



NTNU  
Norwegian University of Science and Technology  
Department of Marine Technology

MSc Master Thesis

---

**Configuration Tool for Conceptual Design  
of an Offshore Supply Vessel using  
3D Modular Building Blocks**

---

by

Tommy Torgersen

Spring 2009

Advisor

Stein Ove Erikstad

Address:  
NTNU  
Department of Marine Technology  
N-7491 Trondheim

Location  
Marinteknisk Senter  
O. Nielsens vei 10

Tel. +47 73 595501  
Fax +47 73 595697



# MSc Master Thesis

Stud. techn. Tommy Torgersen

## “Configuration Tool for Conceptual Design of an Offshore Supply Vessel Using 3D Modular Building Blocks”

Spring 2009

### **Background**

A key research areas within Marine Systems at NTNU is the development of efficient design tools and methodologies to be used in the tendering and conceptual design of complex marine systems, such as ships and offshore structures. One particular area is the efficient development of customized designs based on the configuration of components from an existing library or product platform. This process should be driven forward both by the human designer using an efficient form *and* 3D based user interface to select, scale and arrange components into a synthesized solution, as well as applying rules to propose, effectuate and verify the solution along the way.

### **Overall aim and focus**

The overall aim of the thesis is to develop a first prototype, with corresponding product models that may serve as a basis for a 3D modular building block approach in a configuration-based conceptual design process. The case domain should be OSVs, but to the extent possible the conclusions made should be made general.

### **Scope and main activities**

The candidate should presumably cover the following main points:

1. *Describe a relevant case for the conceptual design of a Platform Supply Vessel or similar, based on an existing design. Identify (high level) main modules both from a geometry and systems point-of-view.*

2. *Discuss the architecture/structure and important features for a PSV design configuration application.*
3. *Investigate the Microsoft Xna framework and how this can be used for the 3D visualization.*
4. *Investigate the arrangement, scaling and selection process for some of the parts and systems.*
5. *Discuss representation models for both the product (ship) to be designed, and the library of modules to select from, in the configuration application.*
6. *Draw a general conclusion on the applicability of this approach to support the conceptual design of marine systems such as PSV, and discuss to what extent these conclusions are also valid for other marine systems.*


### **Modus operandi**

A prototype for an external configuration application should be developed in Visual Studio using C#, since this is preferable both for the BRIX Rule Framework integration and with respect to a possible collaboration with DNV during the project.

At NTNU, Professor Stein Ove Erikstad will be the responsible supervisor for the project.

The project is within the topic area of the KMB R&D project SHIP-4C, and is thus eligible for travelling grants from this project.

The work shall follow the guidelines given by NTNU for the MSc Thesis work.



Stein Ove Erikstad  
Professor /Supervisor at NTNU

# Preface

This project was conducted as the master thesis of my MSc. degree, at the Norwegian University of Science and Technology. I am studying I&IKT, a combination of computer science and marine technology, and have tried to combine both fields in this thesis.

Based on the work done in my previous project [Torgersen, 2008], I have now performed further studies regarding the process of parts and systems selection, as well as arrangement visualization for the configuration of PSVs. An improved version of the first configuration application prototype has been developed. This time with integrated 2D and 3D graphics.

I have had some challenges working with this thesis. One constantly recurring problem was good ideas that seemed valid on paper and in theory, but turned out to be much more complex, and even impossible when it came to implementation. As a result of this I have used much time in this project to verify, that the content presented in this report actually is possible to implement.

I would like to thank my advisor professor Stein Ove Erikstad for all the help I have received during this project. I have also discussed different aspects of my work with the students Eskild Thereby, Magnus Porsmo Stoveland and Janina Therese Meyer, which has been very helpful. They have been working on projects

within some of the same subjects.

Trondheim, June 9, 2009

A handwritten signature in blue ink, consisting of two distinct parts. The first part is a large, sweeping loop that ends with the name 'Tommy' written in a cursive style. The second part is another large, sweeping loop that ends with the name 'Torgersen' written in a cursive style.

---

Tommy Torgersen

# Summary

The individual offshore supply ships are complex and often designed to perform some specific tasks. This makes it complicated to reuse the previous designs. Developing these kinds of vessels require an experienced design team and a lot of time. To help with this process, various tools have been made to assist the engineers to perform their job faster and more efficient. These systems are made in order to try and utilize some of the design knowledge and reuse this in new projects.

In this thesis I have been looking into how a prototype for PSV configuring application could be made. Earlier design knowledge are extracted, to form a basis for selection of parts and system for the vessel being design. Subsets of different components from the database, to select form, are generated by the program. These lists contain parts and systems that are ranked based on rules in the application and requirements inserted by the user. Components with scores that do not meet these criteria are removed.

The graphics is made based on Microsoft Xna. I chose this framework because it was easy to use, had a lot of helpful features and could import models from a lot of different applications. Models and even entire scenes can be designed in external applications like 3D Studio Max, and then be inserted into the configuration application. This is done by the .FBX file format.

The prototype is developed with a four layered architecture, to group coherent code and increase modifiability and maintainabil-

ity. The user interacts with the top layer. This presentation layer contains the configuration panels and a 3D visualization window based on Xna. The business layer beneath performs calculations based on the user input, like determining the size and number of cargo tanks based on cargo capacity. The third layer defines the domain objects, like Tank, Cargo, Ship and Cabin. These are stored and retrieved from the database with queries from the access layer at the bottom.



# Contents

<b>I Problem formulation</b>	<b>III</b>
<b>II Preface</b>	<b>V</b>
<b>III Summary</b>	<b>VII</b>
<b>IV Contents</b>	<b>IX</b>
<b>V List of Figures</b>	<b>XII</b>
<b>VI List of Tables</b>	<b>XVI</b>
<b>VII Abbreviations</b>	<b>XVIII</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Existing configuration systems</b>	<b>5</b>
2.1 Quaestor and Rhinoceros . . . . .	5
2.2 Paramarine . . . . .	7
<b>3 Platform supply vessel</b>	<b>9</b>

3.1 Overall design . . . . .	10
3.2 Requirements . . . . .	10
3.3 Cargo and Tanks . . . . .	11
3.4 Machinery . . . . .	15
3.5 Accommodation . . . . .	15
3.6 Hull . . . . .	17
<b>4 The configuration prototype</b>	<b>19</b>
4.1 General information . . . . .	20
4.2 Customer requirement . . . . .	21
4.2.1 KPI . . . . .	21
4.2.2 Main Dimensions . . . . .	22
4.2.3 Cargo . . . . .	23
4.2.4 Classification/Flag . . . . .	23
4.2.5 Machinery . . . . .	24
4.2.6 Accommodation . . . . .	25
4.3 Selection of parts and systems . . . . .	26
4.3.1 Selection of tanks . . . . .	26
4.4 3D Arrangement . . . . .	29
<b>5 Ranking and arrangement</b>	<b>31</b>
5.1 Tank . . . . .	32
5.1.1 Ranking . . . . .	32
5.1.2 Arrangement . . . . .	33

---

<b>6 Software architecture</b>	<b>35</b>
6.1 Presentation layer . . . . .	36
6.2 Business layer (BL) . . . . .	36
6.2.1 Management classes . . . . .	37
6.3 Domain . . . . .	39
6.4 Data Access layer (DAL) . . . . .	41
<b>7 Xna</b>	<b>43</b>
7.1 Xna -framework . . . . .	44
7.2 The class structure of Game and GameComponents . . . . .	45
7.2.1 The default methods . . . . .	46
7.2.2 The execution sequence . . . . .	47
7.3 Components . . . . .	48
7.4 Models . . . . .	50
7.5 Services . . . . .	51
<b>8 Conclusion and future work</b>	<b>53</b>
Bibliography . . . . .	54
<b>Appendix</b>	<b>56</b>
<b>A Accommodation drawings</b>	<b>57</b>



# List of Figures

- 1.1 Design knowledge [Ullman, 2002] . . . . . 2
  
- 2.1 Quaestor . . . . . 5
- 2.2 A ship model made in Rhinoceros . . . . . 6
- 2.3 Paramarine: Frigate . . . . . 7
- 2.4 Paramarine: Supply vessel . . . . . 8
  
- 3.1 Far Supplier: Tank specification . . . . . 12
- 3.2 Far Supplier: GA drawing . . . . . 13
- 3.3 Lady Grete: GA drawing . . . . . 14
- 3.4 Machinery layout . . . . . 15
- 3.5 Hull dimension . . . . . 17
- 3.6 Hull in Xna . . . . . 18
  
- 4.1 Prototype: Main panel . . . . . 20
- 4.2 Prototype: KPI panel . . . . . 22
- 4.3 Prototype: Main dimensions panel . . . . . 22
- 4.4 Prototype: Cargo panel . . . . . 23

4.5	Prototype: Classification panel . . . . .	24
4.6	Prototype: Machinery panel . . . . .	25
4.7	Prototype: Accommodation panel . . . . .	26
4.8	Tank selection 1 . . . . .	27
4.9	Tank selection 2 . . . . .	28
4.10	Tank selection 3 . . . . .	28
4.11	Prototype: 3D Cargo room . . . . .	29
4.12	Prototype: 2D Cargo room . . . . .	30
4.13	Prototype: 3D Accommodation . . . . .	30
5.1	Arrangement grid . . . . .	33
5.2	Lady Grete: Gid . . . . .	34
5.3	Far Supplier: Grid . . . . .	34
6.1	Architecture layout . . . . .	35
6.2	BL:Flowchart - Cargo tanks . . . . .	37
6.3	Management classes view . . . . .	38
6.4	Doamin: Systems . . . . .	39
6.5	Domain: Hull . . . . .	39
6.6	Domain: Machinery . . . . .	40
6.7	Domain classes . . . . .	40
7.1	Xna execution sequence . . . . .	48
7.2	Xna: Components layout . . . . .	49
7.3	Bridge: Box . . . . .	51

---

7.4 Bridge: Outfitted . . . . .	51
A.1 Vessel side . . . . .	57
A.2 Bridge . . . . .	58
A.3 Officer deck . . . . .	59
A.4 Upper Forecastle deck . . . . .	60
A.5 Forecastle deck . . . . .	60
A.6 Main deck . . . . .	61





# List of Tables

- 3.1 Cargo types . . . . . 11
- 3.2 Accommodation . . . . . 16
- 4.1 KPI rules . . . . . 21
- 7.1 2D/3D Software . . . . . 51



# Abbreviations

API	-	Application Programming Interface
BL	-	Business Logic
B	-	Beam
BLL	-	Business Logic Layer
CLR	-	Common Runtime Language
CPU	-	Central Processing Unit
DAL	-	Data Access Layer
GA	-	General Arrangement
GPU	-	Graphics Processing Unit
GUI	-	Graphical User Interface
Loa	-	Length overall
Lpp	-	Length between perpendiculars
Lwl	-	Length of waterline
PSV	-	Platform Supply Vessel
UI	-	User Interface
WPF	-	Windows Presentation Foundation

# Chapter 1

## Introduction

The financial situation today, as we have experienced, has caused an unfortunate ripple effect and evolved into events affecting the global market. This development has also contributed to reduced workload for many design companies, and perhaps less profit on each job they actually get. It is probably a long time since the possibility of getting new-building contracts for the shipyards has been this low. A result of this is a time consuming task of producing a large number tender invitation, to get the design and building contracts available.

Offshore supply vessels are considered complex, and with a large degree of customization. To design this kind of vessels, the designers and engineers involved are faced with a challenging task. To assist them in this process there has been made various programs, tools and applications to make their job faster and more efficient. Several of them utilize a combination of numerical analysis and simulations, and some combine this with 3D visualization. While many of these programs have different focus, and are built to assist in different aspects of the job, they all have one thing in common, trying to make you a better and more productive employee.

These kinds of programs are different from more traditional design

applications like 3D design packages and CAD systems. It will be discussed in more detail later in this report, but the short version is that those programs often are used in a later design phase. Their focus is more on detailed design of individual components and systems, and finalizing them for production. Some can also do calculations on the entire structure or substructures, as well as stability analyses of the complete vessel. Most CAD systems also perform strength analysis and produce production drawings.

Figure 1.1 shows how the increasing in design knowledge and reduction of design freedom changes as the time progress in a project. From the graph you can see the benefit of using previous knowledge, and how this has a positive influence on the design process. This is because decisions made in an early phase affect the design freedom in subsequent phases.

