An Enterprise Modeling Approach for the Early Ship Design

(Ein Ansatz zur Geschäftsmodellierung in der frühen Schiffsentwurfsphase)

Doctoral Thesis

for

the academic degree of

Doktor-Ingenieur (Dr.-Ing.)

submitted to the Board of the Faculty of

Mechanical Engineering and Marine Technology

of the University of Rostock

Rostock, 2015

submitted by
Wisam Jabary, born on 21.06.1983 in Latakia
from Latakia/Syria

Reviewers:

1. **Reviewer:** Prof. Dr. Ing. Robert Bronsart
   Lehrstuhl Schiffbau, Universität Rostock
2. **Reviewer:** Prof. Dr. Eng. Patrick Kaeding
   Lehrstuhl für schiffstechnische Konstruktionen, Universität Rostock

Date of Submission: 25.02.2015
Date of Defence: 02.07.2015

Contents

List of Figures

List of Tables

Conventions

Within this dissertation the following style conventions are used:

- *Italics* is used for:
  - names of the developed *information objects* and their attributes.
  - names of the *predefined activities* and *activity partitions*
  - names of the implemented *UseCases*
  - within the *Introduction* and *Summery and Outlook* to refer to terms utilized within this thesis.
- `Courier` is used for:
  - names of defined `ConfigurationItems`
  - describing the `Macro Language for Compartmentation-Definition`

1. Introduction

Maritime transport is being substantially influenced by the volatile global economic development. This results in very much fluctuating transport demands, directly linked to the shipbuilding industry as the ship operators' investment strategies towards newbuildings are mainly influenced by this situation. On the other hand, the demand for ships for spending peoples holidays on like cruise vessels and yachts, a very important market segment of the Europaen shipbuilding industry, is damped down and dynamically developing in a higher frequency. Additional aspects with greater impact on the maritime industry are the dramatic changes in fuel oil prices as well as an increased social awareness about environmental issues. These developments lead to a steadily increasing competition in the global and very transparent market. Shipyards facing this situation are forced to continuously develop their strategies to stay competitive and by this successful. In the last decades, many shipyards specially in Europe disappeared from the market as they have failed to develop new products and to implement new production strategies.
This situation forces shipyards to permanently develop their design methodologies in order to be able to efficiently create new designs being competitive under these challenging conditions. The early design stage plays a dominant role for two reasons: the first is related to the great influence of these studies on the entire design, and the latter is linked to the fact that the results of these studies are the basis from which the offer in the tendering process is derived. Considering these facts, the main objective of this research is subjected to the enhancement of this early phase.
In order to achieve the objective of supporting the early ship design phase, the *Enterprise Modeling Approach* is adopted as a basis for the developments accomplished within this dissertation. The main idea of applying the *Enterprise Modeling Approach*, which is introduced principally regarding its compatibility with the *System Design Engineering* perspectives in chapter 3, is to take advantage of the progress and evolution of the computational engineering fields in order to increase the efficiency of studies made within the ship early design stage. The main motivation for adopting this approach is its principle concept of the integration between data and process. This concept is applied by taking into consideration the specifications of ship early design stage introduced in chapter 2. These specifications are derived from the early design stage analysis regarding its importance, complexity and its research-needs. Many management concepts, such as *Configuration*, *Change*, *Version*, etc., will be introduced principally in chapter 3 (as they are needed to be applied within the adopted enterprise approach) with several objectives in mind: to increase the efficiency, to support the dynamic nature of the early design process and to support achieving the objective target of this thesis as it is introduced in chapter 2.
In order to develop an *Early Ship Design Enterprise Model* and to make use of all its advantages regarding the key aspects of the early design process, such as *Decision Making*, *Data Reuse*, *Data Exchange*, etc., (see chapter 6), two types of models considering the early design phase are developed. The former, which is the *Ship Information Model*, is introduced in chapter 4. It is intended to increase the level of details of data at the early design phase as well as to provide these data in an efficient form. It is based on international standards and comprises three types of modelled information: *Ship Product*, *Support* and *Management* (which reflect all needed management fields). These modelled information are not isolated but rather interrelated and they support each other. The built information model on the one hand offers the foundation of consistent data-data interaction and on the other hand offers the foundation of an efficient

