# AN OBJECT-ORIENTED APPROACH FOR VIRTUAL PROTOTYPING IN CONCEPTUAL SHIP DESIGN

Ícaro A. Fonseca; Henrique M. Gaspar
Faculty of Maritime Technology and Operations
Aalesund University College
Postbox 1517 N-6025 – Aalesund – Norway
E-mail: icaro.fonseca@ufpe.br; hega@hials.no

**KEYWORDS**
Object-oriented, Ship design, Virtual Prototyping.

## ABSTRACT

The ship design activity often requires handling and storage of large amounts of data related to different systems inside the vessel, demanding for a structured way to organize it. This article suggests an object-oriented approach to handle virtual prototyping data during conceptual ship design. We start presenting some of the basic concepts related to objects, such as name, property and value. A proposal based on the entity, state and process models is addressed for the virtual prototyping, related to the object-oriented approach. Later, we use the SFI group system as hierarchy structure to represent the ship as an entity and state model. We finish by presenting some simple examples of the proposed approach with a modular ship models and introducing one suggestion of a virtual prototyping model using the concepts presented thorough the paper.

## INTRODUCTION

The conceptual ship design phase consists on the generation, analysis and evaluation of diverse layouts, with a usual output of the main ship parameters (form) and its key performance indicators (KPIs and function). It appears in the literature under many titles – preliminary, feasibility, pre-contract – and even the term *concept* can have different meanings (Sen and Birmingham 1997). But during the early stages, the designer has a larger flexibility to search and select through the design space, with the intention to accumulate knowledge and improve the ship performance (Gaspar and Balland 2010).

In this context, we will discuss an object-oriented approach that should be able to represent the ship and its main systems, satisficing the conceptual phase detailing level, while giving a structured way to organize the design information and providing support to further detailing in the next stages of the lifecycle.

The arguments connected to this approach include basic object-oriented features, such as properties, attributes, values, instantiation, prototypes and encapsulation. We attempt to use those features in a way to support efficient data storage and interaction in order to represent designs, analysis and scenarios.

Our assumption is that the conceptual phase will generate this data in a virtual environment, where the physical shape is considered in its virtual form (e.g. 3D objects). The virtual prototyping (VP) aspect is then understood under Zorriassatine *et al.* (2003) definition, for which VP methods can be classified as:

- visualization,
- fit and interference of mechanical assemblies,
- testing and verification of functions and performance,
- evaluation of manufacturing and assembly operation,
- human factor analysis.

At this stage, we should attain to only some of those applications, such as visualization and performance testing/verification, in order to study or illustrate the potentials of our approach. Still, we may also give insights about how that approach could be useful for different applications.

Object-oriented capabilities may give a good support to virtual-prototyping applications: besides the aforementioned data and relation handling, the instantiation of objects allow the creation of several virtual prototype models with considerable less effort than the necessary to create them individually without that tool.

We also suggest the usage of a well defined structured hierarchy, such as the SFI group system, to represent the ship model digitally according to a function-based division.

## OBJECT ORIENTED SHIP DESIGN RESOURCE

Ship designers often struggle with the large amount and variety of information during conceptual phase. The design resources are considerably complex, since they should handle drawings, analysis and calculations for a wide variety of aspects including stability, hydrodynamic behavior, structural design, machineries and cargo systems, to name only a few. This complexity spans during the entire lifecycle, from conceptual design to the actual performance in operation and later

Proceedings 29th European Conference on Modelling and Simulation ©ECMS Valeri M. Mladenov, Petia Georgieva, Grisha Spasov, Galidiya Petrova (Editors)
ISBN: 978-0-9932440-0-1 / ISBN: 978-0-9932440-1-8 (CD)

decommissioning (Gaspar *et al.* 2012). Therefore, the design elements and their interactions can easily become cumbersome, requiring efficient ways to organize and structure them.

For those reasons, and given the collaborative nature of the design activity, it is expected that a resource aimed at ship design must offer efficient tools for storage, usage, search and version management, as a way to provide the main organization structure required to handle conceptual design information.

We propose a design resource supported by objects containing essential data about the ship, as well as basic performance information and links to all relevant design documentation. Although we present this approach to conceptual ship design, it may also be used during consequent phases as a tool to integrate and maintain the resources through the lifecycle, including virtual prototyping processes.