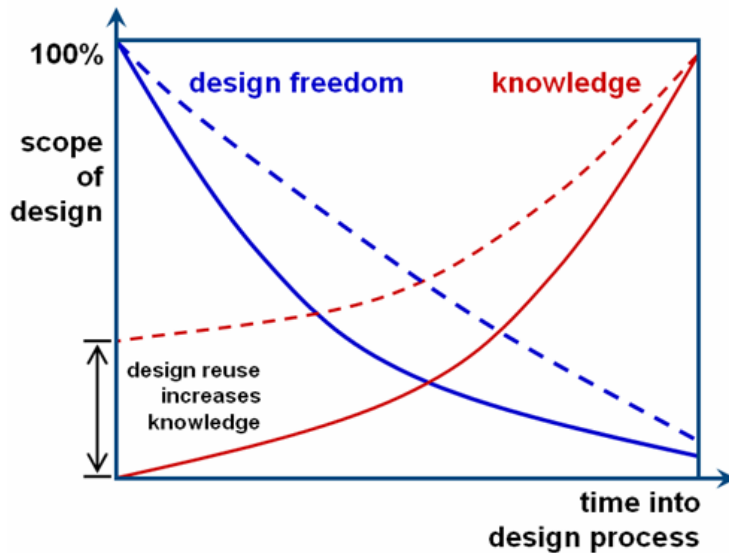


Figure 1.1: Design knowledge [Ullman, 2002]

The application type treated in this project is used in the early conceptual design phase, and tries to take advantage of the knowledge and experience from previous designs. These programs are

usually not off the-shelf packages, and many of these are tailor made software, designed to perform some specific tasks for a given company. Many of these systems are complex packages and rely on skilled and experienced users to give satisfying results.

I have made a prototype application for use in these early stages of the planning and design phase, trying to capture some of this design knowledge. The main focus was to demonstrate how to make a program that is easy to use and also showed fast results. The implementation of user-friendly GUI and understandable graphic visualization was used to achieve this. I decided to try the Microsoft Xna Framework for the visual representation in this project, to see if it could be used in an application like this. One of the advantages from this approach is all the "out of the box" functionality the framework provides. This makes it very easy to create a graphical representation of the desired domain.

The prototype also enables an iterative design approach. When using it, I do not have to provide all the input at once. I can start filling out the information I have at the time, and then the program will use statistical information from the ships stored in a database to fill out the blanks. With an extensive collection of previous ships and design information, the application then have a good possibility for providing satisfying solutions, for selecting different parts and systems.



# Chapter 2

## Existing configuration systems

### 2.1 Quaestor and Rhinoceros

Bart van Oers et al. [2008] have studied how to combine knowledge-base systems with computer aided design packages. Some of the programs they have used are the knowledge system Quaestor and CAD tool Rhinoceros where they have had good results.

**Quaestor** is a knowledge based system used to assist in the design of ships and other products. The system uses an enhanced Newton-Raphson solver to combine design relation from its internal knowledge-base with the calculated models. The models are based on the user's requirements throughout an iterative process, where he has select parameters and goals based on relations and

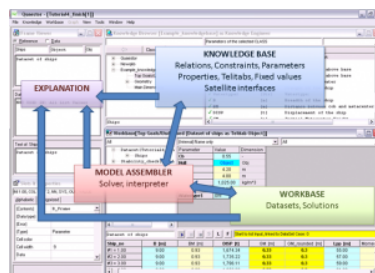


Figure 2.1: Quaestor



constraints from the knowledge-base.

In addition to these constraints, relations and parameters defined in the program, the system also has the ability to connect to external application and use data and relations defined in other databases or spreadsheets.

**Rhinoceros** is a powerful general-purpose CAD package, with a low-price compared to similar applications. Figure 2.2 shows how the program creates models based on curves, surfaces and solids defined by Non-Uniform Rational B-Splines. NURBS is a mathematical model for representing 3D geometry and are commonly used in computer graphics because of its flexibility and accuracy. The Rhinoceros package also includes various analysis tools to establish properties for the curves, areas and volumes, and functions for assisting the position of items inside the hull.

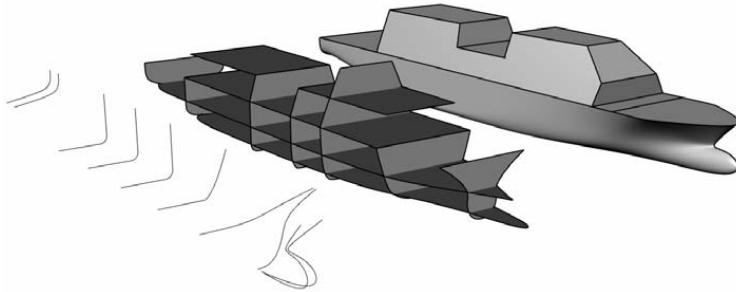


Figure 2.2: A ship model made in Rhinoceros

**The combination** of Quaestor and Rhinoceros makes it a more flexible system than two separate applications. The programs linked together make a system where the user provides requirements, the Rhinoceros part draws geometry and Quaestor positions it. Because the user can choose between different properties and relations, the system enables a rapid re-configuration of the design. When the application has made a design solution it is

possible to manually change the position of a component, this adjustment is then reanalyzed by Quaestor.

## 2.2 Paramarine

Paramarine is an application based on the principle of Building block design. David Andrews is one of the key persons studying methodology at the University College of London (UCL). Building blocks are used instead of the more traditional work breakdown structure, because this design process makes it easier to find possible new ways of arranging and positioning parts and systems. The blocks also contain information like weight and center of gravity. [M Bole, C Forrest, 2005]

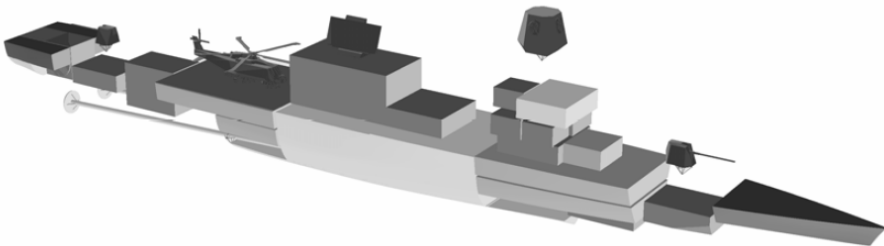


Figure 2.3: Paramarine: Frigate

David Andrews has done a lot of research for the British royal navy, and figure 2.3 shows the layout of a Frigate in Paramarines Building Block view. Supply ships has also been build in this program, figure 2.4 shows the initial configuration of a early stage

design of a supply ship.

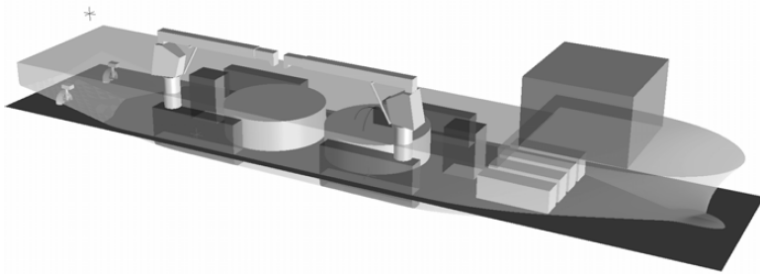


Figure 2.4: Paramarine: Supply vessel

## Chapter 3

# Platform supply vessel

A platform supply vessel (PSV) is a ship designed to support offshore platforms and installations. With its large cargo capacity and good sea keeping performance, it is able to safely operate and perform various tasks in the heavy seas offshore. A typical vessel like this can range from 20-120 m in length, and several of them are specially designed with focus on fuel efficiency and environmental concerns. A PSV can be outfitted to accomplish a particular job, or it can be used as a multipurpose vessel. [Wiki PSV, 20.4.09]

A PSV transports supplies, goods and people to and from platforms. The main deck of the vessel can be filled with containers, and below deck there is a collection of different tanks. These usually contain fluids like fuel oil, methanol, mud, water or chemicals required by the platform. Some chemicals also have to be returned to shore for proper recycling or disposal. In addition to fluids, some types of dry bulk are also supplied. [Farstad, 12.3.09]

The number of crew members accommodated on board the vessel can range from 20 persons on smaller ship and up to 50 on larger support vessels. In addition to spaces like bridge, engine rooms, living quarters, galley and mess, several of the PSVs also have

larger designated areas for work and recreation.

Some of these vessels are also equipped to support other challenges the offshore installations might have, like fire hazard and oil spillage. These ships have firefighting capabilities and oil containment and recovery equipment. [Wright International, 18.3.09]

### **3.1 Overall design**

This is a basic overview of some general PSV designs, focusing on the parts and areas that are most relevant for this report. This chapter is later used as a basis for the development of the configuration application described in chapter 4. I have chosen to focus on cargo capacity and tanks, which are important aspects of a PSV. The general accommodation and machinery composition is also introduced, in addition to a simplified version of the hull.

The reference information and ship specifications I have used as a basis throughout this report is collected from various design companies. The Farstad [12.3.09] website and the shipbroker site Wright International [18.3.09] have been especially helpful. These has also been used to fill the database with reference ships, for the statistical comparison described in chapter 5.

The vessels on Farstad's website had some good General Arrangement (GA) drawings and detailed information regarding the cargo and tank arrangement. Wright International had a lot of general information about PSVs as well as useful GA drawings of the accommodation layout.

### **3.2 Requirements**

These requirements include operational specifications from the customer. They are detailing aspects like range, speed and cargo that have to be met. Other restriction could be related to for

instance maximum draught, if the ship is operated in shallow harbors. The customer might also have requests affecting the choice of accommodation, machinery and propulsion. Other systems like navigation and communication would probably not be taken into consideration this early, as it does not affect this design stage. All these requests regarding systems, areas and their layouts would often be supplemented by internal requirements from the designers, based on their former design knowledge and experience. [Brathaug, 2008]

There are also some external rules and regulations that has to be fulfilled. The ship might be affected by regulations from international organizations like IMO, which deals with environmental issues. Classification societies like DNV, Lloyd's and Bureau Veritas also have requirements that have to be followed. Examples are safety regulation regarding crew, ship and cargo.

### 3.3 Cargo and Tanks

The ability to transport cargo is the most important mission for a supply ship. The cargo capacity then becomes one of the determining factors affecting the main dimensions of the vessel. The most frequent cargo types transported on a PSV are listed in table 3.1.

- Fuel oil
- Potable water
- Drill water
- Ballast water
- Dry Bulk
- Liquid Mud
- Base oil
- Brine
- Methanol

Table 3.1: Cargo types

Figure 3.1 gives an overview of the cargo transported on Farstad's Far Supplier. The table also contains a detailed list of each tank on this ship, its capacity and cargo content. Lists like these have also been used to fill the prototypes database, to present an example of earlier design knowledge.

	SG MP	Fuel Oil 0,830	Pot Water 1,000	Drill Water 1,000	Ballast Water 1,025	Liquid Mud 2,500	Brine 2,000	Base Oil 0,830	Metha- nol 0,791	Dry Bulk 2,400
Lower FOREPEAK	75,4		75,4							
Upper FOREPEAK	54,5		54,5							
Deep TK 1 PS	122,1		122,1							
Deep TK 1 SB	121,8		121,8							
Deep TK 9 PS	143,6		143,6							
Deep TK 9 SB	156,4		156,4							
AFT PEAK PS	174,2		174,2							
AFT PEAK SB	169,9		169,9							
DB TK 1 PS	123,6			123,6	126,6					
DB TK 1 SB	125,8			125,8	129,0					
Deep TK 2 PS	83,9			83,9	86,0					
Deep TK 2 SB	75,4			75,4						
DB TK 5 center	59,6			59,6						
STAB TK 1	213,2			213,2	218,6					
STAB TK 2	212,0			212,0	217,3					
DB WT 3 PS	226,2	192,2								
DB WT 3 SB	225,5	191,7								
DB WT 4 PS	143,4	121,9								
DB WT 4 SB	143,4	121,9								
Deep TK 7 center	144,2	122,6								
Deep TK 3 SB	25,8	21,9								
FO SERVICE TK 1-4	41,8	35,5								
DB WT 2 PS *)	132,9	112,9								
DB WT 2 SB *)	133,1	113,1								
FO SETTLING TK *)	29,2	24,8								
Deep TK 4 PS	182,9					457,3				
Deep TK 4 SB	182,9					457,3				
Deep TK 5 PS	157,5					393,8				
Deep TK 5 SB	157,5					393,8				
Deep TK 6 PS	176,0						440,1			
Deep TK 6 SB	176,0						440,1			
Deep TK 7 PS	212,7							180,8		
Deep TK 7 SB	212,7							180,8		
Deep TK 8 PS *)	110,8	94,1								
Deep TK 8 SB *)	110,8	94,1								
Cement TK 1-8	400,0									960,0
METHANOL TK PS	54,4								43,0	
METHANOL TK SB	54,4								43,0	
Weight in Tons		1247	1018	893	777	1702	880	362	86	960
Volume in m <sup>3</sup>		1467	1018	893	758	681	352	425	109	400
Volume in barrels		9228	6402	5620	4771	4283	2215	2675	685	14124

Figure 3.1: Far Supplier: Tank specification

Figures 3.2 and 3.3 shows the tank layout for two of Farsatd's ships. These and 15-20 similar GA drawings are used as a basis for the selection and arrangement of tanks in the prototype. This process is described further in chapter 5.

