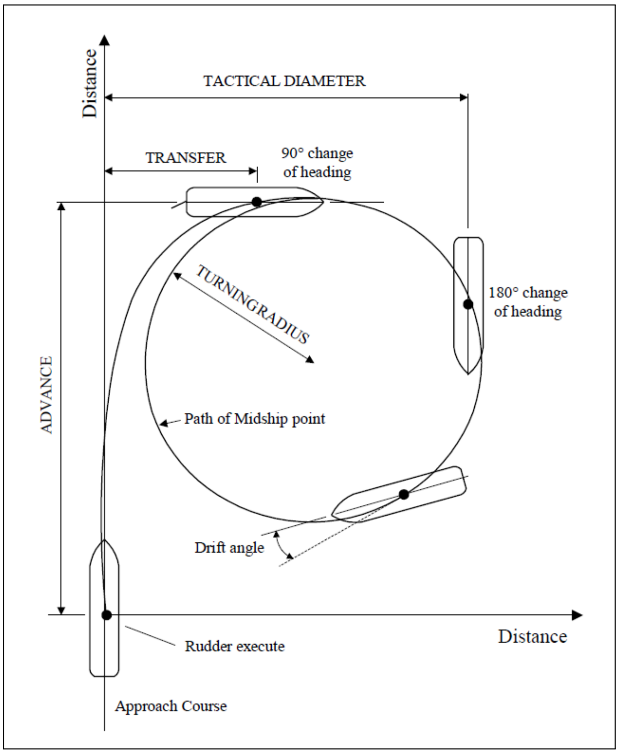


Figure 5.13: Turning cycle test parameters (ABS, 2017).

times the total length of the ship and the advance be less than four and half times the ship length. By definition, the initial condition must be held with the rudder (or in this case the azimuth propeller) without any angle, the rudder angle must then be activated until it reaches its maximum or the value of 35 degrees (the one that is the smaller). The code script is set inside a loop that will end up until the tactical diameter is reached or the simulated time is over five minutes as shown bellow.

The status for this simulation approval is shown in the IMO status panel in the Figure 5.12. The recorder button in the IMO status panel displays the position of the simulated ship accelerated ten times for the route in the IMO test. A yellow tracker marker is shown in the simulation to give the route the ship takes in this criteria. The Figure 5.14 shows three screen shot from this recorder applied to Gunnerus with the no damping matrix.

Sometimes the project may not be fitted to the restriction parameter. This is the

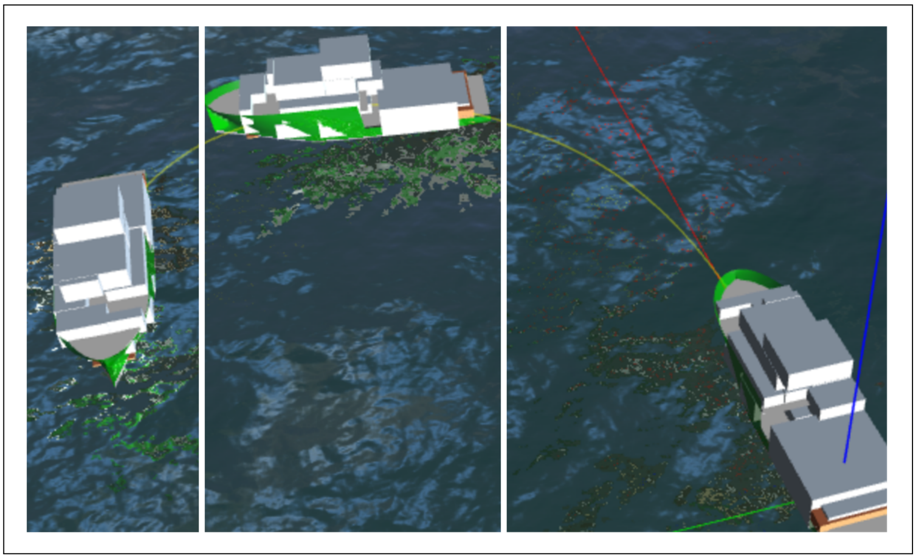
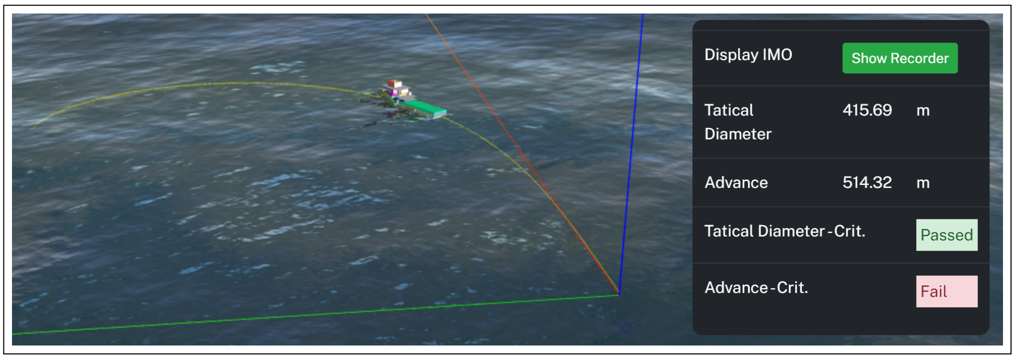


Figure 5.14: Turning cycle record visualization. The ship data is acquired in three points of the simulation.

case for the PSV named in the web platform as "PSX". Figure 5.15 gives a picture for his case, note in the status the advance criteria is not fitted.



Display IMO | Show Recorder

Tatical Diameter	415.69	m
Advance	514.32	m
Tatical Diameter-Crit.	Passed	
Advance-Crit.	Fail	

Figure 5.15: Turning screen shot for PSV named as PSX in the web platform. This version does not pass in the advance criteria as indicated in the status panel.

As next project, I recommend the development of the others maneuvering test case defined by IMO such as the course-yaw change, initial turning ability, stopping ability and the course keeping stability

5.6 Data Base Exchange - Twin Ship Approach

In this chapter we will show a real data application for Gunnerus acquired in 23th octorber 2018 between 12:52:39 and 13:09:18 for Central European Summer Time (CEST) GM+2 offered from this project purpose by NTNU university. This study case presents an application for a database collaboration, in which the information will be exchanged with the database through an external application. The data will be then read in the database, parsed and displayed to the user.

5.6.1 Supporting test app

The data will be inserted in the web platform via jupyter-notebook application which uses python as programming language. All the script that is used can be accessed in `https://github.com/ferrari212/Platform-data-parse`.

Python language together with jupyter-notebook tool were chosen for two reasons, first is the convenience python pymongo package provides to connect with mongo data base, for a detailed explanation about the package usage please consult ‘PyMongo 3.11.4 Documentation’ (2021). The second reason is the applicability of python language for data filtering that thanks to its trade off between simplicity and processing speed became one of the most common languages for data analysis and IoT integration as exposed by (Carraz et al., 2020). Since the data must be filtered from a long series, precisely a set with 1944475 acquiring points, a programming language which is both fast and easy to manage is perfect for this application. Figure 5.16 shows the context of the external juptyter notebook application in relation to the web platform.

The first phase in the data parsing was to eliminate the non numeric values and find the points which presented an non continuity in the data. The Figure 5.17 represents the location of the ship in the test according to its latitude and longitude coordinates after the date was filtered.

The location acquired was passed to the x and y coordinate systems by using the equirectangular projection. Once the distances presented are not significant high in relation to the world radius, this projection will not cause major distortion on the map

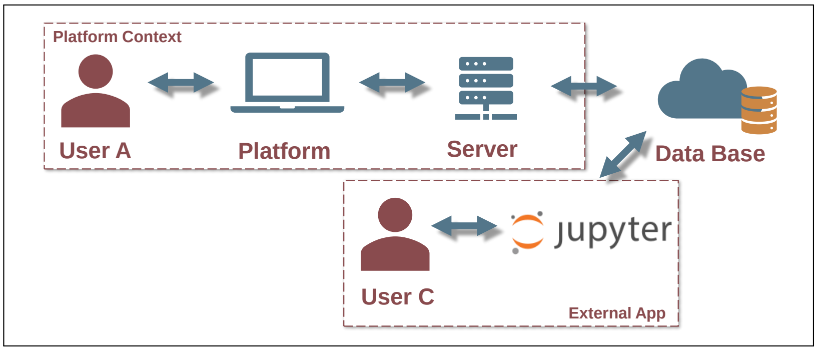


Figure 5.16: External Jupyter notebook app in relation to the main web platform.