The *object* concept is rather simple; objects contain properties and values (Coyne *et al.* 1989). A property is an attribute assigned to an object in order to characterize it, and, in a coding language, it can assume values from a diversity of variable types, from numbers and strings to functions and even other objects. The functions are useful when establishing relations between the object's elements, since they can access other properties in order to manipulate data, perform calculations and obtain results. The possibility of inserting objects inside each other is useful to build hierarchical structures between them. Figure 1 presents a graphical representation of the Object/Property/Value relation.

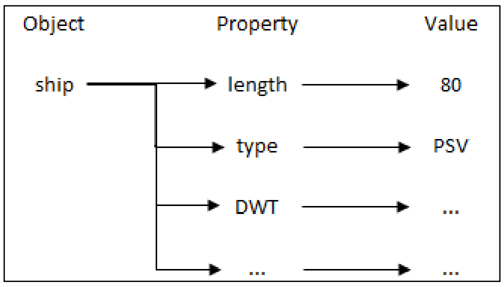
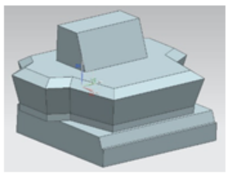


**Figure 1 Graphical Representation of a Simple Ship Object**

Here, the objects should represent the ship conceptual design, while the properties are usually their characteristics, specifications and documentation files. However, our ship object is not necessarily composed simply of properties with numerical or string values. It may also contain other objects, receiving their properties and values. See the example in Figure 2, where the Bridge is a object, part of a hierarchic superior object (Superstructure), and composed by hierarchic inferior objects (components). For each of these hierarchic levels we are able to define name, properties and values.



| Bridge (part of Superstructure object) | | |
|---|---|---|
| Property | Value | Type |
| Length | 13 | number |
| No. of decks | 2 | number |
| Cost | *add cost of each component* | function |
| Component1 | ... | object |
| ... | ... | ... |