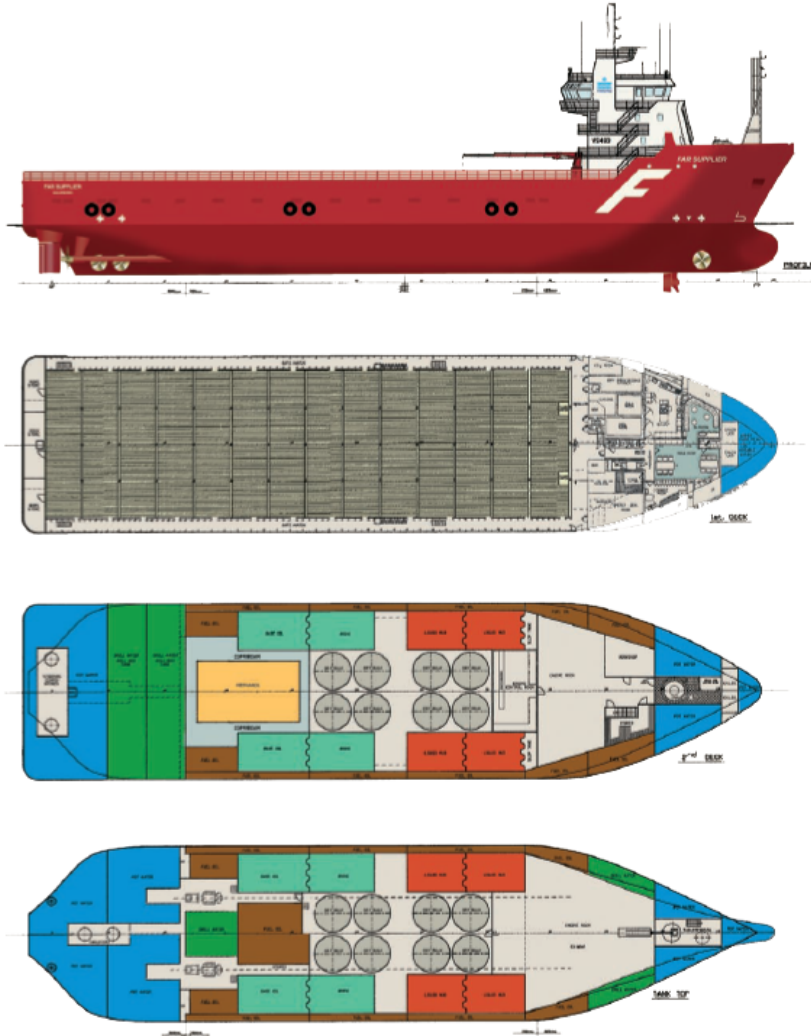


Figure 3.2: Far Supplier: GA drawing



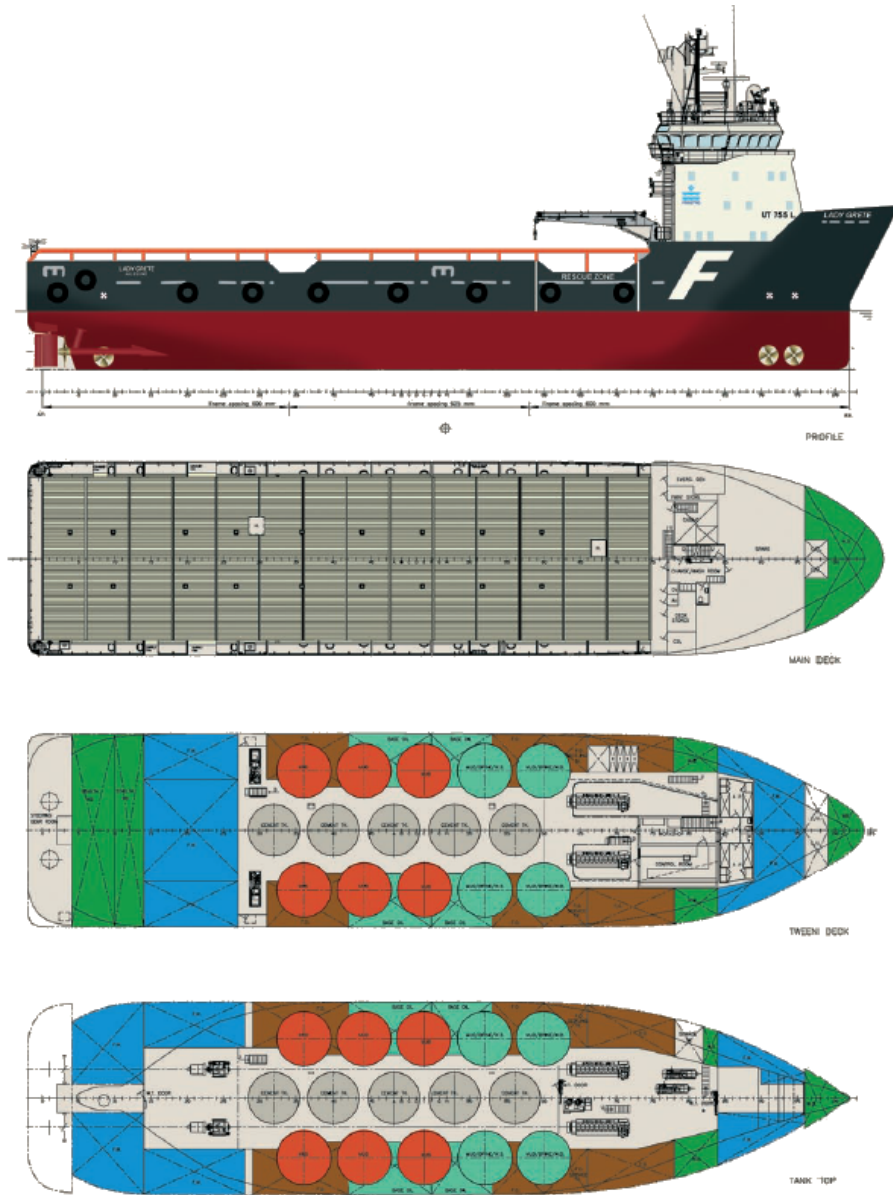


Figure 3.3: Lady Grete: GA drawing

### 3.4 Machinery

As seen from the previous drawings a PSV can have several different machinery configuration and layouts. The following decks are displayed in the GA drawings above.

- Main deck
- Tween deck / 2nd deck
- Tank top

Farstads "Far Supplier" in figure 3.2 have two shafted main diesel engines located in the aft part of the ship. "Lady Grete" is also equipped with two shafted main engines, but these are located in the forward part of the ship. The vessel in figure 3.4 on the other hand have diesel-electric propulsion. The two medium speed diesel engines power the three generators running two fixed pitch rudder propeller.

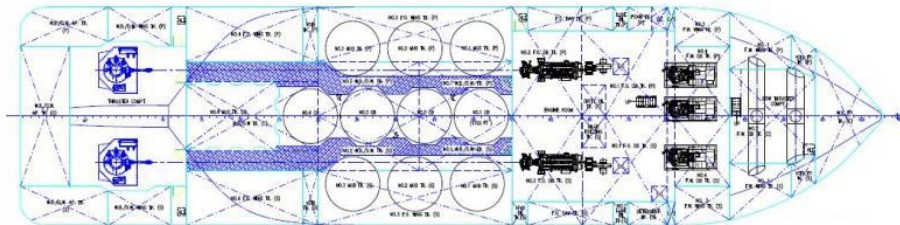


Figure 3.4: Machinery layout

### 3.5 Accommodation

As mentioned earlier a small supply vessel can have about 20 crew members on board, while a larger support vessel can accommodate 50 persons. The number of decks in the wheel house is therefore affected by the number of cabins and other rooms. Appendix A contain drawings of a 70m PSV accommodating 32 persons. Table 3.2 shows the different rooms and their location. The

number of decks is also depending on the main dimensions of the ship, which determine the available base area on main deck.

$$L_{OA} * B - DeckArea \quad (3.1)$$

Cabins		Location
Captain & Chief Eng	2x	2x Officer deck
One-Man Cabins	2x	2x Upper Forecastle deck
Two-Man Cabins	2x	1x Upper Forecastle deck, 1x Forecastle deck
Four-Man Cabins	6x	1x Upper Forecastle deck, 5x Forecastle deck
Crew Mess Room	1x	Main deck
Officer's Mess Room	1x	Main deck
Hospital	1x	Main deck
Office	1x	Upper Forecastle deck
Meeting Room	1x	Forecastle deck
Galley	1x	Main deck
Cold Store	1x	Main deck
Freezer	1x	Main deck
Landry Room	1x	Main deck

Table 3.2: Accommodation arrangement  
Crew: 32

### 3.6 Hull

Figure 3.5 show the cross-section of the tank top, tween deck and main deck. As the detailed design of the hull is not important for this thesis, I have created a simple version of this. Based on the GA drawings shown earlier, the variable  $Loa$ ,  $Lwl$ ,  $Lpp$ ,  $B$ <sup>1</sup> and some assumed constants I have made the following drawing. The reason for the simplification is to reduce the complexity of the representation in Xna.

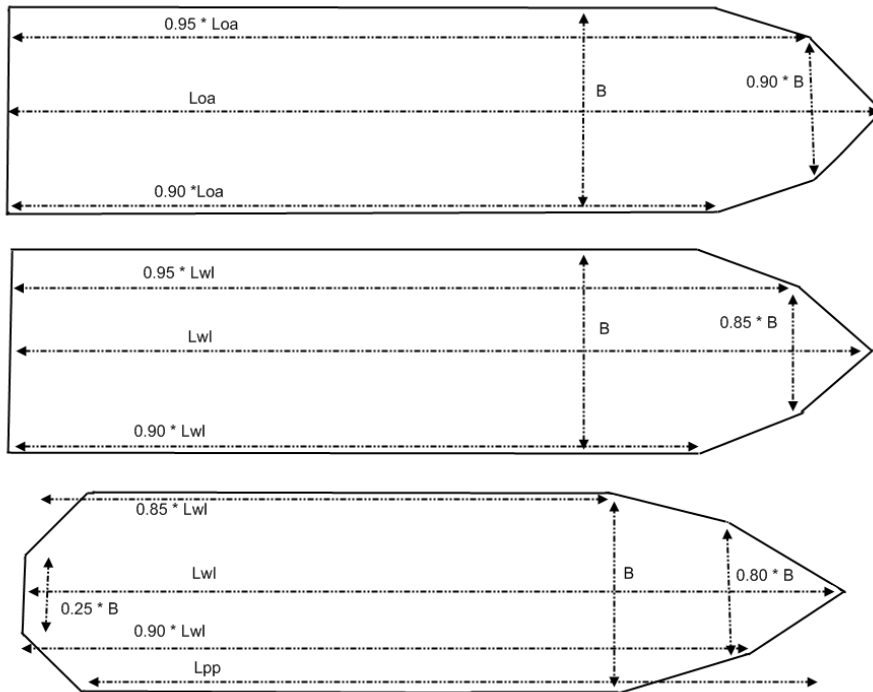


Figure 3.5: Hull dimension

<sup>1</sup> $Loa$  (length overall),  $Lwl$  (length of waterline),  $Lpp$  (length between perpendiculars),  $B$  (beam)

Figure 3.6 shows the simplifications done in order to make the 3D hull. I will not go into details on how graphics is made in Xna, but everything is made by triangles. I have used seven to nine points/corners on each deck, this decrease the complexity of the model significantly. Even with as few as 25 point I still get a satisfying model of the hull. Chapter 7 will describe how this model could be designed in another program instead of creating it from scratch in Xna.

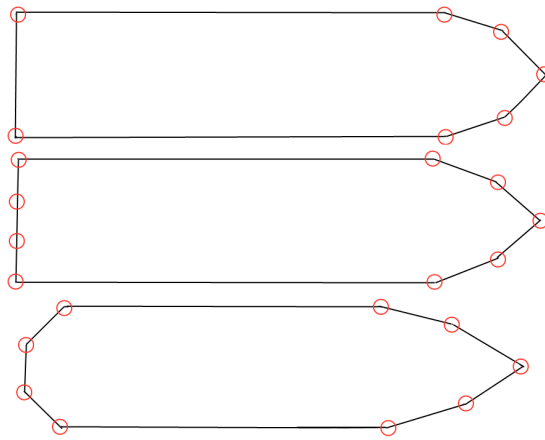


Figure 3.6: Hull in Xna

## **Chapter 4**

# **The configuration prototype**

I started making a prototype for a PSV configuration tool for the project [Torgersen, 2008]. Based on this code I have now developed the program further. The external graphic solution with the connection to Autodesk Inventor from the previous version is now replaced and I have created an integrated graphic engine, based on the Xna framework. In addition to this, the functionality of the program is expanded and its internal architecture is improved.