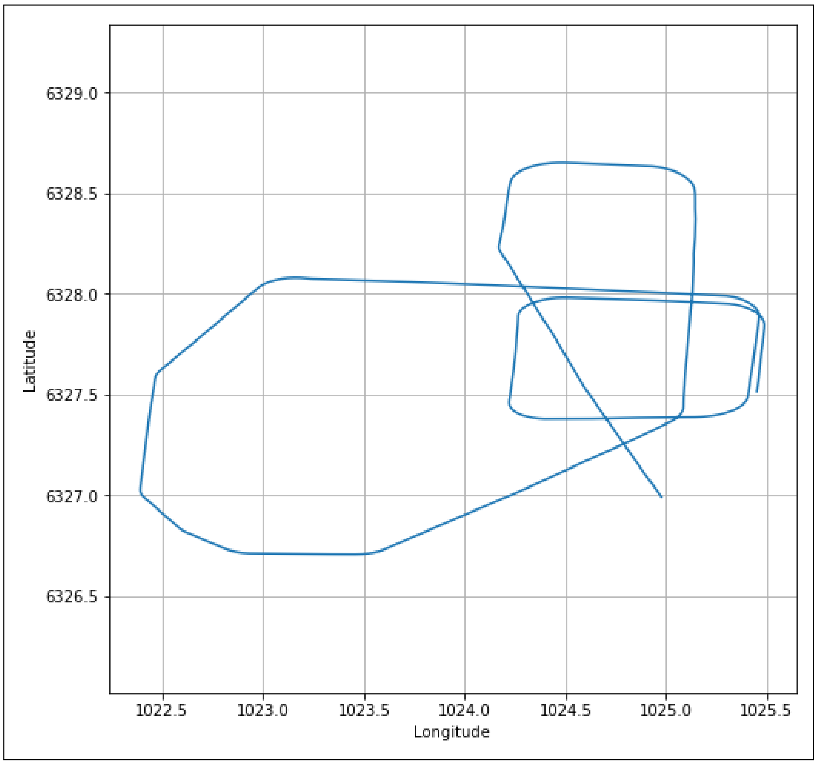


Figure 5.17: Latitude and longitude real data location

direction. The whole code from data acquiring can be conferred in .

$$\begin{cases} x = R \cdot (\lambda - \lambda_0) \cdot \cos(\varphi_1) \\ y = R \cdot (\varphi - \varphi_0) \end{cases} \tag{5.3}$$

where,

- (x, y) are the projections in coordinate system;
- (λ, φ) are the longitude and latitude;
- R is the earth radius.

After filtered the whole data, the database must be accessed via pymongo package by simply install the package and use the following command:

```
import pymongo
from pymongo import MongoClient
import dns

cluster = MongoClient("mongodb+srv://<username>:<password>
    @clustertwinship.xvvto.mongodb.net/TwinShipDatabase?
    retryWrites=true&w=majority")
```

The cluster object will hold the database information from the user and will be the element for data exchange in the format defined in Subsection 3.3.1. The open part of the web platform is set with a ship version called "Gunnerus Data" which is assigned to hold the test data. The `<username>` and the `<password>` represents respectively the owner credentials required for accessing the database, here they were suppressed for safety reasons.

5.6.2 Result validation

One of the main reasons to use digital twin is to confront the information with the models adopted to verify how well they predict the real behavior. For this purpose a specific web platform module called `ThreeTwinShip.js` was constructed to read the sensor data from the database, calculate the modeled output, and stream the results together with the original data.

The `ThreeTwinship.js` is only accessible when the `data` object has information inside it as explained in the Subsection 3.3.1. For this project the ship version which contains the test data is "Gunnerus Data" ship.

The script will read the sensor data and store the value from input for each time step to calculate the states of the ship according to the `ManeuveringModel` with a initial condition equals to real data initial condition. The diagram in Figure 5.18 helps to understand the workflow of the process. A complete description of the code can be found in `https://github.com/ferrari212/Platform-Twin-Ship/blob/main/app/components/ThreeComponents/ThreeTwinShip.js`.

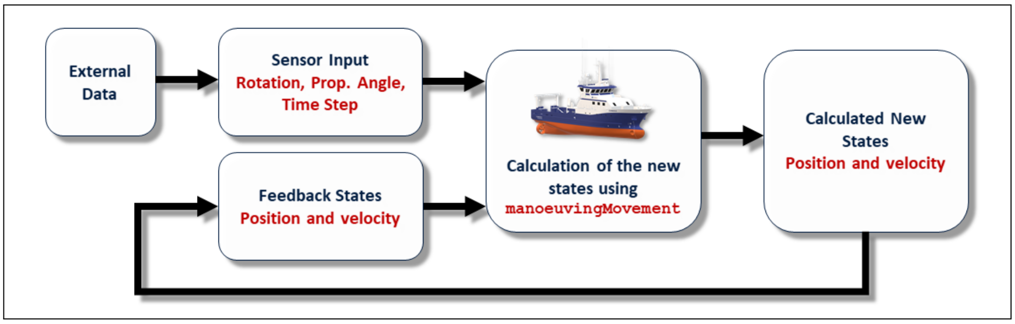


Figure 5.18: Coding loop for the twin ship application.

The result from the real model and the predicted one can be visualized in the Figure 5.19 for the Gunnerus with the linear damping condition. The white track represents the route the calculated model in Vessel.js having the real data of the propeller rotation and angle as input, the yellow track is the real data route of the ship in the field.

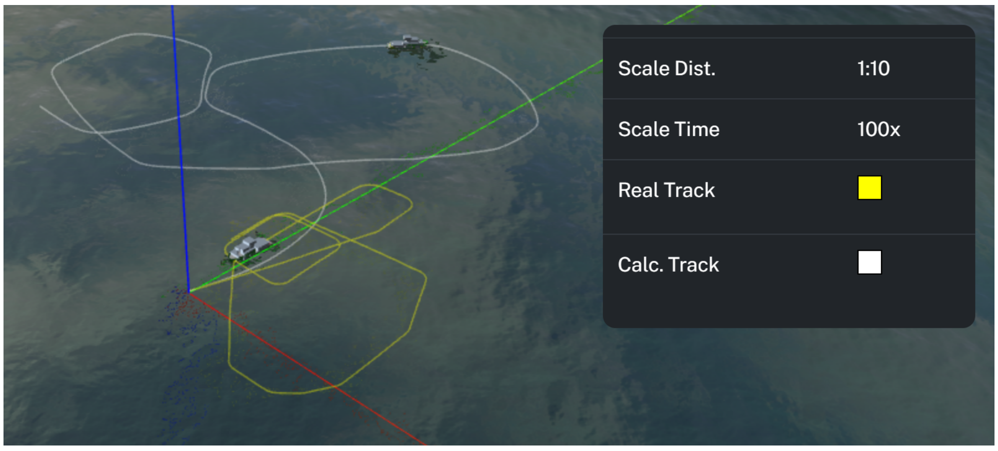


Figure 5.19: Twin ship application with a real location in yellow and a prediction in white.

As it may be seen in the figure the chosen model is far from the predicted condition because of several modeling reasons. One of the modeling restriction is the absence of the bow thrust effect in the group of forces acting in the ship. Another question that effect the exact prediction is not accounting for environmental forces such as the wind, wave and current in the course of the ship. The third reason is the hydrodynamic coefficients matrix, which were chosen with a certain arbitrage in the values and does not fidelity represent the values of the vessel.

Although those restrictions, this application gave a big step forward for applying the Vessel.js library to digital twin, being the first visual application with real ship data using the library. The modeling problems can be assessed in the future by using more consistent models of the ship. In addition, machine learning algorithms could be used to acquire a more realistic hydrodynamic coefficients based on the data set already acquired.

For the web platform perspective the model is also a huge advance in terms of accessibility. With the Gunnerus Data model settled in the web platform it becomes now simple to be accessed and reproduced by literally almost any internet user in the world, without concerns regarding compatibility. In addition the free open source allows the accessibility of the models in the GitHub repository, inviting further users to check the models and suggest improvements.

Chapter 6

Discussion

On this chapter will be discussed the web platform relevance and what are the aspects that the conclusion of this web platform will raise. Also, This chapter will be used to address some possibilities for next projects involving the web platform.

6.1 Objectives Attendance

This section will revisits the objectives stated in Section 1.3 to demonstrate the web platform attended its predefined main goals.