**Figure 2 Ship Bridge Object Represented as a Frame**

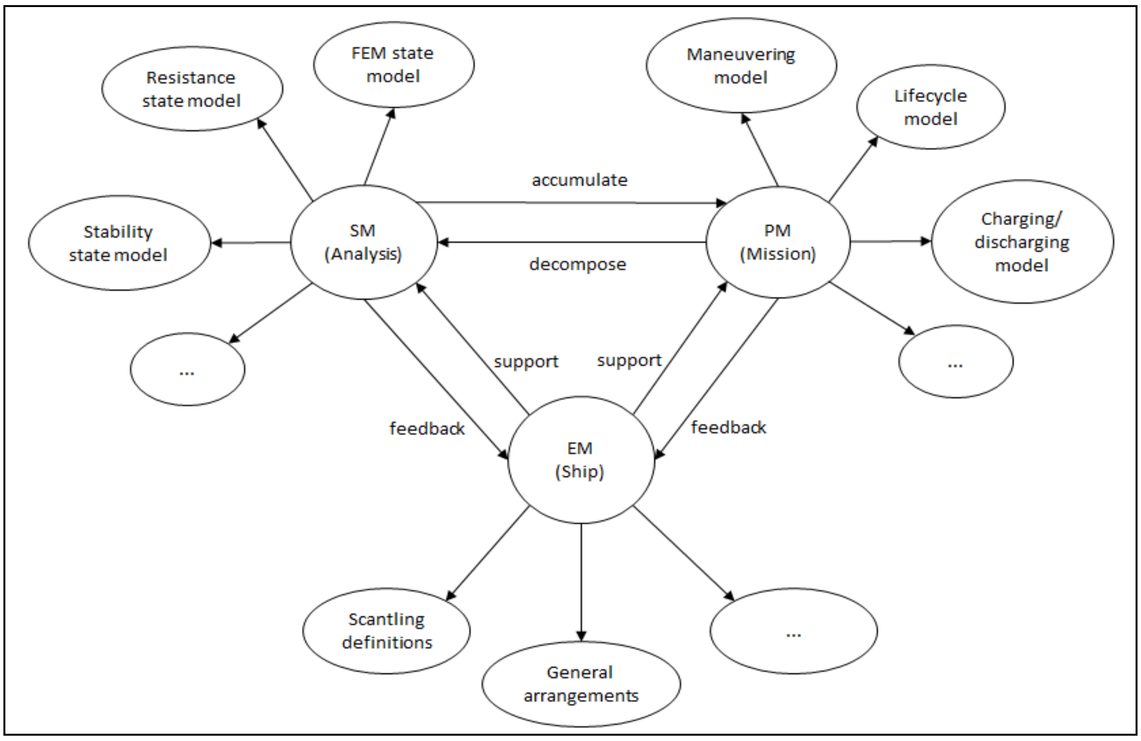


**Figure 3 Virtual Prototyping Representation Model Applied to Ship Design (adapted from He et al.)**

Beyond the physical model, a conceptual phase considers diverse analyses, as well as the performance of the system under many different scenarios. This virtual prototyping environment can be decomposed into three different models, interacting with each other in order to provide the virtual prototype scene (He *et al.* 2014). According to Figure 3:
i. Ship (Entity model - EM): the product model. Includes all components, 2D and 3D drawings or diagrams, mechanical, electrical models and everything that defines the product itself. It is the basis of the virtual prototyping, since the subsequent models are carried upon them.
ii. Analysis (State model - SM): the entity imposed to a certain set of conditions. Exemplarily, we can have analysis for hydrodynamic resistance of different speeds or loading conditions.
iii. Scenario/Mission (Process model - PM): the accumulation of analysis, ranging from the initial to the final state during the process. For our case, this could mean an operation, composed by a succession of seakeeping conditions for two entities (ship and platform).

These three models are related according to Figure 3, which also contains examples of what each model could be in a ship application. The Ship is the base of the whole model. The Mission can be understood either as an accumulation of Analysis or as the Ship subjected to dynamic constraints, ranging from the first to the last static constraint.

The object-oriented approach should be able to compile and structure those three models into a cohesive, unified resource.

## SFI AS A SUPPORT TO THE SHIP MODEL

Hierarchization is a key action to handle structural complexity of any complex system (Simon 1996). There are several *tag* systems available today able to represent a ship's systems, usually proprietary to a design/shipyard company. To document the ship design in a digital environment, with applications to our virtual prototyping model, we choose to use the SFI group system, which is particularly popular in Norwegian shipyards and design companies (Machinu and McConnell 1977).

The SFI group system is a coding and classification system for ship and oil rig components, which allows its users to handle extensive information by dividing the elements inside a vessel hierarchically, a structural breakdown of the ship. That information could include costs, working-hours, purchasing, maintenance or technical records, for example. It can be used for a wide range of purposes inside the industry, such as shipping/costs control and shipbuilding operations (Xantic 2001).

To support the hierarchical division, SFI introduces a code structure including *group, sub-group and detail codes*, each of them encompassing a certain degree of detail or system size. The Main Groups consist in one-digit numeric codes ranging from 0 to 9, where only Main Groups 1-8 come pre-defined. Main Groups 0 and 9 are open to include the user's own classification of systems not covered by the other main groups. Each main group includes up to 10 groups, described by two-digit numerical codes. The groups are further divided into subgroups (3 digit numbers) and detail or material codes (6 digit numbers), where these last describes one component/material of the vessel.

SFI adopts a function-oriented approach: systems and components are arranged in groups according to their functional purposes. Since the SFI group system was designed to conform to a ship's specifications, it performs well when using information related to one ship to estimate costs and other characteristics for similar designs.

A database version of the SFI group system is common available at ship design and shipyards, usually based on a long spreadsheet-like list containing the codes and KPI/cost properties. Those specifications can be related to different levels of the SFI coding structure: outline to Main Group level, functional requirements to Group level, functional solutions to Sub-Group level and component selection/As Built specifications to Detail Code. Drawings can be handled with an additional consecutive number, e.g.: 179-731-001, where the structure is: Ship no. – SFI Sub-Group no. – Consecutive no. Other applications of the SFI Group System are purchasing, maintenance and repair, filing, operation budget and quality assurance.

## OBJECT-ORIENTED VIRTUAL PROTOTYPING APPLIED TO SHIP DESIGN

With an object-oriented approach, different ships, states and missions may be represented as instantiations of a same prototype class (Figure 4). This allows the designer to reproduce the same pattern based on a general description.