I am now going to demonstrate how this prototype works, and go through each step, explaining the user interface and give a general overview of some of the internal procedures. I will supply some input and show how the application could generate a PSV. Later in this report I will look closer into some of the key working principles of this prototype.

I have based the design process for configuring a vessel on an iterating approach. This means that the application does not require me to insert all the information in every panel or text box on the first walkthrough. I can start to fill out all the requirements and information at hand, and based on this the program will gener-

ate different alternatives for the missing values. I will then have the opportunity to select suitable solutions generated from earlier designs. It is possible to go back and alter some of the values inserted earlier in the previous design phases. As I do modifications to the design, the application recalculates and keeps the alternative solution updated.

## 4.1 General information

The general information about the ship is inserted in the Main window. The ID displayed at the top is now given according to the ships primary key in the database, but this could easily be changed to represent a value according to a shipyard's or a design company's ID system. I have also created some fields to insert the vessels name, type and design company. The information in this view are details that do affect the configuration process. The Main window is shown in figure 4.1

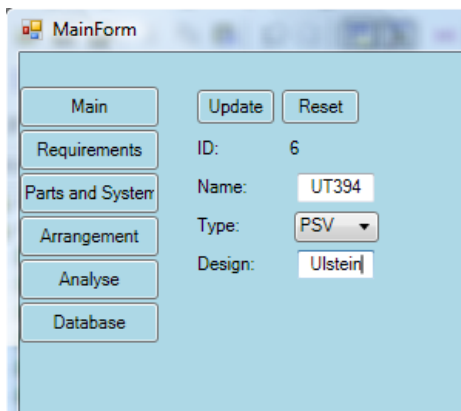


Figure 4.1: Main panel

## 4.2 Customer requirement

### 4.2.1 KPI

Key performance indicators (KPIs) are used when ranking different solutions for parts and systems which are going to be scaled and selected. This is done based on the vessels in the database. The KPIs also help reduce the list of alternative parts and systems the user can select from, and use as a base for his design. This is shown in section 4.3, where I am selecting cargo tanks. Ranking is described in chapter 5.

Figure 4.2 show the KPI panel. In the combo boxes on the left side I chose the KPIs that I think is most important. The first choice get 25 points, the next 20 p and so on. It is also possible increase the value of each KPI further; this is done in the slid panel on the right. The blue interval shows which values I can chose from.

I have implemented the following rules:

KPI	affects
Max tank volum	Cargo
Max deck space	Cargo
Min investment cost	
Speed	Machinery
Min hull size	Main dimensions
Eco-friendly	
Accommodation	Cabin

Table 4.1: KPI rules

In this example I have given the Cargo 45p and Cabin 10p, the minimum value for the blue line is then set to 45 and 10 respectively. I can now increase this value, like I have done with the Machinery, but it is not possible to give them lower score with the slider (this can only be done by adjusting the combo box values).



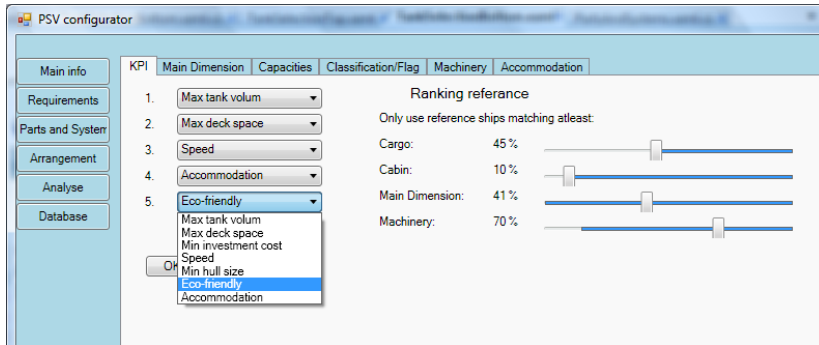


Figure 4.2: KPI panel

### 4.2.2 Main Dimensions

Restrictions for the main dimensions must sometimes be added to the design. If the ship is going to be used in a shallow harbor, all hull solutions with a draught more than x meters must be excluded from the configuration process. In the dimension panel in figure 4.3 I have inserted requirements for Loa, Beam, and Draught. The unspecified values Lpp and Depth are calculated by the application.

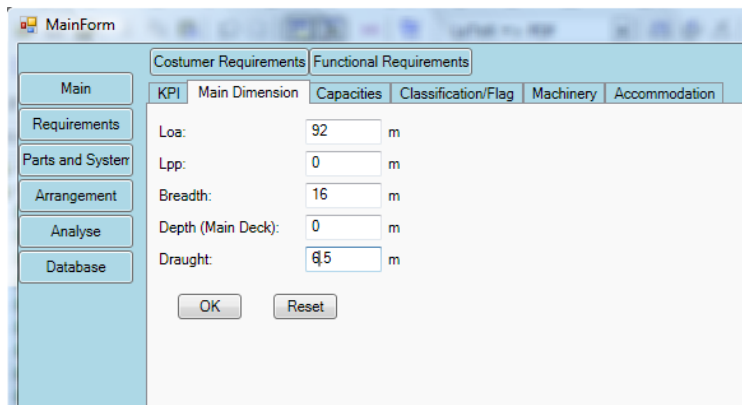


Figure 4.3: Main dimensions panel

### 4.2.3 Cargo

The total tank capacities are set in the panel in figure 4.4, it is also possible to specify the amount of cargo for each individual cargo type. I have listed up the eight most common cargo types transported with PSVs, I have also made an available TBA<sup>1</sup> field for an arbitrary cargo type (if needed). Change the name from TBA to the new cargo type, fill in the volume and specify a SG. The deck load and area can also be inserted in this window.

Cargo Type	Volume (m <sup>3</sup> )	SG
Fuel oil:	1000	0.900
Potable water:	800	1.000
Drill water:	0	1.000
Ballast water:	0	1.025
Brine:	400	2.500
Mud:	1000	2.500
Methanol:	0	0.791
Bulk Cement:	500	2.400
TBA:	0	0

Parameter	Value	Unit
Tank room		
Total tank volume:	3700	m <sup>3</sup>
Deck		
Total deck area:	700	m <sup>2</sup>
Total deck load:	0	tons

Figure 4.4: Cargo panel

### 4.2.4 Classification/Flag

The choice of class notation can have an effect on the available systems to choose from, and the design of the vessel. An example is the class notation of Clean Design, which stat that there has to be a void between the hull side and the cargo. This will reduce the amount of cargo the vessel can transport.

<sup>1</sup>TBA: To Be Assigned

Figure 4.5 show a list of the available class society to choose from. An improvement to this panel could be to list some of the important rules that this choice would have on the design of the vessel, as a reminder for the designer.

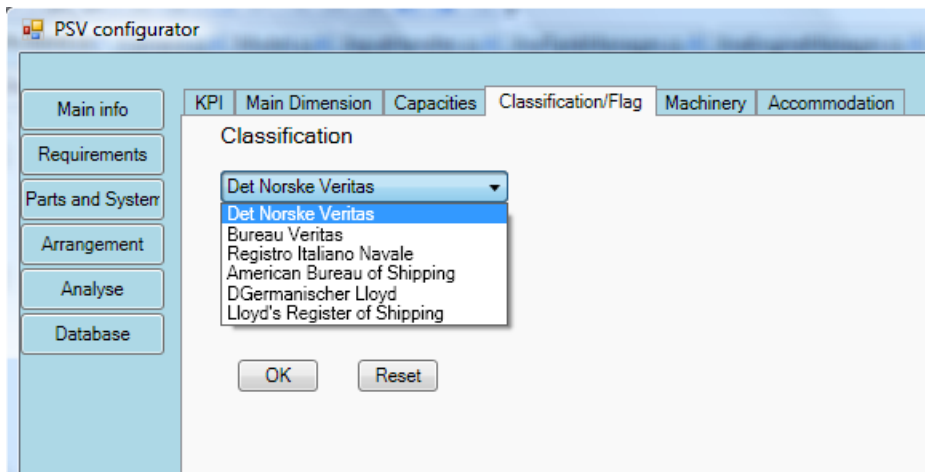


Figure 4.5: Classification/Flagg panel

#### 4.2.5 Machinery

The Machinery panel contains information about propulsion type and power requirements for the machinery. Combined with operational requirement like service speed and range, and the KPIs selected earlier (70p to machinery), this will form the basis for the selection of engines.

Like the other requirement panel, these fields are also optional. All the information that is not inserted will be calculated by the application.

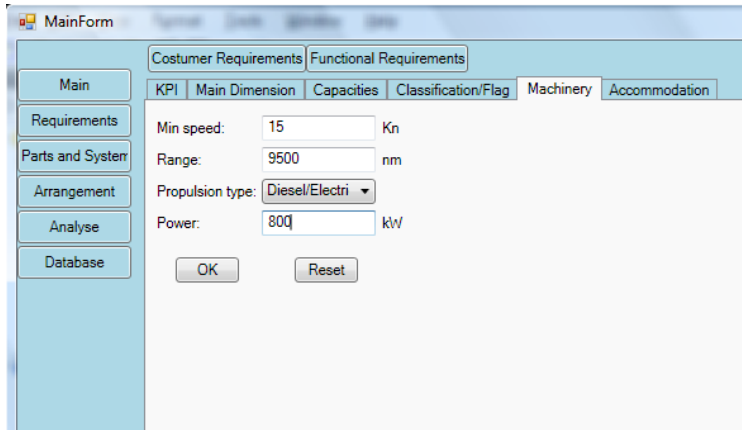


Figure 4.6: Machinery panel

#### 4.2.6 Accommodation

Like the cargo capacity, the number of crew members the vessel is going to support will have an affect on the volumetric, area and weight requirements. In the accommodation panel i figure 4.7 I can insert requirements for cabin type and size. I have also added some other rooms and specified some areas, the application will then calculate the rest.

Category	Quantity	Area (m <sup>3</sup> )
Total Crew	32	
<b>No of Cabins:</b>		
Captain	1	25
Chief Engineer	1	0
On-Man Cabin	5	12
Two-Man Cabin	4	12
Four-Man Cabin	5	0
Crew Mess Room	1	0
Officer's Mess Room	1	30
Hospital	1	0
Galley	1	0
Cold Store	1	0
Freezer	1	0
Landry Room	1	0
Office type 1	0	0
Office type 2	2	10
TBA	0	0

Figure 4.7: Accommodation panel

## 4.3 Selection of parts and systems

### 4.3.1 Selection of tanks

The tank selection window is shown in figure 4.8, 4.8 and ref-fig:appTankSelect1. This panel is divided in two parts. The left side show the values for the vessel I am configuring and the right side show a list of the ships in the database.

**Figure 4.8:** The left side is blank and the scores are 0 before the Update button is pushed. The right side shows the reference ships, their cargo and their cargo tanks. Each alternative at the right side is rated according to the similarities with the capacities I inserted in the cargo requirement panel.