First, the project established a compatible structure in a holistic perspective of the several components of the web platform, using the modularization as a way for expansion. The concept of class in JavaScript was used to structure the visual models in modules, allowing a variation according to the ship model and the current state. In addition, the web platform is able to accommodate external elements, for example, the Vessel.js library improvement and the database exchange.

In the maritime description, the project used a JSON object as a way to represent ship design data, that concept is common to the Vessel.js adoption. In addition to the predefined objects, this project established an functional JSON format for maneuvering information representation. The platform data was also structured to fit a relational database, matching the ship models with the user identification number.

The web platform also presented a series of features that composes the web platform through several case studies examples. The spectrum case studies transits between the design and operation, the two phases considered in this project. This project also gives ground for the improvement of the platform until it is capable to fit the digital thread of the ship, using data across all life cycle phases.

The web platform is also a good step in comparison to the independent web applications the vessel.js were used to do. First because the web platform allows a common source of the data information, avoiding the mismatch between the projects and scenarios easing the adaptation to long term ship evaluation. In additional, the new approach can connect different simulations and analysis in a single architecture, therefore, a modification of the version can be easily reproduced in the other methods while in the independent application approach it requires a programming effort to make one model compatible to another. The modularized aspects also helps the team effort, once new different modules can be produced in an independent matter the effort can be divided to different specialized groups.

6.2 KPI evaluation

Often engineering projects relies on Key Performance Indicators (KPI) as parameters for evaluation that can be based on objective and subjective values. In this section we are going to assess the web platform application through the maintainability, scalability, efficiency and adaptability KPIs using a subjective perspective.

- **Maintainability:** The web platform is easier to maintain thanks to its modular form. An upgrade in one of the modules, both internally by the upgrade of the web platform modules or globally by the change of the servers or database types does not require huge effort in the integration. The exception applies only if the modifications in question relies on the data structure exchange and command, which a total evaluation must be made turning the modification harder;

- **Scalability:** The web platform is constructed using web based application, therefore, it can be accessed from whatever part of the globe at whatever time, making it scalable in terms of potential users and sharing. There is some storage limitations due the application of cost free technologies which can be effortless solved by increasing the funding for the application. There are as well technical limitations for a fast streaming data that are a common issue for several other projects in the industry and academia. The scalability also can be assessed in terms of new modules adaptation, in this area the web platform also presents a good improvement since new modules are easy to insert and does not affect the global performance;

- **Efficiency:** The react framework turns the interface change faster by minimizing the costly DOM process to modify the user interface (UI), being one of the main reasons for React.js application popularity. In that context a continuing modification of the UI, such as the ones in the web platform that presents several interfaces, has a performance increasing if compared to the update of each element in ordinary JavaScript pages;

- **Adaptability:** Although the web platform was planned for a ship design purpose, it is not restrained by only this perspective, having the possibility to be applied to all complex systems or focused to specific ship functions such as energy consumption or heavy lifting operations. As an open source project it permits to be forked in the repository and then goes ahead to a simultaneous project, adapting its features to different purposes.

6.3 Limitations and improvements

Although the benefits presented through this report, it is undeniable the existence of limitations that will be discussed in this section, some of those issues were already briefly pointed out during the Section 6.2 and will be discussed here with more detail. In addition, it will be pointed out what are the visible steps to improve the web platform performance and what are the aspects that deserve certain attention.

Some limitations comes with the decision about relying on free services for operational costs reduction to turn the web platform financially accessible for the maximum number of users. We can start by citing the limitation in the exchange data rate and the memory storage capacity presented in the current free services adopted, this choice prevents a fast checkout rate desirable for a twin ship operation. In fact those concerns also have huge attention in the industry and academia where technologies to turn cheaper the data exchange are extensively under development. The increase in funds for this project could solve those problems, at least in parts, by the adoption of paid services that can provide better performance in those two aspects.

Another issue found on this web platform is the resulting inconsistency for the application of Lee et al. (2003) maneuvering models with the Vessel.js library, suggesting some sort of numerical errors in the methodology. This matter was discussed with more extension in Subsection 5.6.2 and in summary the solutions presented are an investigation of ship projects that obey the method constrains and methodologies for numerical errors reduction.

The web platform smartly integrates in a modular perspective several simulations and visualizations applications which are easily expanded, however, in case the data format in the database or a command logic sequence is changed, the whole web platform must be restructured. An enhancement which could turn the web platform more suitable to accept new modules, but is time consuming to be applied, is to eliminate the sequential logic structure that requires a command chain to accomplish a task, transforming it in a more application based architecture similar to the one present in the recent smartphones operational systems. Both of the schemes can be visualized in the Figure 6.1 with the current web platform logic closer to a linear perspective, as can be seen by its task flow charts in Section 4.3, but ideally should shift towards a application based logic.

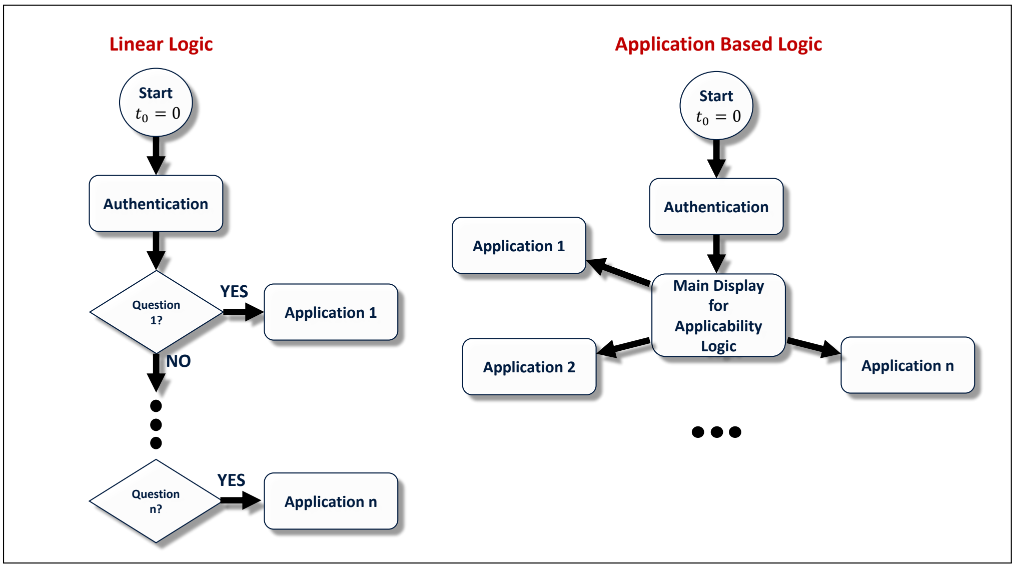


Figure 6.1: Linear and application based exemplifications.

Chapter 7

Conclusion and Future Work

Even those web web platform applications are not a novel technology, the employment in maritime context is still incipient, both for the industry as well as the academia and are even more rare inside an open and collaborative context.

This project aims to foster the idea about implementing more collaborative solutions in the maritime area as a way to assure the data source and enhance the collaboration of different partners. This chapter will be subdivided in two parts, one section to remark the fundamentals this project raised and another to suggest possible future work.

7.1 Concluding Remarks

This projects are build under the modularized technique as a way to reinforce innovation and easily permit the expansion. For this purpose the web platform was constructed under the react framework, which are associated with modular scripts basis under its construction. In addition, this choice also improves the efficiency by eliminating the unnecessary DOM upgrade for whatever modification in the page, making the web platform resembles more an offline software in terms of processing speed.

The web platform stand out from the previews works involving the Vessel.js library by being the first one to use a database as a tool for a web platform under the three-tier scheme. This implementation increased the tier modularization of the web platform, in that sense an upgrade in one of the components, for example the database technology, would not effect the function inside the other two tiers. This factor permits a future upgrade of the components in face of technology advancement to accommodate the project under future objectives.

The project also shows real cases in which the web platform can produce collaboration according to each web platform tier. The collaboration in the front-end consist in the simple input of new information and interface upgrade, used in this report to present the features introduced in the web platform and new versions. On the back-end a collaboration is given by the upgrade of the several packages the web platform is relied upon, in this context this project shows how an upgrade in the Vessel.js library with the maneuvering model insertion increases the web platform potential. Finally, the last form of collaboration is by the database information exchange, in this case an external application using python programming language and jupyter notebook framework was used to filter the experimental data and upload the information in the database without touching any component inside the web platform, as a consequence the inserted data can be then streamed and evaluated in the web platform.