data-process integration. The latter is introduced in chapter 5 and represented by the *Ship Early Design Process Model.* The *Predefined Activity*-concept, which is introduced from the data point of view in chapter 4 and from the process point of view in chapter 5, is the core concept, which is adopted to be used to model the early design process. This process is formed in many tasks, each is modelled regarding the *Predefined Activity*-concept following its analyzing with respect to its input-, control-, and output-data. Thereby, the interactions between these tasks are derived in order to be effectively respected in the implementations introduced in chapter 7.
In chapter 7 a *Ship Early Design System* is presented. It represents an efficient integration of two main parts. The former is the *central database* (data management component), within which the ship information model is set up and managed throughout the design process. The latter are the *UseCase*s, by which the predefined activities are implemented. In this context, the set-up database represents the basic middle-ware of all ship early design tasks, which are modelled as predefined activities. Thereby, the ability of performing these tasks by means of any external tool and by design members may located in geographically dispersed placements is insured. The performing of all early design tasks is based on the data saved within the *central database.* Therein, data-consistency as well as the efficient task-interactions are guaranteed. This is supported by means of *UseCases*, by which the process perspective of the adopted management concepts are implemented. For example, the applied *Configuration Management* concept provides the ability to combine all data, which are related to a ship early design task represented by a *Predefined Activity*-concept, in a high level construct called *ConfigurationItem.* Thereby, the assignment of a *Status* to these combined information is insured.
The developed approach offers a neutral means, which is independent of any shipyard setup. It can be applied regardless of the type of the designed ship. Therefore and in order to evaluate its efficiency, the developed ship early design system is integrated in a ship design workbench. Proof of concept is presented in chapter 8 based on the early design study of a Ro-Ro ship.

2. State of the Art in Ship Design

Ship design is the process of creating the needed models, necessary data and properties for the required subsequent assessments and for the production of a ship optimally fulfilling the customer's requirements and all rules and regulations to be observed.
Like other design engineering fields, the ship design process has seen significant developments in recent decades as a logical consequence of the continued progress in computational sciences and the increasing competition in the global market. This prompted shipyards and design offices to efficient permanently develop new methodologies in order to make this process more effective.
Nevertheless, there are some design related issues which can be considered general and fairly constant aspects [101]. One of these aspects is the correct understanding of the owner's requirements. These requirements set the limits of the design process and must be respected by the design team in order to be able, in the available constrained time, to design a ship which is: on one hand accepted by the owner and on the other hand satisfies all reliability, safety and commercial issues. Another important aspect is the fact that the ship design process is a team process, its studies are achieved through the partnership between many disciplines like hydrodynamics, machinery, outfitting, etc. and many partners like owner, shipyard, designer, supplier, classification societies, model basin, etc. These partners can be located in geographically dispersed areas which increases the complexity as well as the difficulty of the design process.

2.1. Ship Design Process

The ship design and production process can be divided into different phases. In spite of the complementary nature of these phases, they are differentiated from each other with respect to: the tasks performed within each phase, the capabilities required from the persons who perform these tasks, the data detail level of the ship at the end of each phase [101]. However, the