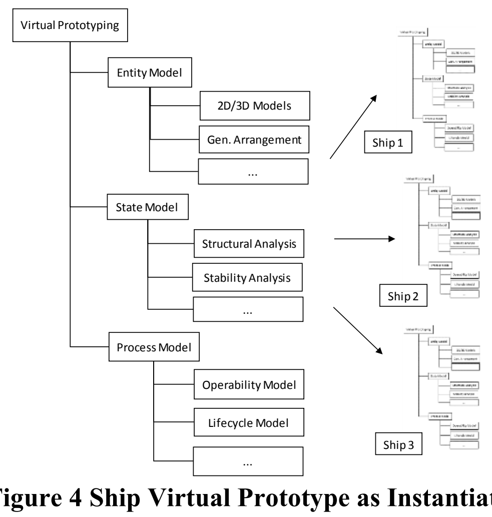
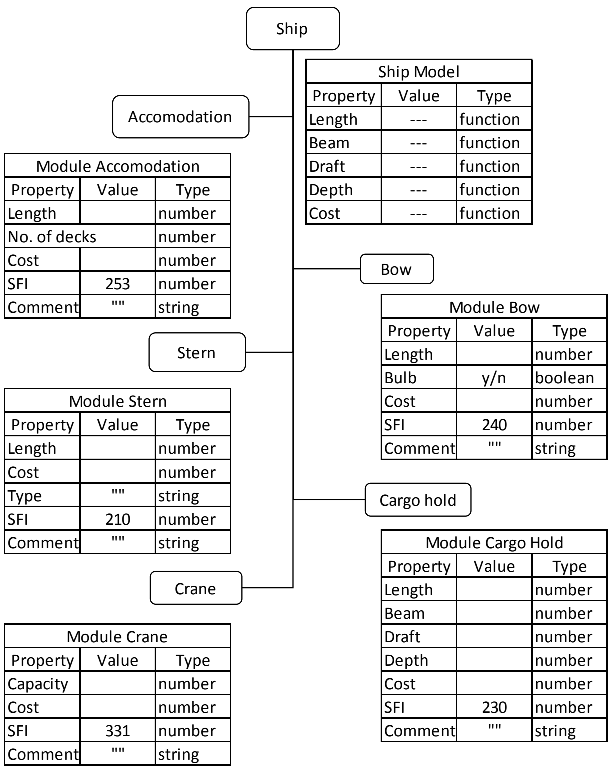


**Figure 4 Ship Virtual Prototype as Instantiated Object (adapted from He et al.)**

2D/3D models would be CAD and CAE models, drawings, arrangements and alike. Structural analysis may be composed of analytical and FEM simulations. The operability model listed as a process model could be an operability evaluation procedure defined by the designers according to their operational requirements. These are only some examples of data that can be relevant in a virtual prototyping context, and this structure can be expanded according to the designer's needs.

In order to illustrate some features of the object-oriented approach, we can present an example with a simplified object that stores some basic information about a ship. For that purpose, we split the ship in some modules with corresponding key characteristics, namely: Superstructure, Bow, Cargo Hold, Crane, Winch and Stern (Figure 5).



| Ship Model | | |
|---|---|---|
| Property | Value | Type |
| Length | --- | function |
| Beam | --- | function |
| Draft | --- | function |
| Depth | --- | function |
| Cost | --- | function |

| Module Accomodation | | |
|---|---|---|
| Property | Value | Type |
| Length | | number |
| No. of decks | | number |
| Cost | | number |
| SFI | 253 | number |
| Comment | "" | string |

| Module Stern | | |
|---|---|---|
| Property | Value | Type |
| Length | | number |
| Cost | | number |
| Type | "" | string |
| SFI | 210 | number |
| Comment | "" | string |

| Module Crane | | |
|---|---|---|
| Property | Value | Type |
| Capacity | | number |
| Cost | | number |
| SFI | 331 | number |
| Comment | "" | string |

| Module Bow | | |
|---|---|---|
| Property | Value | Type |
| Length | | number |
| Bulb | y/n | boolean |
| Cost | | number |
| SFI | 240 | number |
| Comment | "" | string |

| Module Cargo Hold | | |
|---|---|---|
| Property | Value | Type |
| Length | | number |
| Beam | | number |
| Draft | | number |
| Depth | | number |
| Cost | | number |
| SFI | 230 | number |
| Comment | "" | string |

**Figure 5 Ship Entity Model - Class Representation**

Now let us exemplify how the objects can be derived from those classes and have values assigned to their attributes with instantiation using a constructor pattern in pseudocode. Take the stern module, for instance:

```
function Stern(length, cost, type, comment) {
        Stern.length=length;
        Stern.cost=cost;
        Stern.type=type;
        Stern.comment=comment;
};
```

This creates a Stern class that can be instantiated with characteristics from different units, e.g.:

```
S1 = new Stern(20, 1000000, "Shaft", "Rudder needed");
S2 = new Stern(20, 1500000, "Azipod", "Better maneuverability");
Ship.ste = S1; //or S2
```

We can access these objects again by adding to the code a line that logs *Ship.ste* into the console, as well as access a given property inside the Stern object such as *Ship.ste.type*.