7.2 Suggestions for Future Work

The web platform followed its predefined principle, in which it must not constrain the development from other users, to cote once again Andreessen (2007) it can be "adapted to countless needs and niches that the platform's original developers could not have possibly contemplated, much less had time to accommodate". In that sense, the applicability of the web platform could be extended further to unimaginable purposes inside and outside the ship design area. Although the existence of this unpredictable improvement spectrum, in this chapter I will suggest some clear improvements that could certainly increase the web platform potential.

In practical terms, the web platform would earn some potential if it could be adaptable not only according to the ship data, but also to the user needs, accommodating some sort of team management. The user experience is improved if the user in a specific design department, for example in the machinery design, would receive a web platform which only the functionalities related to this area were available and would not waste time finding its specific tools and needs in a set of unrelated interfaces.

The web platform was proposed as a way to manage different data in different contexts, allowing the transition through design and operation phases. In this specific case the web platform was applied roughly from preliminary design, design and operation (digital twin). In the future the web platform could be extended to accommodate to different phases until it is able to evaluate the whole life cycle of the project. There should be also developed a more integrated information interfaces between the different phases, helping the decision makers take information from one phase to another as conceivable in the digital thread concept.

This project was delivered under an engineering perspective, however there is also improvements inside information technology field. For example, the web platform could be shifted to an open and safety database, improving its freedom degree. Techniques such as server side calculations could also be applied to reduce the heavy calculation time and consequently improving the user experience.

Finally, the last suggestion would rely in a more intense use of machine learning techniques in order to acquire a more meaningful information, such as suggested by Erikstad (2017). A short path to this integration would be the application of TensorFlow.js library because of its extensive community providing a good ecosystem for debugging and education. The library is also open source project which eliminates legal restriction for the utilization in the web platform, and is written in JavaScript language avoiding further training and adaptation waste for a new syntax in a integration with the current web platform. The tensor flow also has a primarily base in python script, and could also be used in the context of an external application as done for the data filtering in this project using the jupyter notebook.

Bibliography

American Bureau of Shipping. (2017). Guide for vessel maneuverability. *Marine Design Conference*, 111.

Andrade, S. L. & Gaspar, H. M. (2015). Ship motion application [Site accessed in 2020-12-11]. http://www.shiplab.hials.org/app/shipmotion/

Andreessen, M. (2007). Analyzing the facebook platform, three weeks in. *Blog pmarca*. https://web.archive.org/web/20071002070223/http://blog.pmarca.com/2007/06/analyzing_the_f.html

Andrews, D. & Dicks, C. (1997). The building block design methodology applied to advanced naval ship design. *Marine Design Conference*.

Boschert, S. & Rosen, R. (2016). Digital twin—the simulation aspect. In P. Hehenberger & D. Bradley (Eds.), *Mechatronic futures: Challenges and solutions for mechatronic systems and their designers* (pp. 59–74). Springer International Publishing. https://doi.org/10.1007/978-3-319-32156-1_5

Briano, E. & Caballini, C. (2012). Simulation as a support tool for training logistic operators, 188–193.

Carraz, M., Koraktis, K., Crocker, P., Muir, R. & Voskoglou, C. (2020). *State of the developer nation 18th edition* (Technical Report). SLASHDATA. https://developereconomics.com/resources/reports/

Chaves, O. S. (2018). *A knowledge-based approach for automated design of hull scantling* (Master's Thesis). Norwegian University of Science and Technology. Ålesund.

Clarke, D., Gedling, P. & Hine, G. (1983). The application of manoeuvring criteria in hull design using linear theory. *RINA*.

Class es6 description. (n.d.). `https://262.ecma-international.org/6.0/#sec-class-definitions` (accessed: 18.05.2021)

Coraddu, A., Oneto, L., Baldi, F., Cipollini, F., Atlar, M. & Savio, S. (2019). Data-driven ship digital twin for estimating the speed loss caused by the marine fouling. *Ocean Engineering*, *186*, 106063. `https://doi.org/https://doi.org/10.1016/j.oceaneng.2019.05.045`

Danielsen-Haces, A. (2018). *Digital twin development - condition monitoring and simulation comparison for the revolt autonomous model ship* (Master's Thesis). Norwegian University of Science and Technology. Throndheim.

DoD. (2004). *Dod architecture framework working group* (Technical Report). Department of Defense (DoD). `http://www.acqnotes.com/Attachments/DoDAF_v1_Volume_I.pdf`

El Saddik, A. (2018). Digital twins: The convergence of multimedia technologies. *IEEE MultiMedia*, *25*(2), 87–92. `https://doi.org/10.1109/MMUL.2018.023121167`

Erikstad, S. (2017). Merging physics, big data analytics and simulation for the next-generation digital twins.

Escamilla i Miquel, S. (2019). *Open source simulation for virtual maritime operations* (Master's Thesis). Norwegian University of Science and Technology. Ålesund.

Evans, D. S., Hagiu, A. & Schmalensee, R. (2006). *Invisible engines: How software platforms drive innovation and transform industries*. The MIT Press. `https://doi.org/10.7551/mitpress/3959.001.0001`

Fonseca, Í. A. (2016). *An open and collaborative object-oriented taxonomy for simulation of marine operations* (Master's Thesis). Norwegian University of Science and Technology. Ålesund.

Fonseca, Í. A. (2020). *Design and prototyping of a digital twin platform for r/v gunnerus* (Technical Report). NTNU.

Fonseca, I. A., Gaspar, H. M., de Mello, P. C. & Sasaki, H. A. U. (2021). Standard for digital twin data applied to anexperiment with a scale model ship [Article under review phase]. *Computer-Aided Design*.

Fonseca, Í. A. & Gaspar, H. M. (2019). A prime on web-based simulation. *European Council for Modelling and Simulation*, (33). http://www.scs-europe.net/dlib/2019/2019-0023.htm

Fonseca, Í. A. & Gaspar, H. M. (2020). Challenges when creating a cohesive digital twin ship: A data modelling perspective. *Ship Technology Research*. https://doi.org/https://doi.org/10.1016/j.cad.2007.06.012

Fossen, T. I. (2011). *Handbook marine craft hydrodynamics and motion control*. John Wiley; Sons.

Gaspar, H. M., Brett, P., Ebrahimi, A. & Keane, A. (2014). Data-driven documents (d3) applied to conceptual ship design knowledge.

Gaspar, H. M. (2016). Vessel.js: An open and collaborative ship design object-oriented library. *International Marine Design Conference*, (13).

Glaessgen, E. & Stargel, D. (2012). *The digital twin paradigm for future nasa and u.s. air force vehicles* (tech. rep.). https://doi.org/10.2514/6.2012-1818

Hatledal, L. I., Styve, A., Hovland, G. & Zhang, H. (2019). A language and platform independent co-simulation framework based on the functional mock-up interface. *IEEE Access*, *7*, 109328–109339. https://doi.org/10.1109/ACCESS.2019.2933275

He, B., Wang, Y., Song, W. & Tang, W. (2015). Design resource management for virtual prototyping in product collaborative design. *Proceedings of the Insti-*

tution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, *229*(12), 2284–2300. https://doi.org/10.1177/0954405414551106

Helmond, A. (2007). The platformization of the web: Making web data platform ready. *Sage Journals*, *1*. https://journals.sagepub.com/doi/full/10.1177/2056305115603080

Hildre, H. P., Mork, O. J. & Æsøy, V. (2010). Modular architectures -the third industrial revolution. *The maritime innovation factory*. Høgskolen i Ålesund.

Holtrop, J. (1984). A statistical re-analysis of resistance and propulsion data, 272–276.

international towing tank conference. (2005). The manoeuvring committee. *Final report and recommendations to the 24th ITTC*, 62.

Jensen, J. J., Mansour, A. E. & Olsen, A. S. (2004). Estimation of ship motions using closed-form expressions. *Ocean Engineering*, *31*(1), 61–85. https://doi.org/https://doi.org/10.1016/S0029-8018(03)00108-2

Kimura, T. (2009). On dormand-prince method. *Universidad Nacional Autónoma de México*. http://depa.fquim.unam.mx/amyd/archivero/DormandPrince_19856.pdf

Landolo, N. (2019). Benetau puts digital thread to the test [PTC study case, site accessed in 2020-12-11]. https://www.ptc.com/en/blogs/plm/beneteau-puts-digital-thread-to-the-test

Lee, T.-I., Ahn, K.-S., Lee, H.-S. & Yum, D.-J. (2003). On an empirical prediction of hydrodynamic coefficients for modern ship hulls. *Marsin 03*.

Levander, K. (2012). *System based ship design kompendium, lecture notes*. NTNU.