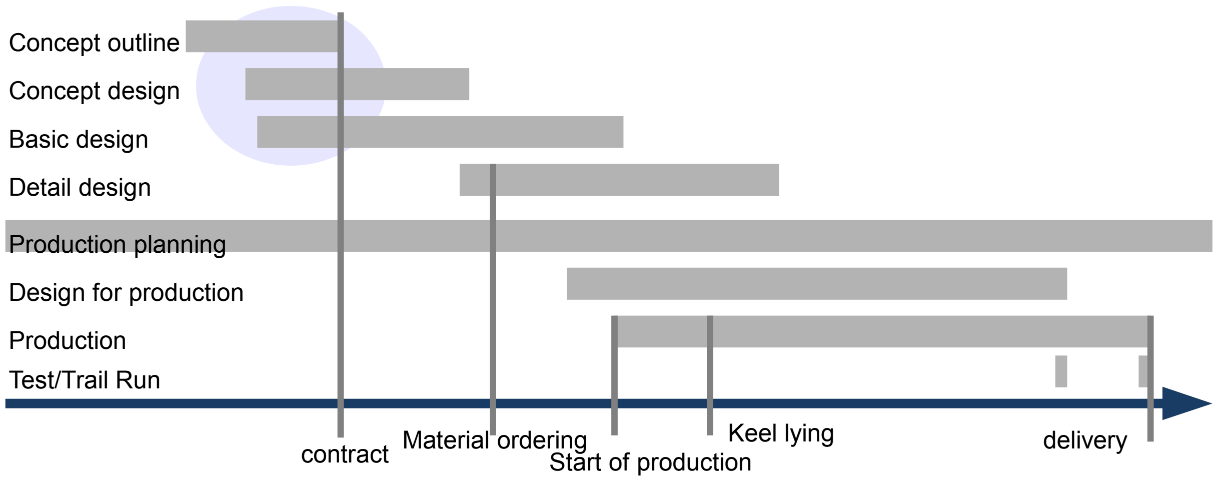


Figure 2.1.: Ship Design and Production Process according to Bronsart [43]

subdivision into phases facilitates the complexity of the design process by partitioning the design progressive efforts. The number and names of the ship design and production phases vary from reference to another. For example, Gale [101] divided the ship design and production process into 'Basic Design' and 'Product Engineering'. The 'Basic Design' has been further divided into: concept-, preliminary-, contract- and functional design. Whereas, the 'Product Engineering' has been divided into: transition design and workstation/zone information preparation. Another example is the phases subdivision introduced by Bronsart [43]. The Figure 2.1 shows the time scale of the ship design and production phases in addition to its major 'milestones' represented by contract, material ordering, start of production, keel lying and delivery. As it is depicted in Figure 2.1, four phases (concept outline, concept design, basic design, detail design) represent the design process and four phases (production planning, design for production, production, test/trail Run) represent the production process. What follows is a brief description of the design phases:

- Concept outline and concept design: The main objective of these phases is to perform the all design tasks, which are needed to specify the offer-parameters (cost, ship general characteristics, speed, etc.) with respect to the shipowner's and other design requirements.
- Basic design: The studies related to this phase are performed traditionally after the contract is signed to build a new ship. Therefore, the main objective of this design phase is to start providing the all required data and properties (such as hull form, intact and damage stability, predicted power, compartmentation, etc.) to build the new ship(s). In fact, the design tasks, which are performed within this phase, are almost similar to those performed before the contract between shipowner and shipyard with the difference in the nature and details of input and output data of these tasks. Moreover, the design tasks, which address the ship structure design (the material and dimension of the ship structure elements like, plates, bulkheads, etc.) are performed within this phase.
- Detail design: The purpose of this phase is the working on the results (properties) gained from the previous phase in order to increase their details to a level suitable for the next procedures. The results of the tasks performed within this phase are the basis for the production process.

However, in this research, the term 'early design phase' is used to refer to the part of the ship design process, which is the subject of the approach developed within this dissertation. As it is highlighted in Figure 2.1, the developed approach can be used to serve the design studies in the time frame from the beginning of the design process until after the contract.