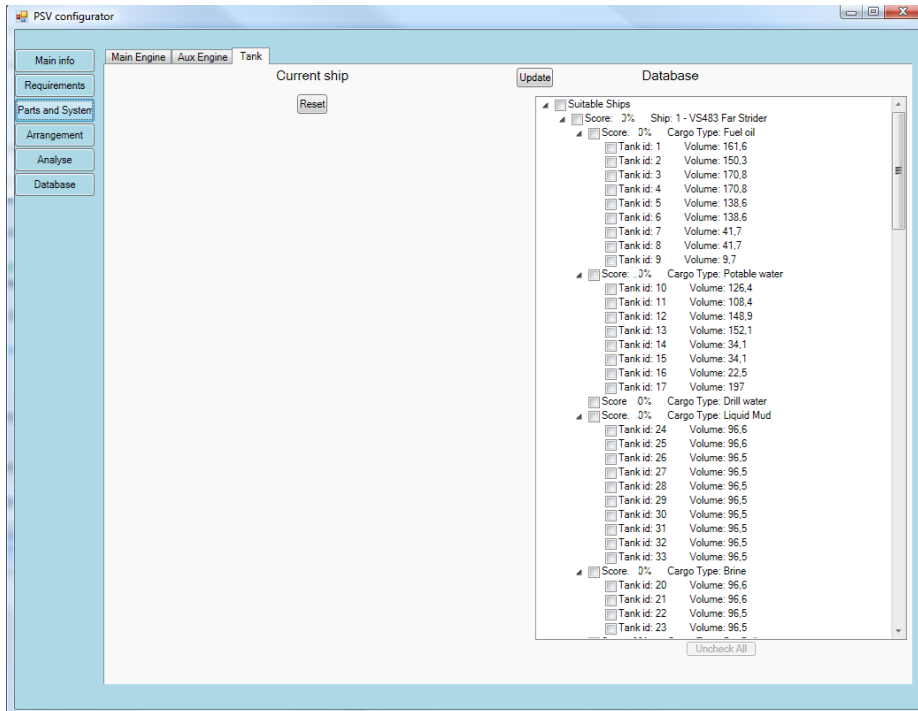


Figure 4.8: Tank selection 1

**Figure 4.9:** When the panel is updated the cargo rows appear on the left and the ships at the right side is ranked. A ship score of 100% means that the volume of each cargo type match the volume I specified in my cargo requirements. With a cargo score of 100% the volume of this cargo match my requirement. A lower score show how much the cargo or the entire ship deviate from my ship. If the score is to low I will start adding tanks from scratch, rather than selecting an alternative from the database.

**Figure 4.9:** It is not easy to show how a program works from a picture, but this is what I have done to get the results in this figure.

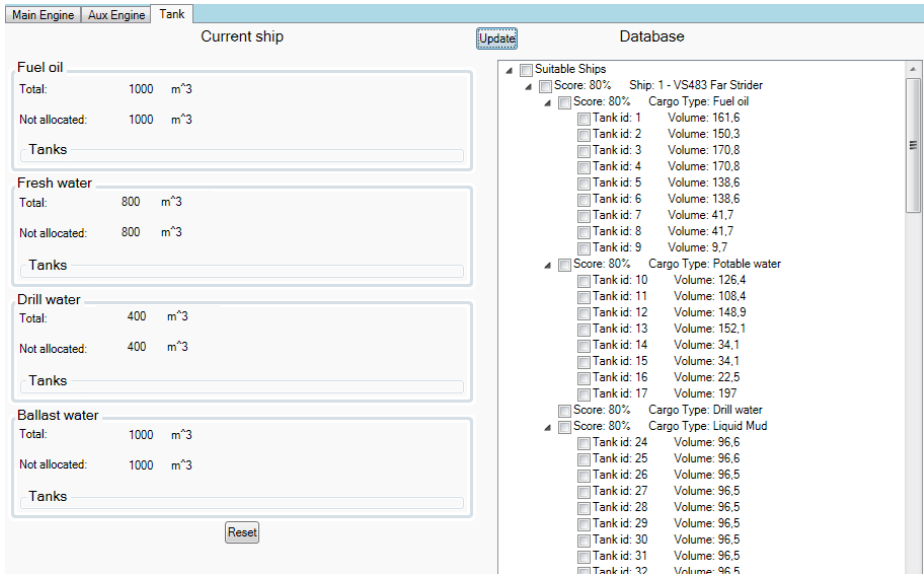


Figure 4.9: Tank selection 2

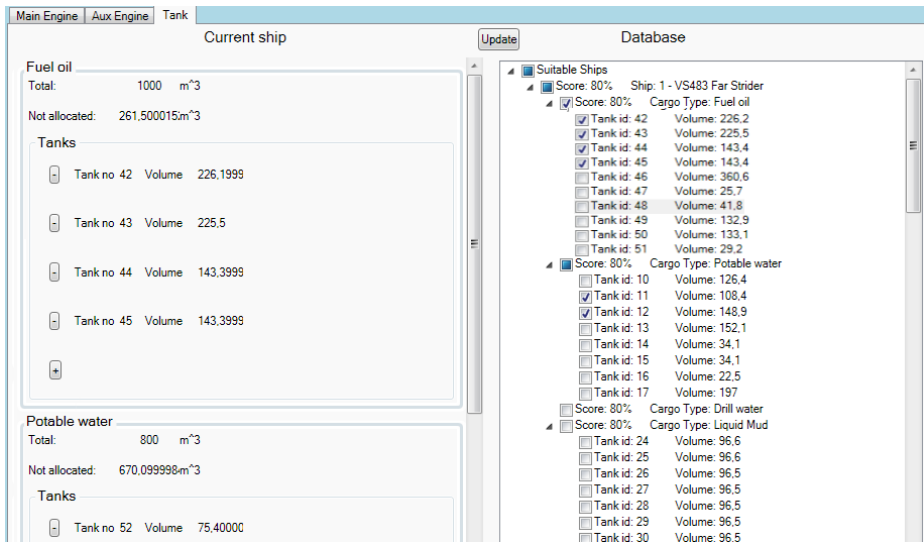


Figure 4.10: Tank selection 3

1. I selected Fuel oil on Far Strider and got a copy of all its fuel tanks.
2. I now have 10 tanks on the left side. I push the - button on six of the them, and have 4 fuel tanks left.
3. I select two single Potable water tanks from Far Strider.
4. I push the + button in my potable water view on the left, and add on tank manually (not shown in the figure)

The details of the selection and rating procedure will be explained further in chapter 5

## 4.4 3D Arrangement

Figure 4.11 show the 3D cargo room from the prototype. Four cylindrical cement tanks are placed in the middle of the ship. It is possible to select a tank, which then is colored blue. This tank can be positioned manually.

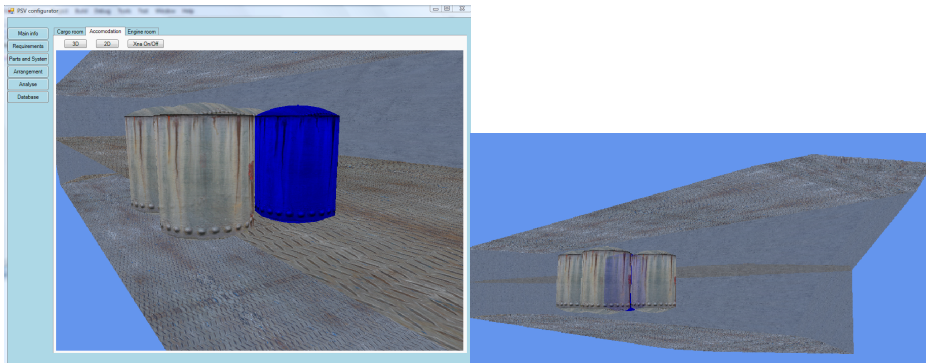


Figure 4.11: 3D Cargo room

Figure 4.12 is the 2D view of the tanks in figure 4.11. This view is inspired by Farstad's 2D drawings in chapter 3, and shows how



the tanks go from the tank top, through the tween deck and up to the main deck <sup>2</sup>.

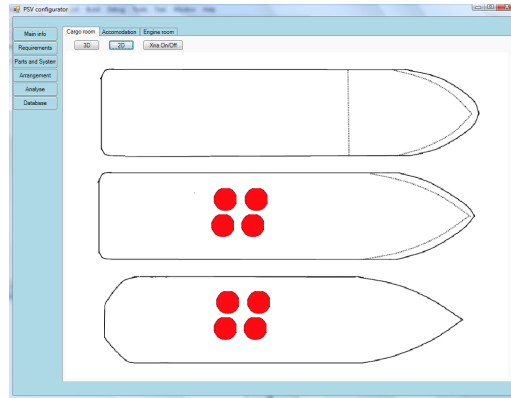


Figure 4.12: 2D Cargo room

Figure 4.13 show one of the accommodation decks, with two cubes representing rooms. When the right size and position of these rooms are determined, they can be replaced with outfitted rooms (scenes) designed in other applications. This is explained in chapter 7

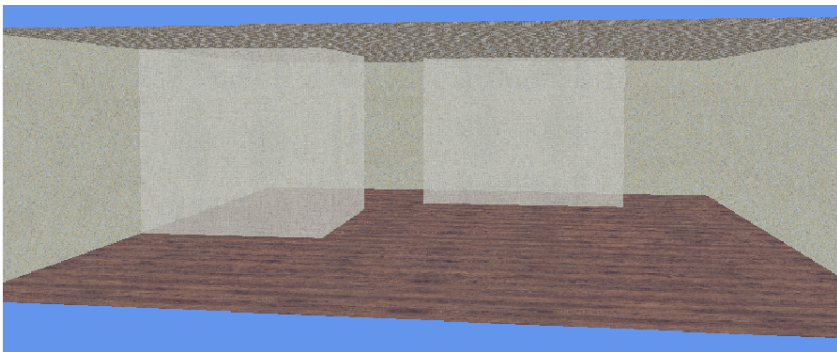


Figure 4.13: 3D Accommodation

---

<sup>2</sup>they are not shown on the top drawing because they do not go through the main deck

## **Chapter 5**

# **Ranking and arrangement**

In chapter 4 I illustrated the different functionality of the application and how the program is used. In this chapter I will show in more detail how some of those functionalities actually work and try to explain some of the routines and procedures.

One of the important aspects with this application is its ability of storing previous ship designs and information of parts and systems. As this database grows larger, it gets harder to find the correct design or design piece to reuse, I have therefore made a ranking system to help decrease the amount of comparable data used in the different selection phases.

The scores, which I will show examples of later, is constantly recalculated and updated as the user go through the different configuration steps shown in chapter 4. Since the score on the parts and systems in the database, can only be associated with the "current configuration" they are not saved with the objects in the database. The score is based on a 0 - 100% match, related to the corresponding part, system or capacity of the current ship being configured.

## 5.1 Tank

### 5.1.1 Ranking

The ranking process for the tanks is performed in the TankManager in the Business layer, which I will get back to in chapter 7. I will now show how the ships in the database are ranked so they can be used to aid in the selection of tanks for the ship being customized.

Each ship in the database, as well as their individual cargo types, get a score in the tank selection view shown in figure 4.9, as explained in chapter 4. This percentage reflects the reference ships deviation from the current ship, and help me select the best basis for a tank assembly.

First the total cargo volume is compared, to rule out the ships that are not in the desired range. To be a part of the reference list the ships has to exceed the lower limit of similarity, which was specified in KPI requirement in chapter 4.

$$\frac{|myCargoVolume - CargoVolume_i|}{myCargoVolume} = x_i \quad 0 < x_i < 1 \quad \forall i \quad (5.1)$$

The variance is calculated based on equation 5.1, and if the score in equation 5.2 is within the KPI limit the ship is added to the list. Where

$$\%Score = (1 - x_i) * 100 \quad \forall i \quad i = referenceShip \quad (5.2)$$

Each cargo type on these ships is ranked individually by the same equation above. The score for the reference ship is then recalculated this time to reflect the total cargo match. It is this new score which is displayed in the database list in the tank selection view. Equation 5.3 shows the new ship score.

$$\%Score = \sum_{j=0}^n \frac{ShipCargoScore_{ij}}{n} \quad \forall i \quad (5.3)$$

### 5.1.2 Arrangement

Chapter 3 showed some GA drawings from some of Farstad's ship designs. Based on these I have tried to extract some general rules and characteristics that could be used for tank arrangement in the prototype. By comparing the layout and location of the tanks, based on the reference ships, I decided to make a simple grid system to help with the arrangement. Figure X shows how this is done on the tank top and tween deck level.

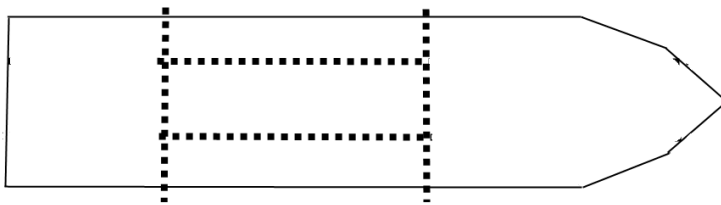


Figure 5.1: Arrangement grid

Each tank is placed in its appropriate grid. To determine which grid a tank should be placed in is done based on statistical information from earlier designs. The tanks in the database contains a position ID specifying which grid it is located in. This could be F (forward), A (aft), S (starboard side), P (port side) or C (center). The size of the grid for the ship being customized is then adjusted based on the number and dimensions of the tanks located inside it.

Statistics show that the cargo with the highest specific gravity (SG) is positioned in the center, usually dry bulk like cement. Each side contains base/fuel oil and liquid mud, and potable/drink water is located in the forward and aft part of the vessel.

Figures 5.2 and 5.3 show how this would look like on the tween deck on some of the reference ships used earlier. As you can see these grid lines nearly always corresponds to a walking path through the ship. They are also positioned differently because Far Supplier have two center row with tanks, and Lady Grete has only

on.