This ship model may also calculate some basic information based on the modules defined by the user. We implemented some simple arithmetic operations with the cost of the vessel for the sake of exemplification. Below is the code for the cost, where the total cost of the ship is the sum of the costs of each module.

```
variable Ship={
        cost: function() {
                variable totalCost = Ship.bow.cost + Ship.sup.cost + Ship.car.cost + Ship.ste.cost + Ship.cra.cost;
                return totalCost;
        }
};
```

Now let us suppose that one would like to expand this model to a greater level of detailing. As we detail the ship further and further during conceptual design and consequent stages, the amount of information would start to become overwhelming. We suggest the use of the SFI group system to address that issue. For example, an object mirroring the SFI different groups through the hierarchical structure, where the object representing Group 63 is a property of the object representing Main Group 6.

Still using the stern as an example, let us suppose now that we intend to decompose that part of the ship to a lower level of detail, and the propeller should now be described by an individual object inside the model. We can observe that structure in Figure 6.

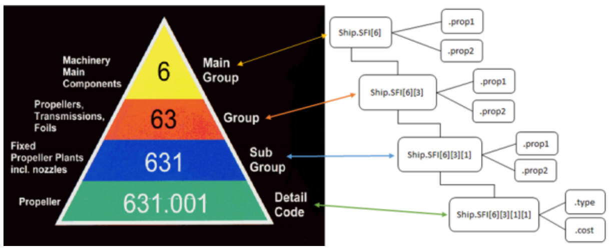


**Figure 6 Mirroring SFI with an Object-Oriented Data Structure (Xantic 2001)**

These group levels allow the designer to describe the ship in different levels of detailing, depending on their needs. If there are common properties among the ship's objects, say, many of the objects will have links to files with technical drawings, then it is also possible to use instantiation patterns to create those objects with the similar properties. For instance, *Ship.ste.documents* returns a list with all documentation from the Stern and its subgroups. *Ship.ste.documents.DWG* would filter down this list to only .DWG files while *Ship.ste.documents.STL* would return the 3D .STL files.

To expand the virtual prototype model into a collaborative design resource management, it is necessary to include functionalities such as version management, security and different levels of management authority. We do not focus on those functions at this stage of the work, but there are already several out of the box commercially available solutions to address those needs, such as Dropbox, Alfresco and Microsoft SharePoint.

We are also attempting to ally this approach to other ship design tools, which will let us feed the models with technical analysis. Currently for instance, Siemens NX is used for preliminary CAD tool and finite element analysis.

JavaScript language was chosen at this level of project, given the academic benefits that such language may offer. Current developments at the Ship Design Lab are strongly focused on online, open source and collaborative research. Practical development of such object in a proprietary language (such as C#) or server based (such as Python) would hide the advantage of any person in the word being able to run the virtual prototype in her machine/resources, with access to the source code in order to understand, change and improve it.

Although JavaScript may present performance limitations in the future, we are more concerned about accessibility and open collaboration than optimizing local performance and building a close software solution. Moreover, we can ally JavaScript to public visualization libraries (such as D3 – Bostock *et al.*, 2011), STL models, graphic user interfaces (GUI) using WebGL and to other features that come in hand with virtual prototyping applications, such as our own online prototypes (Chaves *et al.* 2015; Andrade *et al.* 2015).

## MODEL INTERACTIONS

Let us formulate how the different models interact among themselves to produce a virtual prototype model. For the Ship or Entity Model, besides the characteristics mentioned in the previous section (physical dimensions, SFI tags), we would have 2D/3D models and arrangements, with prototype visualizations.

To exemplify the prototype visualization features, we provide the example in Figure 7, where the user assembles the modules presented in the previous section with a simple GUI, which loads a STL file of the final model. The STL loading function may be added into the object so that the object itself loads the prototype visualization.

The interface can be expanded to include a brief description of the model according to the options chosen by the user, for example, a cost functions similar to the one presented on the previous section.