2.1.1. Design Process Modeling

The ship design process has been represented traditionally by using the design spiral (shown in the Figure 2.2). The design spiral was introduced by Evan [72] to describe the structural design and then extended to describe the complete ship design process. There are many versions of the design spiral, some of them represent the general design process and others have been used to describe a specific design stage. In latter case, the characteristics of the addressed design phase is taken into consideration. Some of these versions are shown by D. Andrews et al. [35]. In general, the representation of ship design process by means of the design spiral reflects the following aspects:

- The iterative nature of the design process, which in fact reflects the complexity of this process. The process cannot be represented by some equations, which are solved directly, but rather by iterated cycles until the achievement of a satisfactory design solution. This is an advantage of the representation when the nature of the early phase is taken into consideration.

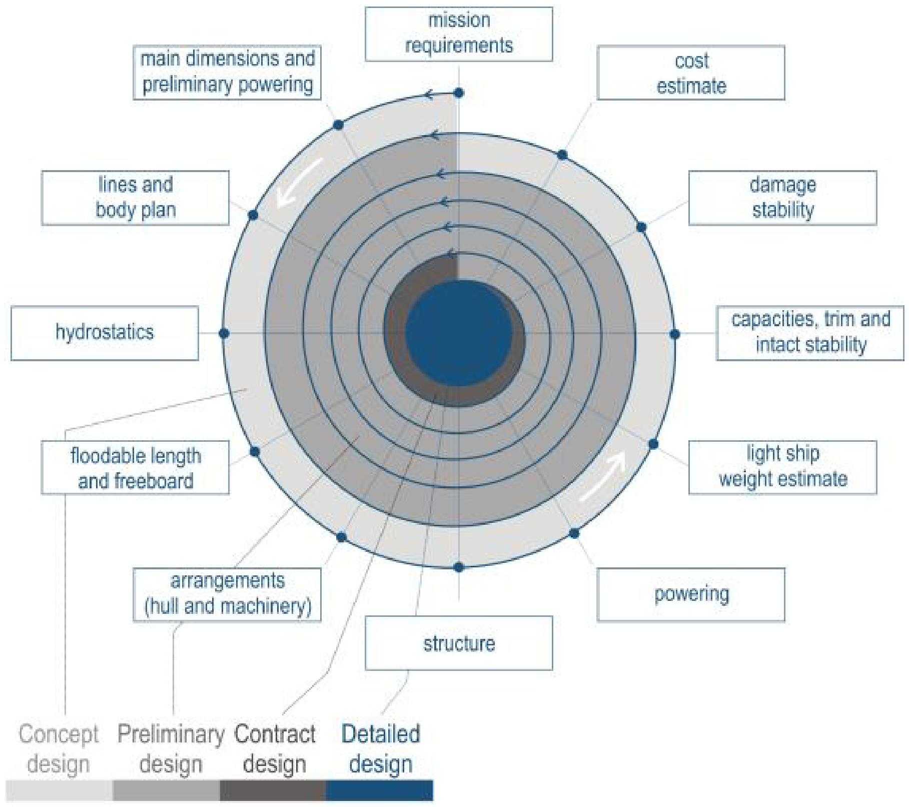

mission requirements
cost estimate
damage stability
capacities, trim and intact stability
light ship weight estimate
powering
structure
arrangements (hull and machinery)
floodable length and freeboard
hydrostatics
lines and body plan
main dimensions and preliminary powering
Concept design
Preliminary design
Contract design
Detailed design

Figure 2.2.: Traditional Design Spiral, [120]

- The sequential nature of the design process through a number of steps followed successively. Each step represents a specific discipline such as (hydrostatics, powering, etc.)
- The complexity nature of the design process through the diversity of perspectives of the design participants, for example hydrodynamic-, structure-perspective,...
- The progressive elaboration nature of the design process [63]. Mistree et al. [110] use the "convergent/divergent" aspect to describe this nature. They point out that as the process is progressed and converged with respect to the general design spiral, the amount of information about the product (ship) is increased reflecting the divergent aspect. This led Rawson et al. [126] to reverse the spiraling direction of the represented ship design process.