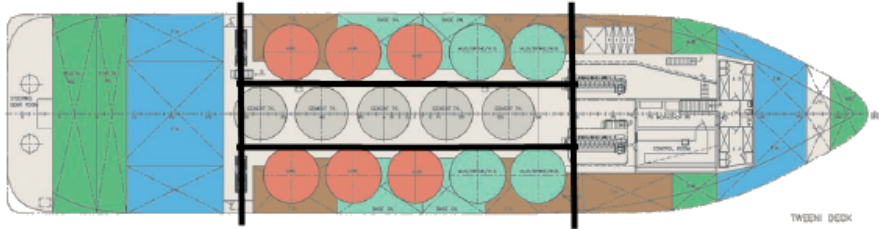


Figure 5.2: Lady Grete: Grid

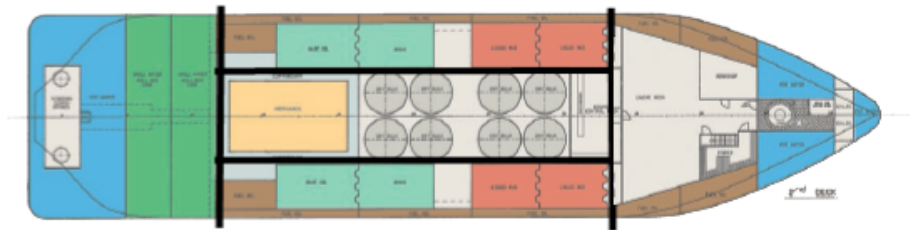


Figure 5.3: Far Supplier: Grid

## Chapter 6

# Software architecture

I have divided the application into four main parts. The top layer contains the user interface, which is the part of the program the user interacts with. This is where the input is provided and the results and visualization is displayed. Beneath this layer, the actual brain of the program is located. The Business layer receives user input and does calculations based on an interpretation of this information. The generated results are then returned to the Presentation layer. The business layer also holds the instantiated domain objects from the classes in the layer beneath. The domain layer contains a definition of all the domain classes that represent the relevant concepts of the real world. The program is based on classes like Ship, Cargo, Tank and Cabin. The bottom layer deals with data storage. The Data access layer connects the domain classes to either a local database or an external server.

The structure of each part will be discussed later in this chapter, but the main reason for this separation into layers, is to achieve high coherency within the

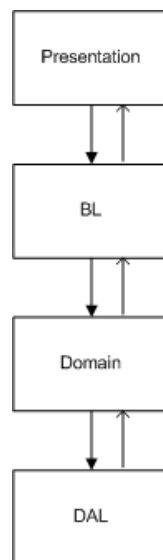


Figure 6.1:  
Architecture layout

program. This will give the advantage of increased modifiability and maintainability, which will make it easier to develop the program, as well as performing future implementations or improvements. With this benefit I can work on each individual layer of the program separately without being afraid of influencing other parts of the application.

## **6.1 Presentation layer**

The user interface consists of two main parts, the different configuration panels and the 3D representation. The configuration windows are based on the Windows Presentation Foundation. I have continued the work I started in my previous project [Torgersen, 2008], and develop this concept further. All the panels are shown in chapter 4.

I have also been looking into different approaches for the integration of 3D graphics. In project [Torgersen, 2008] I tried to use Autodesk Inventor as an external visualize. This solution turned out to be more complication than first anticipated, and due to lack of useful guidelines and a difficult API, I decided to not pursue this alternative. OpenGL was another possibility I briefly looked at in my previous project, and revisited in the start of this thesis. I compared the framework with Microsoft's Xna, and found that Xna was a better choice. As I will show in chapter 7 this framework provides a lot of "out of the box" functionality that require a small amount of code to get the project up and running. These pre-defined features I could later exchange with my own customized parts and routines.

## **6.2 Business layer (BL)**

The business layer defines the jobs the software is supposed to do. This layer contains the business logic and rules and defines

the purpose of the application. In chapter 5 I described some of the ranging and arrangement procedures, these are located in this layer.

The rules and procedures are located in different management classes. I will describe the three of them involved with the selection of cargo tanks. All these classes are made static, this way they can be used without instantiation, and having to transfer the objects around in the presentation layer. Chapter 7 describe how the Xna framework uses Services to avoid this problem.

### 6.2.1 Management classes

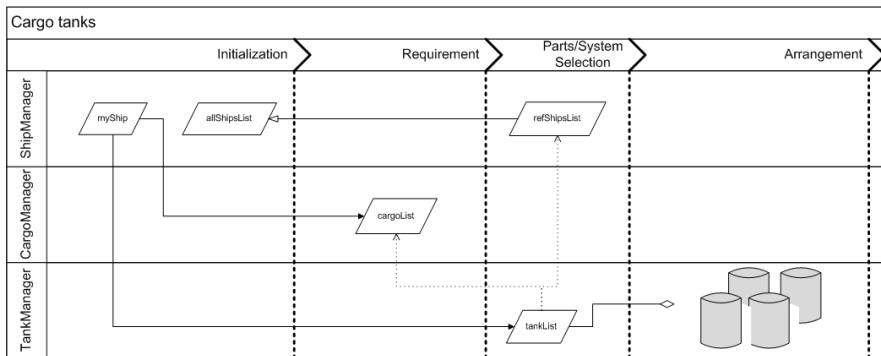


Figure 6.2: BL:Flowchart - Cargo tanks

In figure 6.2 I have made a flowchart to show what is involved in the process of selecting and arranging the cargo tanks. First the ship being configured is initialized, here represented by the object myShip. The Ship Manager takes care of this object as well as a list of all ships stored in the database.

In the Requirement phase, the ship's cargo list is generated in the CargoManager. This is based on the user input from the Requirement panel showed in chapter 4. The list is then sent to myShip.

In the Parts and System Selection phase the ships tank list is



generated in the TankManager, based on the cargo list made in the previous phase, as well as a reference ship list. This ship list is a copy of the list made in the Ship Manager, which is reduced by the ranking procedure described in chapter 5. This list is then used as a basis for the tank arrangement.

Figure 6.3 shows how the management classes is organized. The Ship Manager at the center is the main class and contains the information about the customized this. All the other classes, here represented by the management classes for Accommodation, Tank and Cargo are used to support the ship.

They support classes contains methods for calculating the number of tanks or cabins, and routines for ranking different solutions.



Figure 6.3: Management classes view

## 6.3 Domain

These figures show an overview of the domain layer. The main ship is divided into components, which is divided into subcomponents.

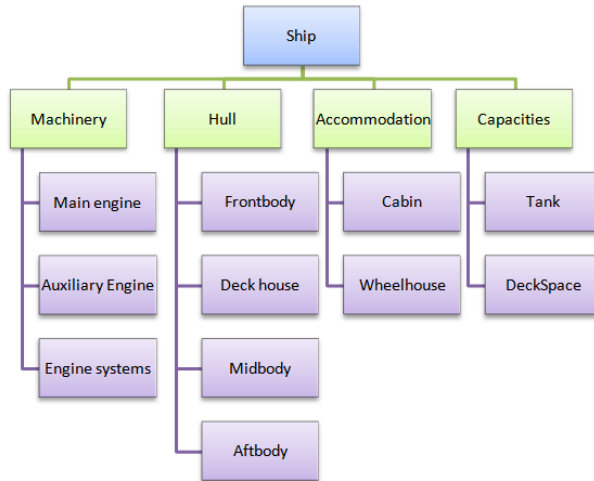


Figure 6.4: Doamin: Systems

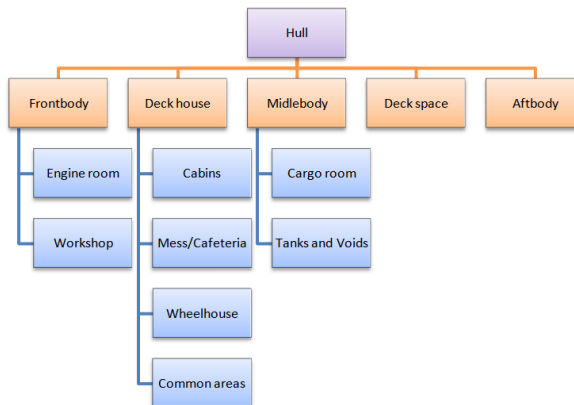


Figure 6.5: Domain: Hull

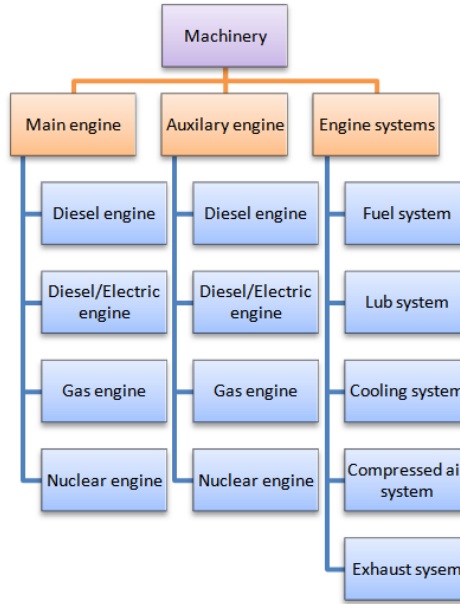


Figure 6.6: Domain: Machinery

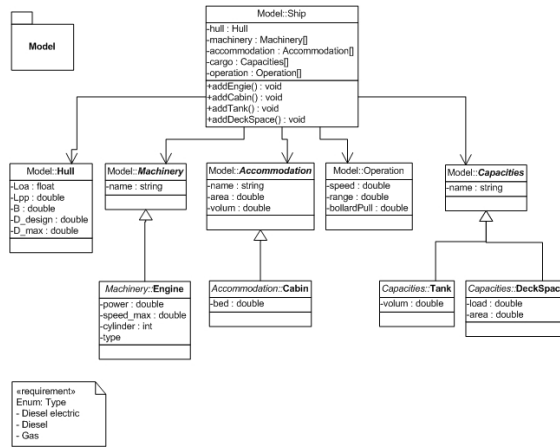


Figure 6.7: Domain classes

## 6.4 Data Access layer (DAL)

The data access layer contains routines for storing and retrieving information from the database. I have used LINQ to SQL to connect to a local database. The advantage with using LINQ is that it is easy to use and it auto-generate a lot of code. It make partial classes with properties for all the tables in the database, which saves me a lot of time when making the domain layer, because all the properties is already made.

Listing 6.1: LinQ: Domain

```
1 public class Ship
2 {
3     \\ Properties
4     public int ID { get; set; }
5     public string Name { get; set; }
6     public string Design { get; set; }
7 }
```

To utilize the advantage described above the main LINQ class <sup>1</sup> has to be placed in the domain layer. All the queries and storage procedures on the other hand is placed in the access layer as common. Like the code below. This code also shows the small amount of code needed to load a ship, and its cargo and tanks.

Listing 6.2: LinQ: Load ship

```
8     private static LocalDB db = new LocalDB("LocalDB.sdf");
9
10    public static IShip LoadShip(IShip ship)
11    {
12        ship = ((from s in db.Ship
13                where s.ID == ship.ID
14                select s).SingleOrDefault()) as IShip;
15
16        // Fill cargo list
17        var cargos = (from c in db.Cargo
18                    where c.ShipID == ship.ID
19                    select c);
```

---

<sup>1</sup>the .dbml file

```
20     foreach (Cargo c in cargos)
21     {
22         ship.CargoList.Add((ICargo)c);
23     }
24
25     // Fill tank list
26     var tanks = (from t in db.Tank
27                 where t.ShipID == ship.ID
28                 select t);
29     foreach (Tank t in tanks)
30     {
31         ship.TankList.Add((ITank)t);
32     }
33
34     return ship;
35 }
```

# Chapter 7

## Xna

As mentioned earlier in chapter 5 and 6, I have used Xna as a component in the prototype's user interface. I will now give a brief overview of some of the components and features of this framework and why I have chosen this as the basis for the 3D representation.

Xna is actually the brand for Microsoft's game-related technologies, and its primary user group is the gaming industry. Xna is used to make games for the Windows platform, Xbox game console and Mobile-based devices. The technology is built upon frameworks like DirectX and .NET, and does also contain the Xna Game Studio, which is the development environment used to program against this framework. [Wiki, Xna, 14.4.09]

One of the benefits of using Xna is good libraries and functionality to avoid writing "repetitive boilerplate code". These are sections of similar code that needs to be included in several parts of the code. It is also fairly easy to get started with a project, and to see the visual results. The code below shows the few lines that are necessary to have a model displayed on screen. Line 8 is used to load the model, and the double for-loop goes through each mesh to draw the part. If the model is solid, it is sufficient with only one for-loop. The BasicEffect is one the existing predefined features

in Xna.