McKinsey&Company. (2016). The age of analytics: Competing in a data-driven world. (1), 1–136.

Monteiro, T. & Gaspar, H. (2016). An open source approach for a conceptual ship design tools library.

Morgan, A. (2020). The modern application stack – part 1: Introducing the MEAN stack. *Mongo DB blog pmarca*. https://www.mongodb.com/blog/

post/the-modern-application-stack-part-1-introducing-the-mean-stack

Morgan, A. & Pawling, R. (2009). The impact of simulation on preliminary ship design. *Marine Design Conference*.

Oliveira, F. F. (2021a). From concept to simulation - a vessel.js tutorial. *Observable HQ - online report*. https://observablehq.com/@ferrari212/from-the-hull-to-simulation-a-vessel-js-tutorial

Oliveira, F. F. (2021b). Vessel.js npm repository [site accessed in 2021-05-17]. https://www.npmjs.com/package/@ferrari212/ntnu-vessel-dopri

Oliveira, F. F., Prata, D. & Gaspar, H. M. (2021). Salt caves [site accessed in 2021-05-29]. https://threejs.org/

Open Source Initiative. (2007). *The open source definition* (tech. rep.). https://opensource.org/osd

O'Reilly, T. (2007). What is web 2.0: Design patterns and business models for the next generation of software. *International Journal of Digital Economics*, (65). https://mpra.ub.uni-muenchen.de/4580/

Pymongo 3.11.4 documentation [site accessed in 2021-05-25]. (2021). https://pymongo.readthedocs.io/en/stable/index.html#contributing

Rachuri, S., Subrahmanian, E., Bouras, A., Fenves, S. J., Foufou, S. & Sriram, R. D. (2008). Information sharing and exchange in the context of product lifecycle management: Role of standards [Current State and Future of Product Data Technologies (PDT)]. *Computer-Aided Design*, *40*(7), 789–800. https://doi.org/https://doi.org/10.1016/j.cad.2007.06.012

Schiff, B. (2020). Model template for a platform framework using MERN architecture [GitHub repository accessed in 2020-12-09]. https://github.com/LearnWebCode/react-course

Schirmann, M., J., M. C. & Gose. (2019). Ship motion and fatigue damage estimation via a digital twin. *International Symposium on Life-Cycle Civil Engineering (IALCCE)*.

Sen, D. T. & Vinh, T. C. (2016). Determination of added mass and inertia moment of marine ships moving in 6 degrees of freedom. *International Journal of Transportation Engineering and Technology*.

Shiplab website [site accessed in 2021-05-17]. (2021). `http://www.shiplab.hials.org/`

Singh, V. & Willcox, K. E. (2018). Engineering design with digital thread. *AIAA Journal*, *56*(11), 4515–4528. `https://doi.org/10.2514/1.J057255`

Sowa, J. F. & Zachman, J. A. (1992). Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, *31*(3).

Taber, M., Zhang, J., Strom, T., Immerman, D. & Duncan, D. (2019). *Digital thread, building continuity across products, processes, and people* (White Paper). PTC. `https://www.ptc.com/-/media/Files/PDFs/PLM/J14289-Digital-Thread-Whitepaper-Single.pdf`

Three.js website [site accessed in 2021-05-29]. (2021). `https://threejs.org/`

Triantafyllou, S., M. & Hover, F. S. (2004). Maneuvering and control of marine vehicles. *MIT press*. `https://dspace.mit.edu/bitstream/handle/1721.1/36835/13-49Fall-2004/NR/rdonlyres/Ocean-Engineering/13-49Fall-2004/2D8F821F-A953-4631-9A8B-9DDC2ED755BB/0/1349_notes.pdf`

Varela, J., Rodrigues, J. & Soares, C. G. (2015). 3D simulation of ship motions to support the planning of rescue operations on damaged ships [International Conference On Computational Science, ICCS 2015]. *Procedia Computer Science*, *51*, 2397–2405. `https://doi.org/https://doi.org/10.1016/j.procs.2015.05.416`

Vieira, D. P., Nishimoto, K., de Oliveira, F. F. & Gaspar, H. M. (2020). Simulation of the conceptual design of offshore salt caves for CO2 storage. *34*, 106063. `https://doi.org/10.7148/2020-0214`

Vindøy, V. (2020). *A functionally oriented vessel data model used as basis for classification* (Technical Report). Det Norske Veritas. `https://data.dnvgl.com/dnvgl-vis/`

Vorontsova, M. (2019). React 16 + three.js integration tips (2019) [Site accessed in 2020-12-11]. `https://codeburst.io/react-16-three-js-integration-tips-2019-b6afe19c0b83`

Vossen, C., Kleppe, R. & Hjørungnes, S. (2013). Ship design and system integration.

Wang, G. G. (2003). Definition and Review of Virtual Prototyping. *Journal of Computing and Information Science in Engineering*, *2*(3), 232–236. `https://doi.org/10.1115/1.1526508`

Xantic. (2001). *Sfi group system - a system for classification of technical and economic ship information* (Product Description). PTC. `https://www.xantic.net/`

Appendix A

Listing Three.js Template

The listing bellow shows the main template for creating a Three.js model with the vessel.js library in React framework second the article writen by Vorontsova (2019). The module is structured in the three phases of constructor, mount, and updated as explained in details in Subsection 3.2.1.

```
// Import React and three
import React, { useEffect, Component } from "react"
import * as THREE from "three"

// Import Main Page
import Page from "./Page"

// Import Vessel js and ship 3D
import { Vessel } from "../vessel/build/vessel"
import { Ship3D } from "../vessel/build/Ship3D"


class ThreeModel extends Component {

    constructor(props) {
        // Mount the Globals and Props Here
        super(props)

    }

    // Component Events

  componentDidMount() {
      // Apply the methods right after the component mount

      this.sceneSetup()

      window.addEventListener("resize", this.handleWindowResize)

      window.addEventListener("resize", this.handleWindowResize)
  }

  componentDidUpdate(prevVersion) {
      // Apply the methods right after the states update

      if (prevVersion !== newVersion) {

          this.removeShip()
          this.addShip()

          } else {
          this.ship = new Vessel.Ship(this.state.newShip)

          this.addScenario()

          this.addShip()

          this.startAnimationLoop()
          }
     }

  sceneSetup = () => {
      // Setup the scene
      this.renderer = new THREE.WebGLRenderer({ antialias: true })
  }

  addScenario = () => {
    // Insert objects and make the necessary calculations
  }

  addShip = () => {
    // Add ship 3D
  }

  removeShip = () => {
      // Remove ship 3D
  }

  handleWindowResize = () => {
      // Maintain the figure with resizing
  }

  startAnimationLoop = () => {
    // Applies the render
    this.renderer.render(this.scene, this.camera)
    this.requestID = window.requestAnimationFrame(this.
        startAnimationLoop)
  }

  render() {
    return (
    // Imput component back to the main page
      <Page title="Three-js">
        <div ref={ref => (this.mount = ref)} />
      </Page>
    )
  }
}

// Export back to the ThreeModel
export default ThreeModel
```

Appendix B

Chart Modules

The chart modules can be divided into tow parts: one is the configuration default object containing the predefined information into the web platform and the render container that will return the information back to the web platform.

The default object uses a static instance and in that sense it will acquire the data only if a defined value is inserted into the web platform. The following snippet shows the code for the line chart type, the other modules for the other types of charts can be found in `https://github.com/ferrari212/Platform-Twin-Ship/tree/main/app/components/ChartComponents`.

```
import { right } from "@popperjs/core"
import React, { Component } from "react"
import { Line } from "react-chartjs-2"

class LineChart extends Component {
  constructor(props) {
    super(props)
    this.state = {
      chartData: props.chartData
    }
  }

```

```
static defaultProps = {
  height: 400,
  displayTitle: true,
  displayLegend: true,
  textTitle: "Bar Chart",
  legendPosition: "right",
  legendAlign: "center",
  xLabel: "xLabel",
  yLabel: "yLabel"
}

render() {
  return (
    <div className="chart">
      <div className="container">
        <div className="row">
          <div className="col-sm"></div>
          <div className="col-sm-9">
            <Line
              data={this.state.chartData}
              height={this.props.height}
              options={{
                title: {
                  display: this.props.displayTitle,
                  text: this.props.textTitle,
                  fontSize: 18
                },
                legend: {
                  display: this.props.displayLegend,
                  position: this.props.legendPosition,
                  align: this.props.legendAlign
                },
                maintainAspectRatio: false,
                fill: false,
                scales: {
                  xAxes: [
                    {
                        scaleLabel: {
                          display: true,
                          labelString: this.props.xLabel
                        }
                      }
                    ],
                    yAxes: [
                      {
                        scaleLabel: {
                          display: true,
                          labelString: this.props.yLabel
                        }
                      }
                    ]
                  }
                }}
              />
            </div>
            <div className="col-sm"></div>
          </div>
        </div>
      </div>
    )
  }
}

export default LineChart
```