Despite of the advantages, which can be counted to the traditional design spiral representation, it can be stated that this representation is not capable to describe the real nature of the ship design process today. The weaknesses (disadvantages) of the traditional design spiral representation have been addressed by many authors, and are briefly described in the following:

- In practice the design process is not sequential. Gale [101] pointed out that, during the design process, the designers may move or jump to perform design tasks, which are not located directly next to each other regarding the order of the design tasks represented in the design spiral. The same aspect has been mentioned by Pradillon et al. [92], he stated that the ship design process (and accordingly its associated design tasks) is a dynamic process. As a result, it is possible to start some independent tasks without considering the sequence giving by the design spiral. Furthermore, Nowacki [116] argued that this representation of the design process in sequential closed loops contrasts with the decision making nature of the design process especially at the early stage. Thus, Andrews [38] attempts to overcome this closed nature and to take into consideration the interactions between designer and external resources or constraints by adding a new dimension to the design spiral. He represents the design process of the major phases in a three dimensional design spiral as a "tapering corkscrew".

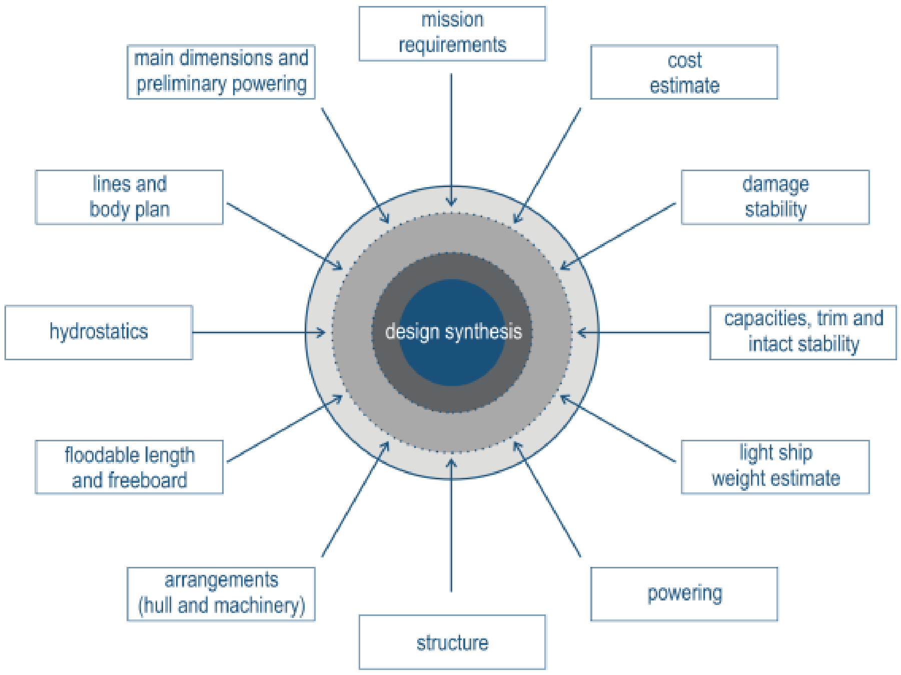

mission requirements
main dimensions and preliminary powering
cost estimate
lines and body plan
damage stability
hydrostatics
design synthesis
capacities, trim and intact stability
floodable length and freeboard
light ship weight estimate
arrangements (hull and machinery)
powering
structure

Figure 2.3.: Integrated Design Spiral, [120]