### Draw a 3D model

```
1  Model myModel;
2  BasicEffect basicEffect;
3
4
5  // Loads the model into the ContentManager
6  protected override void LoadContent()
7  {
8      myModel = Content.Load<Model>("Models\\p1_wedge");
9  }
10
11 protected override void Draw(GameTime gameTime)
12 {
13     foreach (ModelMesh mesh in myModel.Meshes)
14     {
15         foreach (BasicEffect effect in mesh.Effects)
16         {
17             effect.EnableDefaultLighting();
18             effect.View = Matrix.CreateLookAt(cameraPosition, ←
19                 Vector3.Zero, Vector3.Up);
20             effect.Projection = Matrix.CreatePerspectiveFieldOfView←
21                 (MathHelper.ToRadians(45.0f), aspectRatio, 1.0f, ←
22                 10000.0f);
23         }
24         mesh.Draw();
25     }
26     base.Draw(gameTime);
27 }
```

## 7.1 Xna -framework

The Xna Framework is based on the native implementation of .NET 2.0 (Xna 2.0), .NET 3.0 (Xna 3.0) or the Compact Framework (Xbox 360). It runs on a version of the Common Language Runtime (CLR), which is optimized for graphics to provide a managed execution environment. The CLR executes under the management of a virtual machine, in contrast to unmanaged code that is executed directly by the computer's CPU. One of the benefits of using managed code in this context is enhanced performance.

---

The Xna Framework includes an extensive set of class libraries, containing classes, interfaces and value types, which are available through Xna Game Studio. They are specialized for graphic development, and created for maximum code reuse, as well as being used in many different applications, components or controls. [Wiki, Xna, 14.4.09]

The framework provides classes for keyboard and mouse input, which identifies and retrieves keystrokes, mouse clicks and positions. There are also defined standard structures for matrix, planes, points and vectors with 2,3 and 4 components, as well as support classes for converting and manipulate them. The Content Manager is another important class. It is responsible for loading and managing different objects from external files, like models, effects and textures. A model could be designed in other programs, the textures would give it color and depth and the effects will define how the light is reflected off the surface. This is combined a components Draw method and become the model you see on the screen.

## 7.2 The class structure of Game and GameComponents

Game<sup>1</sup> is the main class in an Xna project. The code below shows the default methods provided for this class or for a standard game component class. The game implements the Xna Frameworks Game interface, and a component would implement either a GameComponent or a DrawableGameComponent interface. The main difference between these two is the methods supported. The game component has an Initialize and Update method, and the drawable game component has the additional LoadContent, UnloadContent and Draw methods.

---

<sup>1</sup>The name just reflects that game designers are this frameworks primary users



## Game/Component class structure

```
1 public class Game1 : Game
2 {
3
4     public Game1()
5     {
6     }
7
8     protected override void Initialize()
9     {
10
11         base.Initialize();
12     }
13
14     protected override void LoadContent()
15     {
16     }
17
18     protected override void UnloadContent()
19     {
20     }
21
22     protected override void Update(GameTime gameTime)
23     {
24         base.Update(gameTime);
25     }
26
27     protected override void Draw(GameTime gameTime)
28     {
29         GraphicsDevice.Clear(Color.CornflowerBlue);
30
31         base.Draw(gameTime);
32     }
33
34 }
```

### 7.2.1 The default methods

**Initialize** allows the game or a component to perform any initialization it needs to do before the application starts to run. This method is often used to query for required services, like the ContentManager, a Camera or an InputHandler. It is also used to load other non-graphic content. The game class uses this method to create instances of the other components and add them to the ContentManager.

**LoadContent** this method is called once when the program starts. This is the place for loading content, especially related to graphics. This is where the models, textures and effects are loaded into the content pipeline.

**UnloadContent** is called when the application closed, and is where the content is unloaded for releasing the memory used.

**Update** is an important method. This is where the logic for updating the view is placed. This could be collision detection, code for gathering input or moving or scaling of models.

**Draw** contains the code that displays the models on screen.

### 7.2.2 The execution sequence

Figure 7.1 show how the different methods described above is executed. [Carter, 2008]

1. Xna calls the **Initialize** method in Game
2. Xna calls the **Initialize** method in Components and DrawableComponents
3. Xna calls the **LoadGraphicsContent** method in DrawableComponents
4. Xna calls the **LoadGraphicsContent** method in Game
5. Xna calls the **Update** method in Game
6. Xna calls the **Update** method in Components and DrawableComponents
7. Xna calls the **Draw** method in Game
8. Xna calls the **Draw** method in DrawableComponents
9. Steps 5 through 8 are repeated about 60 times each second

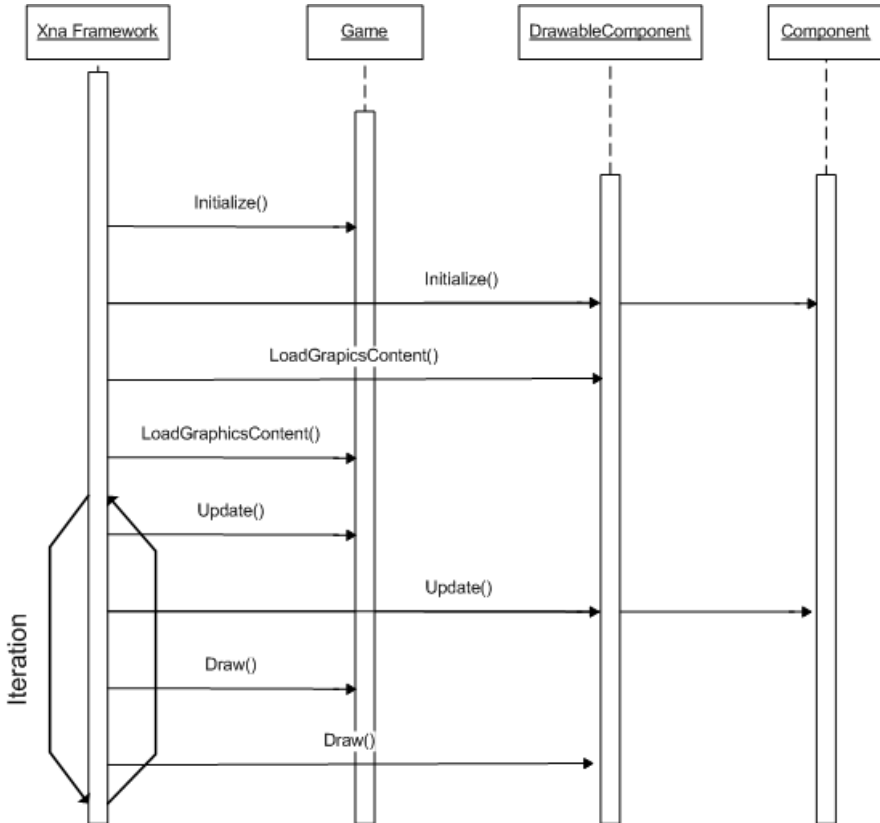


Figure 7.1: Xna execution sequence

### 7.3 Components

The ContentManager is actually a just a Dictionary and one of the features provided by the framework. This object is used to map the contents name to the actual content. This way the given content only has to be loaded once and could then be used by multiple classes. In the prototype content like this would be models like the tanks or engines and cubes representing different cabins or entire rooms.

## Components layout

As shown in the figure 7.2 I have built the Xna graphics based on components in a hierarchical structure. The class on top is Xna's main Game class which holds some game components. The components themselves hold their own components. When moving from for instance the Cargo view in the Configuration application to the Accommodation view, the main game class activates the accommodation component (AccomodationRoom), which itself activate its subcomponents (CabinManager etc.). The game then deactivates the other components (CargoRoom), which in turn deactivates its own subcomponents (TankManager etc.) down the hierarchy.

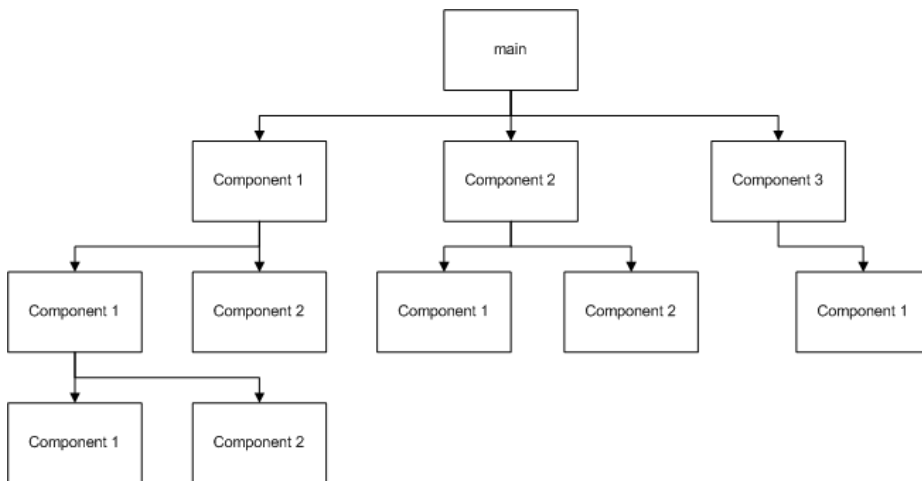


Figure 7.2: Xna: Components layout

## Activate/Deactivate components

This happens when changing from one Xna-view to another, in the arrangement window. For instance when going from the cargo room to the accommodation view. I used the Xna's DrawableCom-

ponent properties "Enable" and "Visual", which let me turn the components Update and Draw method on and off, respectively.

## 7.4 Models

One of the main advantages of using the Xna Framework, as explained in chapter 7, is how easy it is to use models created in other design application. I can make models, assemblies and GA drawings in one of several different 2D or 3D applications and then import the work into a program based on Xna, like the PSV configuration application described in this thesis. The two main formats for models used in Xna are .X and .FBX. The first one is one of Microsoft's own formats. It is also used in the development of Direct3D, the graphical part of the DirectX framework.

The FBX file format is developed by Autodesk, and is the one I have used in this project. This format is Autodesk's own preferred choice of solution for transferring data between their different products. This technology also supports platform-independent 3D data exchange, and in addition to models and assemblies it can also move entire scenes of geometry and animations. [AutoDesk, 1.5.09]

The fact that FBX can transfer complete scenes is a great advantage. This could be utilized when it comes to the accommodation models. The bridge could be modeled as a box, and when the main dimensions are set, it could be replaced with a complete room, made in for instance 3ds max.<sup>2 3</sup>

Tabel 7.1 has a list of the most popular packages for designing 2D and 3D. These are all applications that can export models or scenes to .fbx.<sup>4</sup>

---

<sup>2</sup>Figure 7.4 is not made in 3ds, but visualize the idea

<sup>3</sup><http://www.mardep.gov.hk/en/others/facilities.html>

<sup>4</sup>Some require a plug-in

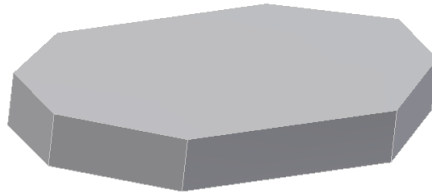


Figure 7.3: Bridge: Box



Figure 7.4: Bridge: Outfitted

Autodesk 3D Studio max	(3D design)
Autodesk Maya	(3D design)
Autodesk AutoCAD	(2D design)
SolidWorks	(3D CAD)
Solid Edge	(3D CAD)
Autodesk Inventor	(3D CAD)

Table 7.1: 2D/3D Software

## 7.5 Services

Services are one of Xna's useful features, they are used to maintain loose coupling between different objects that need to interact with each other. I will show an example using camera. I have made a general Camera class with a ICamera interface, which contains the key camera attributes (Position, Orientation, Target,

View and Projection). This is extended into a more specific FirstPersonCamera.

In the main game class I have instantiated the FirstPersonCamera into a camera object and added the camera as a game component. So far nothing special has been done. The game class itself does not use a camera, but other components and models do. The common way to solve this would be to pass the camera instance to each and every part of the program that needs it. This will be avoided with services.

So instead of passing along the camera instance I will register the FirstPersonCamera as a service. This is done by adding this to the constructor. Registered components can be requested by any object, and it does not have to know where the instance where made (the ServiceProvider).

```
1 game.Services.AddService(typeof(ICamera), this);
```

When a drawable component then wants to use the camera, I only have to add the following to its constructor.

```
1 ICamera camera = (FirstPersonCamera)game.Services.GetService(↔  
    typeof(ICamera));
```

## **Chapter 8**

# **Conclusion and future work**

Having an application where the design team can take advantage of earlier design knowledge is important. This reduces the design time, and an application might also generate alternative solution that the designers would not have come up with on their own.