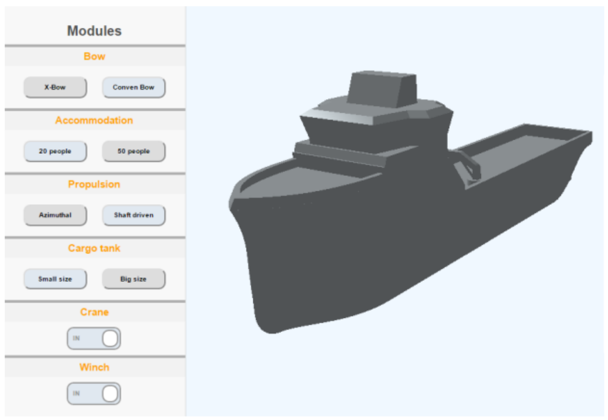


**Figure 7 Prototype Configurator - STL Visualization (3D Models)**

The next steps would be then to integrate the Entity Model with the other sub models. The interaction happens on two levels: first, the design resource works as a hub linking analysis files originating from third party software solutions (FEM, hydrodynamic resistance, stability and sea-keeping reports); second, it builds upon the ship, analysis and mission modules in order to offer the designer additional features. While we have found (and used) literature covering the first aspect, the second is still incipient from an object-oriented point of view, although opening many possibilities for application.

Besides the Entity, the other sub models inside the virtual prototyping are the State Model and Process Model. The first represents the analysis of the ship under a given state, while the later stands for a sequence of analysis corresponding to different states of the simulated operation. As previously explained, those models rely on the Entity Model, which collects all the files that define the ship (drawings, models, arrangements). The State Model accounts for the analysis, which includes, for example, finite-element analysis from third-party software, stability calculations, hydrodynamic responses and alike. The Process Model relates sum of the dynamic behaviors and simulations (from software such as NX, ADAMS) for a specific case/mission.

Those three sub models interact with each other to build the virtual prototyping. In a practical application, this means that the user defines those three aspects to obtain one Virtual Prototype model (Figure 8).

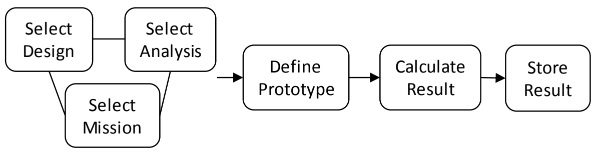


**Figure 8 Virtual Prototyping Process Integration**

In a utopic world, efficient computing would be able to calculate the whole virtual prototype tradespace, that is, analyze every state (e.g. FEM, CFD) for every design (entity) for every mission (process). Clearly the complexity of such approach grows increasingly as new elements are added, leading quickly to the unfeasibility of the VP. A set of rules/constraints is then necessary, preferentially applied during the construction of the object. If a process requires the CFD or FEM of a design, this value must be previously calculated and stored in the VP object. Quick parametric approaches may be useful (Parsons 1998; Gaspar *et al.* 2014; Ebrahimi *et al.* 2015) once that JavaScript allows similar benefits of other script languages to perform a quick function evaluation.

Other constraints will be imposed by the own nature of the virtual environment. A numerical offshore model basin focused on seakeeping of a new design will require a very precise state calculation, for instance seakeeping for every entity with a 0.01 second time step for a 2 hours analyses, while the 3D models may loose in detail for the sake of performance (Nishimoto *et al.* 2003; Gaspar *et al.* 2009). On the other hand, a VP aimed to convince customers that a design is innovative may focus on the details of the entity, while state and process are simplified. VP for training may require a trade-off among the three, with models realistic enough to emulate reality for the ship operator, while the ship response in real-time for a mission involving multiple operations, such as supply and anchor-handling (OSC 2015).

## LIFECYCLE COST APPLICATION

The initial test for the object-oriented is a simple *sum,* based on the necessity of designers and shipyards to acquire real-time costs for their multiple designs during the conceptual phase, requiring a holistic control for costs thorough the simulated vessel value chain (Figure 9). In this context, a simple lifecycle management scenario is presented. Such sum is nowadays done with the aid of diverse *spreadsheet-like* database, and such object could be fed by these databases, to be used in a VP environment.

The entity model representing the ship provides the necessary information to calculate and infer the costs for acquaintance and construction of every structural element of the ship, from the main groups (SFI 0-9), to an specific component (e.g. *SFI XXX.1234 oil pump*).