- The parallel or simultaneously nature of the design process particularly in the early stage (see Figure 2.1) is not considered in the traditional spiral. The model shown in Figure 2.3, which has been introduced by Papanikolaou et al. [120], represents the integrated nature of the design process. It reflects the ability to start any design task as soon as the needed information is available.
- The depiction of the design process activities (tasks) in consecutive steps implies many discrepancies with the fact of the interaction and overlapping of the activities during the design process. Furthermore, it disguises the activities details, for example, it hides the fact that these activities are interdependent and give input or take output from each other.
- Mistree et al. [110] criticized the traditional spiraling representation of the design process in terms of its ineffectiveness in providing innovative design solutions. They state that this approach is uneconomical, especially when the needs of superior design solutions are taken into consideration.
- Levander [105] pointed out that the describing of the design process by means of the design spiral undermines the innovation and creativity during the design process. The designer is locked to his first assumptions, which has a negative effect regarding the developing and evaluating of new design alternatives. Therefore, Levander introduced a "System Based Design" approach with the main objective of reducing the number of loops needed to find the suitable (technically and economically) design solution. It is based on the transforming of the different design requirements (issued by the owner and legal organizations (see section 2.2.3.4)) into an identifying of specific functional requirements.

In summary, it can be concluded that the depiction of the design process in a spiraling approach reflects many true facts about the nature of the ship design process. But at the same time, it masks a number of important aspects, which have to be represented and considered very early in the ship design process.

2.2. Ship Early Design Phase

In the literature many authors who have addressed the early ship design phase differ in the naming of the phase, such as concept, early, feasibility, initial, etc. However, they unanimously agree that it is demanded to continue the research related to this stage. This is due to the intrinsic importance and the complexity associated with the studies related to this stage. In the following, each of these aspects in addition to the characteristics associated to the early design phase will be illustrated in some detail. This helps to portray the motivations for the studies and developments of this dissertation.

2.2.1. Relevance-Impact on Overall Ship Design

As it is shown in Figure 2.2, the identification and elucidation of the mission requirements like service speed, cargo capacity, etc. formulate the starting point of the early design process. The importance of this stage originates mainly from the following:

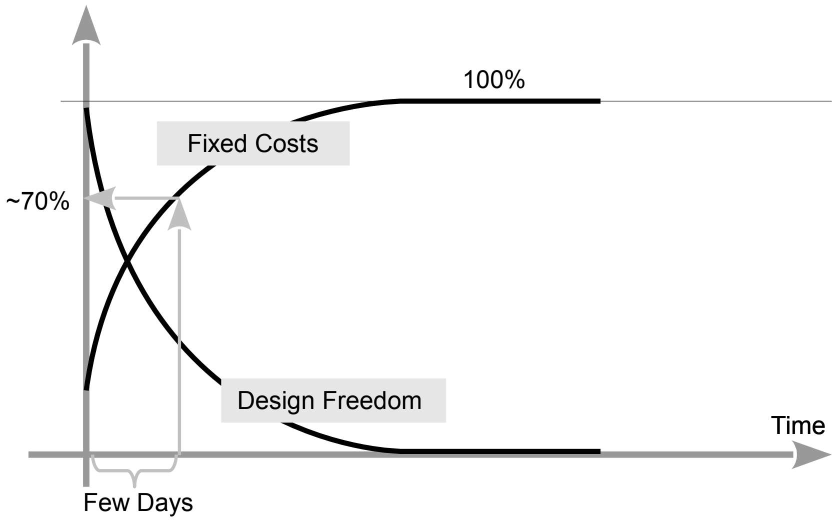


Figure 2.4.: Design Time Line, [43]

- Bronsart [43], Pawling et al. [69], Erikstad [56] and other authors pointed out that about 70% of the costs associated with the life cycle of the ship are influenced by the decisions made at this stage as it is shown in Figure 2.4, which is reproduced from [43]
- The preparation of an offer by the shipyard for a new contract is made through and based on activities and results achieved within this stage. As a result, as much as the development of this stage in a shipyard, in terms of accuracy and achievement, as much as its ability to compete in the global maritime market. This has a great importance due to the intense competition in the global maritime market and forces shipyards to permanently improve themselves to face these considerable challenges.
- After getting a contract to build a new ship/small series, this stage with all its details is considered the basis upon which will be built in the following design phases. Therefore, the providing of the resulted data from this stage in a form suitable to be used in the following

stages will increase the effectiveness of the overall design process. A brief example is the ship general arrangement. It is created traditionally based on two dimensional drawings in the early design stage and three dimensional models in the following stages. The providing of the general arrangement model in 3D from the very beginning in the design process will led to overcome the traditional negative effects (associated with its change environment (2D and 3D)) on the entire design process.