In this thesis I have studied the configuration process for designing an OSV. I have looked at how I can use previous designs to assist with the selection of parts and system, based on KPI requests. I have also investigated the Xna framework, and found that this could be used to make a 3D visualization of the vessel. Compared to the Autodesk Inventor API used in the project Torgersen [2008], Xna is a far better choice.

For future development and studies I will suggest you look into the following areas to improve this prototype:

- Improving the ranking procedure. This would be needed when the database expands.
- Increase the domain model and add more functionality to the configuration process



- Adding more information in the database or connect to an external database. With more parts and system the application would provide better alternative design solutions.
- The Xna based graphics also need some improvements. And more models from for instance 3d Studio Max has to be made.

# Bibliography

- Abel Avram. Domain-Driven Design, Quickly. C4Media, 2006.
- David Andrews. A Creative Approach to Ship Architecture, 2007.
- Anthony S. Daniels et al. Development of a Hybrid Agent-Generic Algorithm Approach to Genral Arrangement, 2008.
- AutoDesk. FBX. <http://www.autodesk.com/fbx>, 1.5.09.
- Bart van Oers et al. Combining a Knowledge System with Computer-Aided Design, 2008.
- BMT Fluid Mech. BMT Fluid Mechanics, web. <http://www.bmtfm.com/?/335/222/140>, 10.5.09.
- Thomas Brathaug. Product configuration in ship design. Master's thesis, Norwegian University of Science and Technology, 2008.
- Chad Carter. Microsoft XNA Unleashed, page 525. Sams, 2008.
- D. J. Andrews. Simulation and the design building block approach in the design of ships and other complex systems, 2006.
- Farstad. Farstad, web. <http://www.farstad.no/?menu=98>, 12.3.09.
- Fredrick Hillier Gerald Liberman. Introduction to Operations Research, pages 644–658. Eighth Edition, 2005.
- Captain Vic Gibson. Supply Ship Operations. OPL, 1999.

Riemer Grotjans. Xna 2.0, Game Programmng Recipes, page 626. Apress, 2008.

Hernani L. Brinati et al. Learning Aspects of Procedures for Ship Conceptual Design Based on First Principles, 2007.

Jesper Riis Lars Hvam, Niels Henrik Mortensen. Product customization. 2008.

M Bole, C Forrest. Early Stage Integrated Parametric Ship Design. Graphical Research Corporation Ltd, UK, 2005.

Marintek. Tidligutrustning. 1996.

Paul Oldfield. Domain Modeling. Appropriate Pcess Group, 2002.

Ajit Sheno. Concurrent Engineering in the Context of FRP Boats, 2007.

Tommy Torgersen. Conceptual Ship Design using 3D Modular Building Blocks. Master's thesis, 2008.

David G Ullman. The Mechanical Design Process, page 432. McGraw-Hill Higher Education, 2002.

Wiki PSV. PSV, wikipedia. [http://en.wikipedia.org/wiki/Platform\\_supply\\_vessels](http://en.wikipedia.org/wiki/Platform_supply_vessels), 20.4.09.

Wiki, Xna. Xna wikipedia. [http://en.wikipedia.org/wiki/Microsoft\\_XNA](http://en.wikipedia.org/wiki/Microsoft_XNA), 14.4.09.

Wright International. Wright International Ltd, web. <http://www.wright-international.com>, 18.3.09.

# Appendix A

## Accommodation drawings

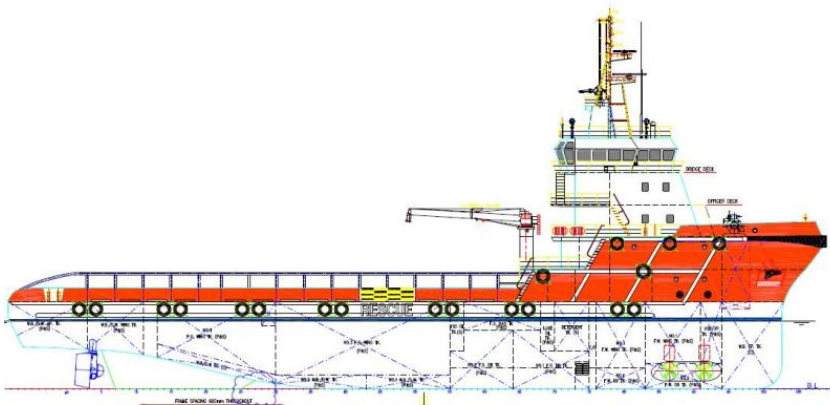
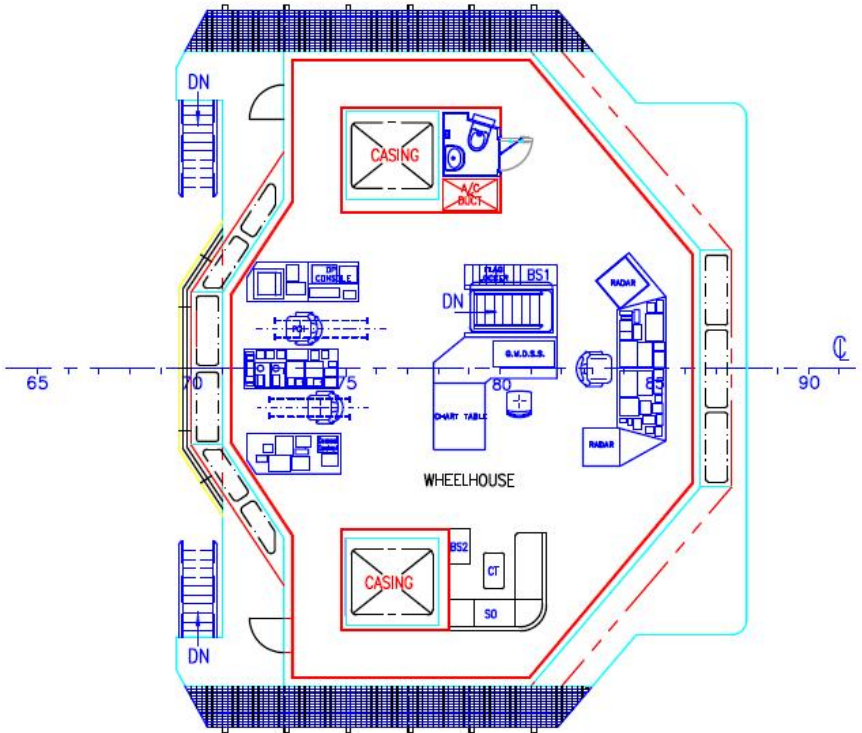
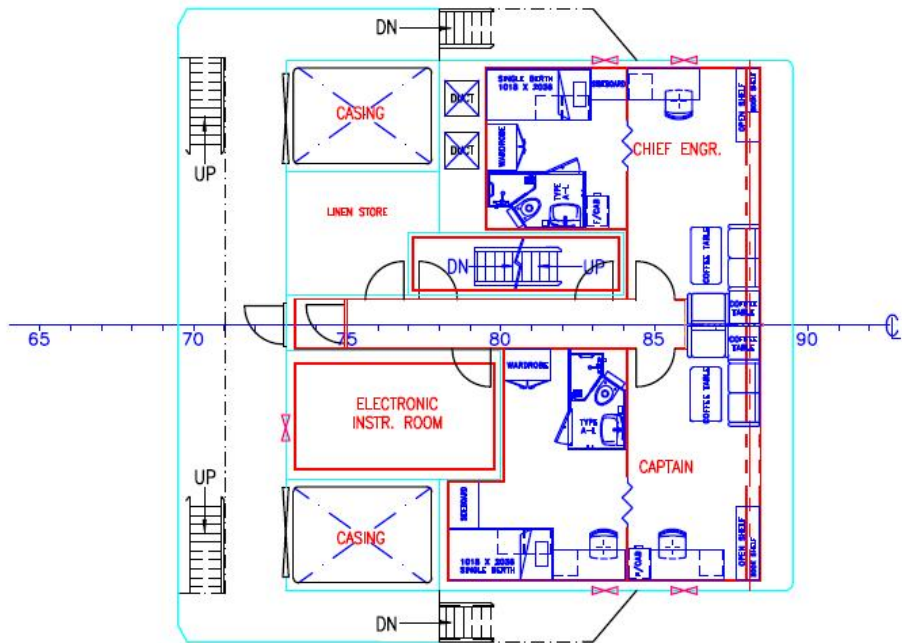


Figure A.1: Vessel side



BRIDGE DECK PLAN

Figure A.2: Bridge



OFFICER DECK

Figure A.3: Officer deck

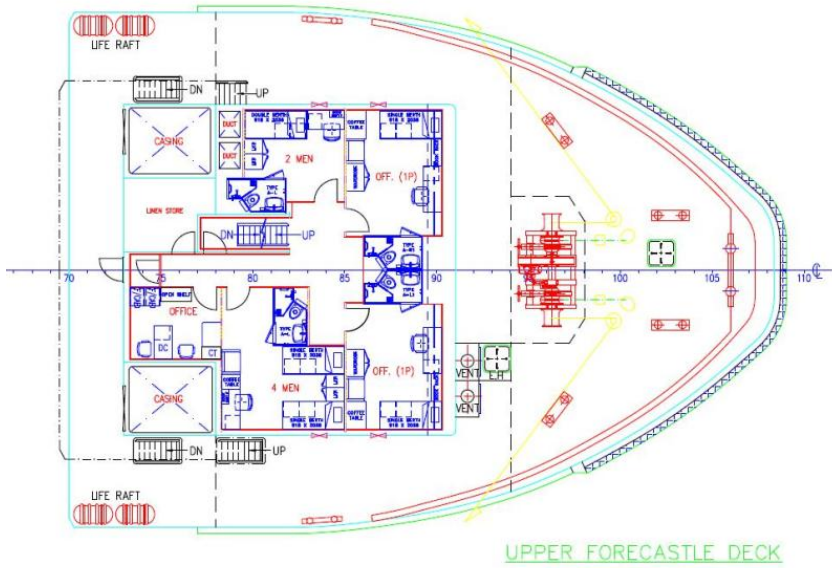


Figure A.4: Upper Forecastle deck

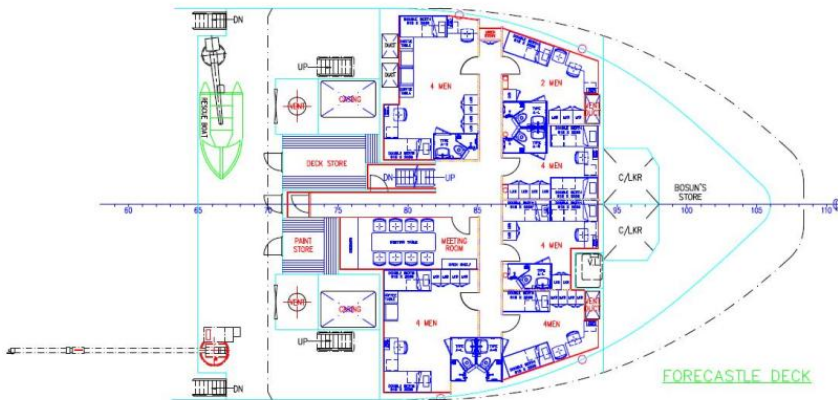


Figure A.5: Forecastle deck

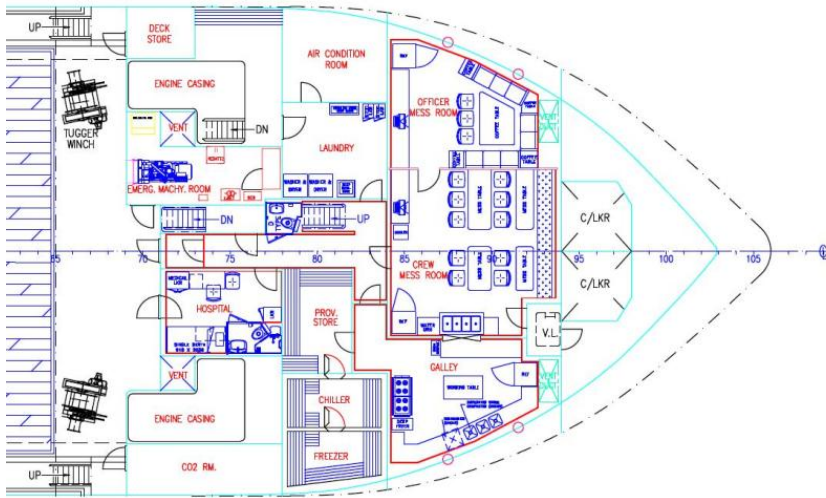


Figure A.6: Main deck