Appendix C

Manoeuvring

C.1 Derivative form for the maneuvering equation

To solve in time the equations of movement for the ship we must rewrite the Equation C.12 in Appendix C to find an equation in the form of $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{B}$ that is compatible with the derivatives approximation methods. To find a solution in this form we can rearrange the elements in the equation, starting by writing the equation in according to newton law format, using as reference Fossen (2011), separating the terms into acceleration and velocity dependent:

$$\mathbf{M}.\dot{V} + \mathbf{N}(V).V = \mathbf{F} \tag{C.1}$$

The value of **M** and **N** can be separated in two terms:

$$\begin{cases} \mathbf{M} = M_{RB} + M_A \\ \mathbf{N}(V) = C_A(V) + C_{RB}(V) + D(V) \end{cases} \tag{C.2}$$

The initials RB mean rigid body, therefore the term M_{RB} and C_{RB} represents the matrix of inertia of the whole body and the Coriolis effects due the representation in a rotational system without accounting for hydrodynamic effects. The terms marked with A are referent to the fluid added mass dependent, when a body moves around a fluid the variation of pressure makes part of the fluid particles move with the body, causing a virtual effect of additional mass, this effect is presented in the acceleration terms like M_A and in its Coriolis impression in C_A. The last term $D(V)$ relates with viscous friction between the fluid and the body walls.

Fossen (2011) presents different models to arrange the variables depending on the complexity modeling which in general would have a trade off between the method accuracy and computational time for the application in a simulation. This project opted for the simplest solution to reduce the time step in expense of a bigger approximation error with the argument that a higher calculation time can spoil the user experience by subtracting the realistic perception that fast response rates offer. There is also an undesirable numerical error presented in derivative approximation by the increment of time step that is reduced by using simpler methodologies. The management between the computational time and complexity solution is in fact a broader concept that must be taken with care and has space for creative solutions.

Having that in mind the maneuvering is programmed using a linearized three degree of freedom model. This model is indicated for predominant and slow varying foward speed $|V| = \sqrt{u^2 + v^2} \approx u$. The problem can be then divide in two approaches: the foward speed and Sway-Yaw model equations.

The forward speed computes the resistance in the same manner as in a straight line route situation:

$$(m - X_{\dot{u}}) \cdot \dot{u} - X_u \cdot u - X_{u|u|} \cdot u \cdot |u| = f_x \tag{C.3}$$

Using a quasi static approximation, we can use the Holtrop (1984) models already developed in Vessel.js library to compute the hydrodynamic forward forces as stated in Equation C.13. For performance reasons the Resistance curve will be computed as a quadratic function represented by $k \cdot u^2 = R_T$, with the values of k a numerical constant approximate for the project speed u_p and the resistance of project R_p in its service speed:

$$k = \frac{{u_p}^2}{R_p} \tag{C.4}$$

The Sway-Yaw model can be used written together with in Equation C.4 to find the matrices in Equation C.2 for a reference system in the center of gravity of the ship:

$$\mathbf{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m - Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -Y_{\dot{r}} & I_{zz} - N_{\dot{r}} \end{bmatrix} \tag{C.5}$$

$$\mathbf{N} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -Y_v & m \cdot u - Y_r \\ 0 & -Y_{\dot{v}} \cdot u - N_v & Y_{\dot{r}} - N_r \end{bmatrix} \tag{C.6}$$

Writing the terms on a global fixed coordinate system for the position and yaw angle $\eta = [x\ y\ \varphi]^T$:

$$\dot{\eta} = R(\delta).V^T \tag{C.7}$$

$$R(\delta) = \begin{bmatrix} cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.8}$$

Finally, we can reach the ideal equation for the global system represented by the matrix $\mathbf{X}_{6\times 1}$:

$$\dot{\mathbf{X}}_{6\times 1} = \mathbf{A}_{6\times 6}\mathbf{X}_{6\times 1} + \mathbf{B}_{6\times 1} \tag{C.9}$$

Defining $\mathbf{null}_{i\times j}$ as a null matrix with size $(i, j) \in N$ we can write the coefficients $\mathbf{A}_{6\times 6}$ and $\mathbf{B}_{6\times 1}$:

$$\mathbf{A}_{6\times 6} = \begin{bmatrix} \mathbf{null}_{3\times 3} & R(\delta) \\ \mathbf{null}_{3\times 3} & -\mathbf{M}^{-1}\mathbf{N} \end{bmatrix} \tag{C.10}$$

$$\mathbf{B}_{6\times 1} = \begin{bmatrix} \mathbf{null}_{3\times 1} \\ \mathbf{M}^{-1}\mathbf{F} \end{bmatrix} \tag{C.11}$$

The Equation C.9 can be solved using the Dormand-prince method already written inside the `numeric.js` library used for matricial calculation in this project. The Dormand-prince solution is explained in detail in the Kimura (2009).

C.2 Manoeuvring Simulation Model

The manoeuvring model is the chosen component to be developed in Vessel.js according to the arguments raised in Subsection 2.2.1. Differently from other models consolidated inside Vessel.js the current prototype is derived from differential equations that rules the physical dynamics of the system. Using the notation applied in ABS (2017), we can write the second Newtown law applied in a reference system coupled with the ship center of gravity in three degrees of freedom (surge, sway and yaw), that are the most relevant for the manoeuvring calculations:

$$\begin{cases} m \cdot [\dot{u} - vr] = \mathbf{X} \\ m \cdot [\dot{v} + ur] = \mathbf{Y} \\ I_{zz} = \mathbf{N} \end{cases} \tag{C.12}$$

Where:

- m: ship total mass $[kg]$;
- $[\mathbf{X},\ \mathbf{Y},\ \mathbf{N}]$: Total forces in x and y directions and total momentum around z axis $[N; N, N \cdot m]$;
- $V = [u, v, r]$: surge, sway and yaw rate velocities $[m/s;\ m/s,\ rad/s]$;
- $[\dot{u}, \dot{v}, \dot{r}]$: surge, sway and yaw rate accelerations $[m/s^2;\ m/s^2,\ rad/s^2]$;

- I_{zz}: ship total moment of inertia $[kg \cdot m^2]$;

The vector in $\vec{X} = (\mathbf{X}, \mathbf{Y}, \mathbf{N})$ are the result of forces and momentum in relation to local (x, y, z) coordinates. The components are the vector sum of inner forces such as hull resistance and propeller thrust with external forces, for example, wind forces and current. In this project no extra external forces will be modeled, except for as estimation of the wave added forces that were already settled in Vessel.js library advance resistance.

The $\mathbf{X}$ represents the resulting advance balance modeled as a difference between the hull resistance force, calculated via Hooltrop formulas (Holtrop, 1984), and the propeller modeled using the Wageningen B-series shape according to Triantafyllou et al. (2004), with both models already integrated inside the Vessel.js lybrary. This approximation neglects the non linear components and crossed dimensions effect.

$$\mathbf{X} = T_{x(\text{Wageningen})} - R_{(\text{Holltrop})} \tag{C.13}$$

The inertia term I_{zz} in Equation C.12 can be approximated by an ellipsoid according to the Sen and Vinh (2016), giving the formula C.14. It is important to have in mind that this approximation is limited, and it is as accurate as much the ship shape resembles an ellipsoid form.

$$I_{zz} = \frac{1}{120}\pi\rho LBT(B^2 + L^2) \tag{C.14}$$

The propeller were modeled assuming an azimuth thrust, since all self propelled vessels to be presented as study case in this report use this technology, therefore no rudder modeling were carried and the simulations allow the rotation of the propeller 360°. The configuration also does not comport the variation of the hull propeller interaction due the angle variation, more specifically the variations in the coefficients w and t. This approximation was taking having in mind that no simple formula is easily accessible for those coefficients estimation. A study to compute this level of detail would require a CFD evaluation or a towing tank experiment specific for each type of hull and propeller geometry in a certain condition of propeller angle and velocity, making extremely hard to fit a single model to all possible variables admitted by the web platform .