2.2.2. Complexity

The early ship design process is associated with some difficulties, challenges and problems which increase its complexity comparing to other design stages [60]. This led Pawling et al. [69] to describe the studies related to the early design stage as a "wicked problem". Some of these difficulties are discussed in the following:

- The low amount of available data at this early stage. This leads shipyards to reuse the data of the previously built ships after adapting them to be suitable to the new ship requirements (this issue is discussed later in more detail).
- More and more information and aspects have to be taken into consideration by the early design phase studies. Gaspar et al. [68] have concluded that this fact imposes new constraints and requirements on the design process. They have reviewed the growth of the data associated with the early stage at the recent decades and introduced an approach to handle the increased complexity. The introduced approach based mainly on the "decomposition-encapsulation"-concept by dealing with the design problem through five aspects which are: behavioral, structural, contextual, temporal and perceptual.
- Time constraints of this stage: shipyards have just about few weeks [69] to perform all studies of this stage before they give an offer, which must be economically safe, achievable and compatible with the design requirements. As a result, the designers, who are responsible for performing this stage studies, are working permanently under time pressure to make the compatibility between the affordability and capabilities in order to create suitable and achievable design solutions (see section 2.2.3.1).
- The design elements are not independent, but rather, they are interrelated and interacted. An example is the important interaction between the hull form and the compartments, the compartments and machinery, etc. As a result, any change in one of these interdependent design elements will affect the others. Furthermore, these elements can relate to different disciplines, such as machinery, outfitting, structure, etc., which makes this interaction even more difficult to be treated due to the different points of view of these disciplines regarding the ship design process.
- The procedures associated with the design process and correspondingly with the early phase as shown in Figure 2.2 can be classified as design tasks like definition of arrangements (hull form, compartmentation, etc.) and analysis tasks like (damage stability, lightweight estimation, powering, etc.). These procedures, as discussed in section 2.1.1, are performed sequentially, simultaneously and iteratively. However, during this and as the design process proceeds, some issues can be raised and should be treated carefully. In the following are some of these issues:
  - A change on the characteristics like dimensions of the hull-form or compartments is requested and forced as a result of design or analysis task(s). An example to illustrate this aspect are the modifications required to change the dimensions of the engine room compartment, which is resulted from the performing of the compartmentation task, in accordance to the dimensions of the specified main engine (specified depending on the results of the power prediction task in terms of the defined hull form at the required

speed range).

– The required changes for one task conflicts with the results or requirements of one or more other tasks due to the interaction between the design elements as mentioned previously. As a clarification, by following up the previous example, if the modification of the engine room dimensions affects the results of the damage stability analysis of the ship or may affect the dimensions of other compartments such as tanks.
  These issues can be expressed briefly as follows: "changes" in the early design phase are constantly required and in many cases lead to "conflicts". Therefore, it is very important to efficiently address these changes in order to manage their effects. On the other hand, the large change allowance at early design phase in comparison to the remaining design phases is the other fact which must be taken into consideration. The design freedom depending on the design time line is depicted in Figure 2.4. As the design proceeds, the freedom to make changes is further constrained. A change in the advanced design phases can be very expensive to implement especially in the case when it affects one or more design elements of the design solution on which it has been based.
– The execution of changes leads to new results, which in turn are used by different activities which in turn lead to new results. This can led to a "disconnect" or "inconsistent" problem, which raises the following important questions: which data are used to get which results? Who performed which procedures and when to get specific data? These questions might be easy to answer for a limited number of procedures done by the same person, but very difficult with respect to the ship design as the design process includes a large number of activities, which are done by persons related to different disciplines and might be located in geographically dispersed places.