The State Model would encompass individual cost analysis for different engineering evaluation during lifecycle stages, for instance hours/cost for preparing each of the FEM and CFD analysis required during conceptual phase, as well as hours to prepare documentation to the yard during the basic phase. A single ship (entity) can have multiple states, if such ship is simulated to be constructed in Norway or in China, given the different engineering work required by each of the yards.

The Process Model should be a succession of State Models, strongly connected to the economical evaluation of the vessel operation. Modern value-robustness techniques, such as Epoch-Era Analysis (Ross and Rhodes 2008; Gaspar *et al.* 2015) can be used to check if the simulated vessel is the *right vessel for the right mission*, based on the return on investment of the design.

In this case, it is a lifecycle cost model for a given design, which adds the different costs estimated on the State Model. The cost of the simulated lifecycle is thus the cost of the physical object (entity) plus the required analysis for analyzing and evaluating that object (state), plus the return on investment of the design for a given period or era (process).

It is possible as well to extend the model to take into account discount factors to calculate the net present value of the design proposal. This could be accomplished by breaking the costs into several parcels with different discount coefficients applied to them. With the results provided by this model, the designer is able to evaluate the proposals from a long-term financial point of view.

A configurator similar to Figure 8 is then created as GUI to handle our object. A list of the available designs is presented, either by the modular approach (Chaves *et al.* 2015), or by pre-defined designs. The object is able to link the necessary states when the design is evaluated in a process, evaluating then the lifecycle cost for many scenarios, such as construction in China and operation in Brazil; or construction in Norway and operation in Africa (Figure 9).

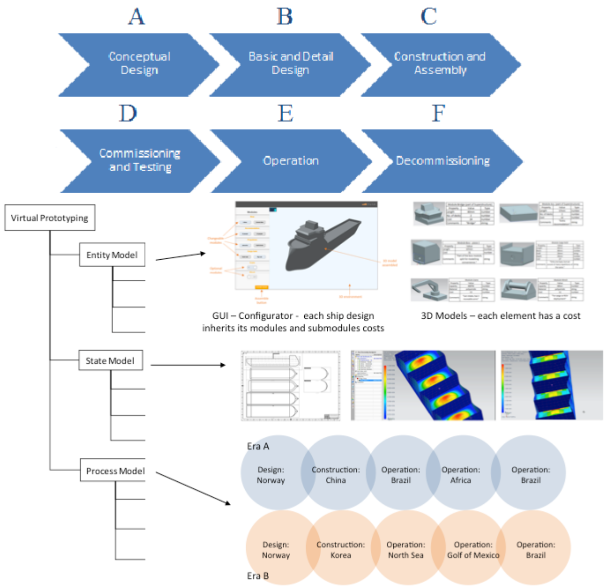


**Figure 9 Ship Design Value Chain and Virtual Prototype Object Illustration**

## DISCUSSION AND NEXT STEPS

The next steps of this project, which should span during the next two and a half years, is to research on how to build virtual models that can handle the ship design complexity using the principles here presented. The interactions between the models may indeed prove to be a challenging part of this task, since they tend to increase in complexity as we look for improved ways to represent the designs.

An online virtual prototype environment is expected to be created, incorporating the main models presented in this article. A 3D library of ship designs will be connected to the entities. Parametric and pre-calculated analyses will be stored in states. Pre-defined missions will be part of the processes models.

As we develop this project and identify the functionalities required from the design resource by the designer, as well as the ways the virtual prototyping models interact with each other, we expect to develop elements of reusable object-oriented software (creational patterns) into the code.

Summing up, this article introduced the main features of an object-oriented approach applied to virtual prototyping, laying down some of its fundaments to further development in ship design. In our perception, those capabilities make this approach a potentially powerful tool to assist virtual prototyping during conceptual ship design stage. We would be glad if more researchers also felt enthusiastic about this idea and shared our interest in further developing the standards and models here discussed.

## REFERENCES

Andrade, L. S.; T. G. Monteiro and H. M. Gaspar. 2015. "Product Life-Cycle Management in Ship Design: from Concept to Decommission in a Virtual Environment". In *Proceedings 29th ECMS.* Varna.

Bostock, M.; V. Ogievetsky; J. Heer. 2011. “D3: Data-Driven Documents”, *IEEE Trans. Visualization and Computer Graphics*.

Chaves, O.; M. Nickelsen, and H. M. Gaspar. 2015. "Enhancing Virtual Prototype in Ship Design Using Modular Techniques". In *Proceedings 29th ECMS.* Varna.