The forces can be approximated over a first order Taylor expansion, which assumes $\partial^2 \vec{X}/\partial U^2 \approx 0$ and the decoupling between the forces and velocities in surge from its sway and yaw counterparts, giving $\partial Y/\partial u \approx \partial N/\partial u \approx 0$:

$$\begin{cases} \mathbf{Y} = \frac{\partial Y}{\partial \dot{v}} \cdot \dot{v} + \frac{\partial Y}{\partial \dot{r}} \cdot \dot{r} + \frac{\partial Y}{\partial v} \cdot v + \frac{\partial Y}{\partial r} \cdot r + T_y \\ \mathbf{N} = \frac{\partial N}{\partial \dot{v}} \cdot \dot{v} + \frac{\partial N}{\partial \dot{r}} \cdot \dot{r} + \frac{\partial N}{\partial v} \cdot v + \frac{\partial N}{\partial r} \cdot r + T_y \cdot d \end{cases} \tag{C.15}$$

The force T_y represents the inner force result in transverse direction, the term d represent the arm of this force in relation to the center of gravity of the ship, where the product $T_y \cdot d$ is the resulting momentum around z axis.

The derivatives used in those formulas are called "Hydrodynamic Derivatives" and they can be assessed by three ways: computation fluid dynamics routines, scaled model tests, and data base statistics modeling. The computational fluid dynamics use numerical methods for this estimation, being a more controlled prediction alternative and a way for costs reduction by eliminating the expenditure in sensors and infrastructure. For the other hand, the numerical methods always presents some imprecision derived from the nature of the methodology, which in best cases are sufficiently small depending on the ability of the design to be guided upon the available methods and mesh styles for the situation. Another drawback for the numerical methods is the high computational power requirement to shorter the predictions, which can be an obstacle for some simulation such as the real time calculation proposed in this report.

The scaled model tests and data base statics uses the real physical environment for test, eliminating the numerical errors in computational fluid dynamics. The method is also less operator dependent, since it relies more in the application procedure than the choose of complex mathematical models. For the other hand those methods arouse experimental errors coming from the sensors and have less control from the physical constants such as environmental temperature and pressure.

In summary, each of the alternatives have its own advantages and drawbacks and must be chosen according to the resources available. Having in mind this aspect, the web platform will give freedom for the user to input the data relative to the hydrodynamic derivatives, that could be estimated from whatever method type, or once those information are absent the library will estimate the hydrodynamic parameters using system based formulas. The Figure C.1 summarizes the three type of estimations categorizing them into a no simulation, system based simulation or CFD simulation group.

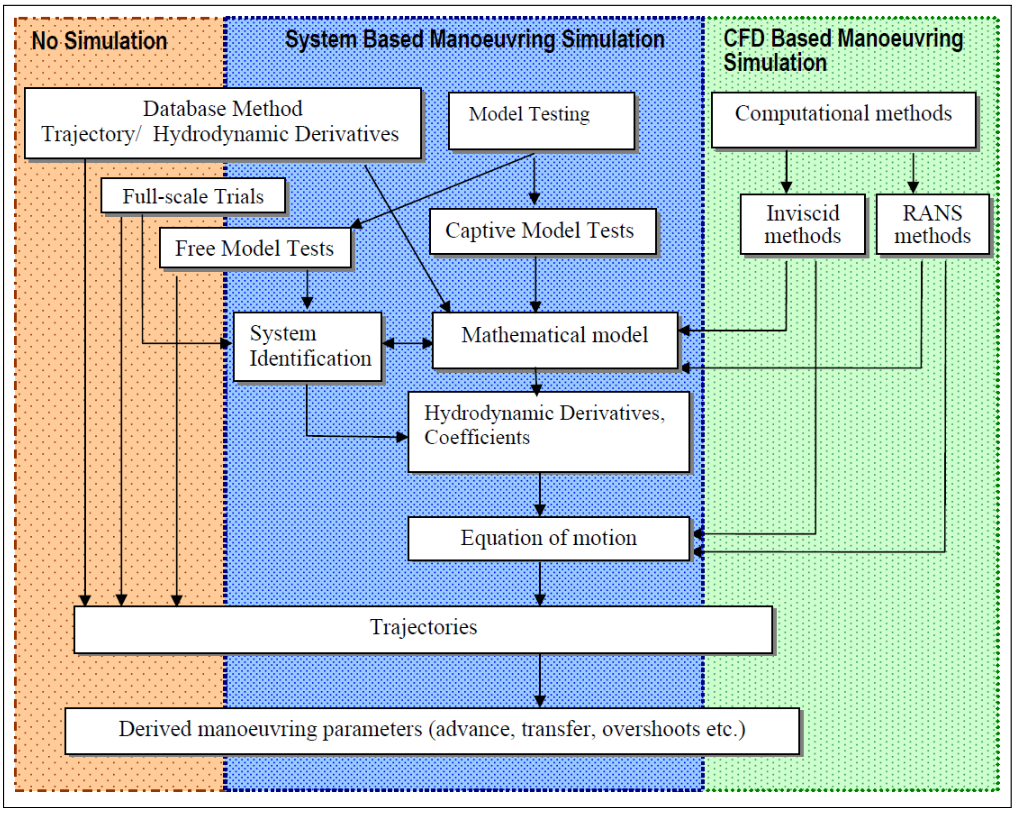

No Simulation
System Based Manoeuvring Simulation
CFD Based Manoeuvring Simulation
Database Method Trajectory/ Hydrodynamic Derivatives
Model Testing
Computational methods
Full-scale Trials
Free Model Tests
Captive Model Tests
Inviscid methods
RANS methods
System Identification
Mathematical model
Hydrodynamic Derivatives, Coefficients
Equation of motion
Trajectories
Derived manoeuvring parameters (advance, transfer, overshoots etc.)

Figure C.1: Manoeuvring prediction methods and definition of "System Based Manouvring Simulation", (ITTC, 2005)

The system based formulas used to estimate the forces **Y** and **N** in the absence of further information are the ones developed by Lee et al. (2003). The formulas were acquired through several statistics towing experiments of planer motion mechanism (PMM) for different ship types in the ballast and unballast condition. The ranges of the ships used and the ones that are more suitable for the formula application are C_b :

$0,55 \sim 0.87, d/L : 0.022 \sim 0.071, L/B : 5.0 \sim 8.8, C_B \cdot (B/L) : 0.075 \sim 0.166$. The formulas are presented as follow:

$$\begin{cases} {Y_v}' = -(0.145 + 2.25(d/L) - 0.2\Delta_{SR}) \\ {Y_r}' - m = -(0.282 + 0.1\Delta_{SR}) + (0.0086\Delta_{B/L} + 0.004) \cdot (L/d) \\ {N_v}' = -(0.222 + 0.1\Delta_{SR}) + 0.00484(L/D) \\ {N_r}' = -(0.0424 - 0.03\Delta_{SR}) - (0.004\Delta_{C_b} - 0.00027) \cdot (L/d) \end{cases} \tag{C.16}$$

Where:

- $\Delta_{C_B} = 1 - C_b/P_{C_b}$;
- $P_{C_b} = 1.12(d/L) + 0.735$;
- $\Delta_{B/L} = 1 - (B/L)/0.18$;
- $\Delta_{SR} = 1 - S_R/P_{SR}$;
- $S_R = B_{P07}/B_{PS}$, where B_{PS} are the half breadth at the height of the propeller in 2.0 station, and B_{PS} are the half breadth at height of $0.7R$ (Propeller radius) in station 2.0. The Figure C.2 shows visually how the measurements must be taken;
- $P_{SR} = 28.7\nabla' + 0.54$;
- $\nabla' = \nabla/L^3$;

The apostrophe in the Equation C.16 indicates the terms are written in their adimensional form. The terms can be written in their definitive form according to Equation C.17:

$$\begin{cases} {Y_v}' = \frac{Y_v}{0.5\rho L^2 |V|}; \ {Y_r}' = \frac{Y_r}{0.5\rho L^3 |V|} \\ {N_v}' = \frac{N_v}{0.5\rho L^3 |V|}; \ {N_r}' = \frac{N_r}{0.5\rho L^4 |V|}; \end{cases} \tag{C.17}$$

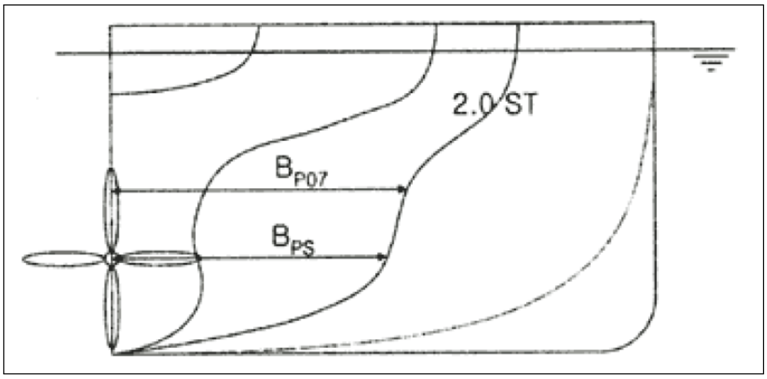


Figure C.2: Measurement of B_{P07} and B_{PS}, Reference: https://nippon.zaidan.info

The acceleration derivatives can be expressed using the Clarke et al. (1983) formulas that is based in 36 PMM experiments:

$$\begin{cases} Y_{\dot{v}}' = -\pi \left(\frac{T}{L}\right)^2 \left(1 + 0.16\frac{C_B \cdot B}{T} - 5.1\frac{B}{L}^2\right) \\ Y_{\dot{r}}' = -\pi \left(\frac{T}{L}\right)^2 \cdot \left(0.67 \cdot \frac{B}{L} + 0.0033\frac{B}{T}^2\right) \\ N_{\dot{v}}' = -\pi \left(\frac{T}{L}\right)^2 \cdot \left(1.1 \cdot \frac{B}{L} + 0.041\frac{B}{T}\right) \\ N_{\dot{r}}' = -\pi \left(\frac{T}{L}\right)^2 \cdot \left(\frac{1}{12} + 0.017\frac{C_B B}{T} - 0.33\frac{B}{L}\right) \end{cases} \tag{C.18}$$

The expressions have a transformation described in C.19 according to abs ABS (2017):

$$\begin{cases} Y_{\dot{v}}' = \frac{Y_{\dot{v}}}{0.5\rho L^3}; Y_{\dot{r}}' = \frac{Y_{\dot{v}}}{0.5\rho L^4}; \\ N_{\dot{v}}' = \frac{Y_{\dot{v}}}{0.5\rho L^4};\ N_{\dot{r}}' = \frac{N_{\dot{r}}}{0.5\rho L^5} \end{cases} \tag{C.19}$$

NTNU – Ålesund
Norwegian University of Science and Technology

Date 18/09/2020
Department of Ocean Operations and Civil Engineering

Web Platform Architecture for Digital Twin in Ship Operation

Introduction

The reduced costs for data storage and sensors lead to an unprecedented explosion of information which helps decision makers to choose more profitable solutions with a clear perspective. The marketing sector was the first able to dig into the data gold, providing social medial web platforms to analyze the customers' behavior and apply statistics experiments to validate the models. Despite a few good solutions that start to rise, such as the veracity platform from DNV, the industrial sector still has a considerable unexplored horizon to use data in a more efficient way in order to minimize costs in the production and operation. One way to achieve this goal is by incorporating user friendly web platforms that permit simulation and data visualization.

Motivation

In this context, comes to maritime industry the concept of Digital Twin Ship, which aims to create a digital reproduction of the floating system in operation by applying the approach of the Internet of Things towards the shipping industry.

A web platform to support Digital Twin Ship must ideally present the acquired data information concisely for each specialist field and contrast it with the theoretical models in order to validate the sensors and systems. Also, those platforms ought to allow the user to create hypothetical scenarios to test the system behavior in a certain condition through a simulation.

Scope

This project aims to construct a web platform for presenting data and simulating operations scenarios. The platform must be able to exchange the essential information with a database and present the data in a visual way depending on the knowledge context, allowing the implementation of scenarios. The data usage life cycle can be summarized in the Figure 1, the activities of an user must be ordered in the phases of acquire data from a database, process data and show it contextualized by user, validate by the physical models, apply an study case to investigate possible fails or improvements, analyze the simulation output looking for meaningful insights.

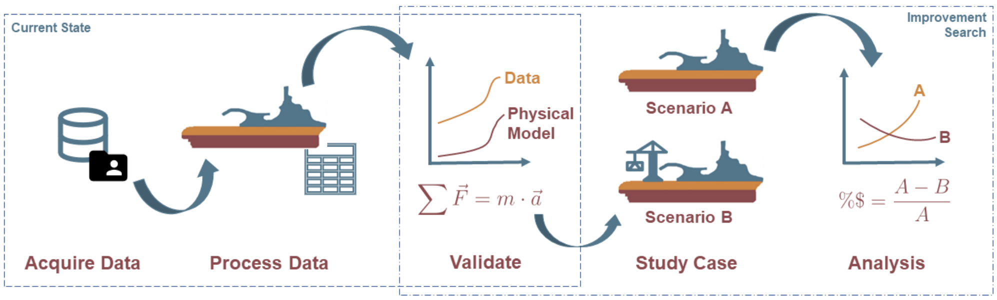


Figure 1 - Proposal of data usage life cycle.

The platform will be treated as a complex system, containing elements from different areas such as the IT tools to be used (database, visual platforms, libraries), physical models (stability, maneuverability and sea keeping) and knowledge areas (management, structural analysis, and navigation), therefore a systems engineering approach will be used to structure the information and define the life cycle of data use.

Objectives

The objectives of this project are introduced bellow:

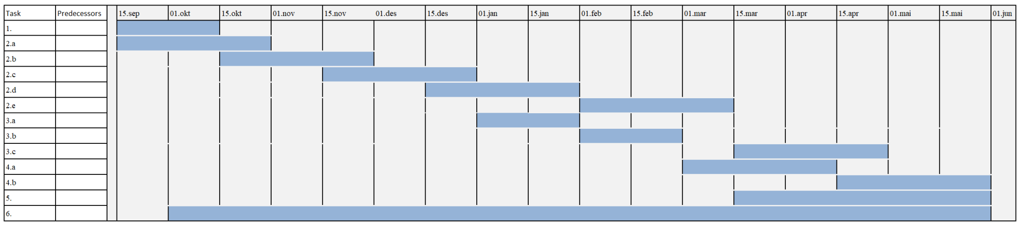
- Create a web-platform capable of reading from a third source database the relevant information of the ship;
- Expose concisely the information read depending on the considered user;
- Allow scenario creation and virtual testing;
- Propose or apply a taxonomy compatible with the platform and ship operation;
- Structure the different components using a systems engineering perspective;
- Evaluate the simulated scenarios according to the collected data and literature review.

Milestones

Tasks:

1. Literature review:
   a) Taxonomies for marine and engineering systems;
   b) Simulation in engineering and marine operations;
   c) Systems engineering;
2. Methodology:
   a) Definition of modules and user interface;
   b) Incorporate pre-existing physical modules constructed in Vessel.js to the web platform;
   c) Incorporate visualization graphics for specific areas;
   d) Create new physical modules;
   e) Allow a modification on the ship and contrast the information with the current scenario;
3. Case Studies:
   a) Import existing data and show it in the context of the web platform;
   b) Simulate the current state based on the physical modules;
   c) Create a modification and simulate a specific scenario into different physical modules, contrasting the current behavior with the expected one;
4. Analysis and Evaluation:
   a) Evaluate the current state based on the collected data and the literature;
   b) Evaluate the scenario expectation with the current data, showing the trade-offs about the modification;
5. Discussion of the results
6. Write report

Schedule



The work scope may prove to be different than initially anticipated. Subject to approval from the supervisor, topics may be added or deleted from the list above or reduce in extent.

The thesis shall be written as a research report, following the template given in Inspera. During preparation of the text, the candidate should make efforts to create a well-arranged and well-written report. To ease the evaluation of the thesis, it is important to cross-reference text, tables and figures. For evaluation of the work a thorough discussion of results is needed. Discussion of research method, validation and generalization of results is also appreciated.

The thesis shall be submitted in electronic version according to standard procedures (.PDF or .ZIP files). Instructions are found on the NTNU website (Inspera) and on Blackboard. In addition to the specified tasks, an A3 poster should be prepared and delivered together with this proposal, and a conference paper will be handled at the end of the research.

After finalizing and delivering the thesis, it must be sent a copy to the supervisor(s).

Deliveries:
Preliminary Thesis (XXth March)
Final Thesis + Article draft (XXth June)

[illegible]

Felipe Ferrari de Oliveira
Master Student - Ship Design
+47 462 73 655
felipe.oliveira@ntnu.no

Henrique Murilo Gaspar
Supervisor - IHB

Ícaro Aragão Fonseca
Co supervisor - IHB
icaro.a.fonseca@ntnu.no

Felipe Ferrari de Oliveira

NTNU
Norwegian University of
Science and Technology