Coyne, R.D.; M.A. Rosenman; A.D. Radford; M. Balachandran and J.S. Gero. 1989. “Chapter 3”. In *Knowledge-Based Design Systems*, 87-151.

Ebrahimi, A.; P. O. Brett; J. J. Garcia; H. M. Gaspar and Ø. Kamsvåg. 2015. “Better decision making to improve robustness of OCV designs”. In *Proceedings 12th IMDC (2015).* Tokyo.

Gaspar, H. M.; C. Fucatu and K. Nishimoto. 2009. “Design of Conceptual Offshore Systems based on Numerical Model-Basin Simulations”. In *Proceedings 10th IMDC (2009).* Trondheim.

Gaspar, H. M.; D. H. Rhodes; A. M. Ross and S. O. Erikstad. 2012. "Handling Complexity Aspects in Conceptual Ship Design: A Systems Engineering Approach". *Journal of Ship Production and Design*, November, Volume 28, pp. 145-159.

Gaspar, H. M. and O. Balland. 2010. “Approaching environmental performance in conceptual ship design.” In *Proceedings 11th PRADS.* Rio de Janeiro.

Gaspar, H. M.; P. O. Brett; A. Ebrahim and A. Keane. 2014. “Data-Driven Documents (D3) Applied to Conceptual Ship Design Knowledge”. In *Proceedings COMPIT 2014.* Redworth.

Gaspar, H. M.; P. O. Brett; S. O. Erikstad and A. M. Ross. 2015. “Quantifying value robustness of OSV designs taking into consideration medium to long term stakeholders’ expectations”. In Proceedings 12th IMDC (2015). Tokyo.

He, B.; Y. Wang; W. Song and W. Tang. 2014. “Design resource management for virtual prototyping in product collaborative design.” *Proc. IMechE, Part B: Journal of Engineering Manufacture*, 1-17.

Manchinu, A. and F. McConnell. 1977. “The SFI Coding and Classification System for Ship Information”. *Proceedings of the REAPS Technical Symposium Paper No. 19*.

Nishimoto, K.; M. Donato; I. Q. Maseti; B. P. Jacob; M. Martins; I. Menezes and K. Hirata. 2003. “Development of Numerical Offshore Tank for ultra deep water oil production systems”. In *Proceedings 22nd OMAE*. Cancun.

OSC, 2015. Training [Online].
Available at: http://www.offsim.no/Training

Parsons, M. G. 1998. “Parametric Design”. In *Ship Design and Construction vol. 1. SNAME*.

Ross, A. M. and D. H. Rhodes. 2008. “Architecting systems for value robustness: Research motivations and progress”. In *SysCon 2008 - IEEE International Systems Conference*.

Sen, P. and R. Birmingham. 1997. “State of Art Reports”. In *Proceedings 6th IMDC (1997).* Newcastle upon Tyne.

Simon, H. 1996. The sciences of the artificial. MIT Press.

Xantic. 2001. *SFI Group System – A system for classification of technical and economic ship information. Product Description.*

Zorriassatine, F.; C. Wykes; R. Parkin and N. Gindy. 2003. “A survey of virtual prototyping techniques for mechanical product development.” *Proc. IMechE, Vol. 217 Part B: Journal of Engineering Manufacture*, 513-530.

## ACKNOWLEDGEMENTS

This research is connected to the Ship Design and Operations Lab at HIALS, which is partly supported by the EMIS Project, in cooperation with Ulstein International AS (Norway) and the Research Council of Norway. Ícaro A. Fonseca is an exchange student from Federal University of Pernambuco (UFPE), and he is partially supported by the Science without Borders CAPES scholarship, proc. no. 88888. 055633/2013-00.

## AUTHOR BIOGRAPHIES

**ÍCARO A. FONSECA** is an undergraduate student in Marine Engineering at Federal University of Pernambuco. Currently he is attending to a one-year exchange program in Ship Design at Aalesund University College. He looks forward to receiving opinions about this paper at his email `icaro.fonseca@ufpe.br`.

**HENRIQUE M. GASPAR** Associate Professor at the Aalesund University College. PhD degree in Marine Engineering at the Norwegian University of Science and Technology, Marine Systems group. Part of the PhD developed at the Systems Engineering Advancement Research Initiative (SEAri) at MIT, with complex system engineering methods applied to the maritime case. Previous experience as Senior Consultant at Det Norske Veritas (Norway) and in Oil & Gas in Brazil. `http://www.shiplab.hials.org/`.