Summarized, after analyzing the above described aspects, it can be stated that the difficulties of the early ship design stage are existing due to reasons associated to data, data-interaction, activities, activities-dependencies in addition to the interaction between the data and activities. Therefore, the supporting of these aspects can facilitate the early design process and lead to significant advantages regarding the entire design process. This conclusion represents the core consideration of the developed approach within this research.

2.2.3. Early Design Characteristics

Ship design is characterized with some issues, which can be considered as fundamental aspects associated generally with the design process and especially with the early phase. The efficient support of these aspects which are: ‘Decision Making’, ‘Data-Reuse’, ‘Data Exchange’ and ‘Design Constraints’ will lead to considerable benefits regarding early design process. They are described in the following sections with two objectives in mind: firstly, to get more insight into the early design process and secondly, to highlight the importance of supporting these aspects in any approach under development.

2.2.3.1. Decision Making

Decision making, alternative and trade-off studies are three most important interconnected activities the designer faces in the early design process. Alternatives are solutions of the design problem. There is not a unique solution, but rather, many solutions/alternatives are developed for each requirement related to the design of a new ship. Many decisions are made during the design process until a suitable design solution is achieved. These decisions in the early design stage can be characterized as partial decisions. The achieved design alternatives in turn are subjected to selection procedures, which are widely called as trade-off studies. Trade-off studies can be considered as decision-making processes at an advanced level. Thus, the decisions in the ship design process are made in different levels [110] and from different persons, for example each designer, who is responsible to perform a specific design task(s), makes decisions related to his/her task(s), whereas the decisions associated to the trade-off studies are made by the project manager or other person who is responsible for providing offers to build new ships. This led Nowacki [116] to define the ship design process as a decision making process. Thus, the supporting of this aspect can results in many benefits for the effectiveness of the overall design process. In general, the making of any decision at any level requires adequate information about

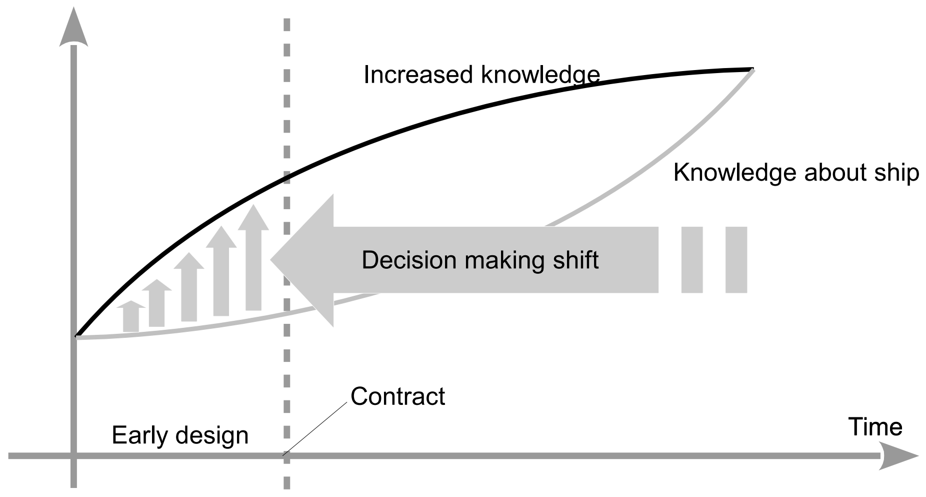


Figure 2.5.: Efficient decision making according to [39]

the related subject in addition to the need of specifying the criteria, which control the making of the decision. Thus, supporting information and criteria issues assists designers to adopt right decisions. At the early phase, these criteria can be divided into two principle categories: firstly, the constraints related to the owner's requirements such as service speed, sea endurance, etc., and secondly, criteria related to fundamental design issues like rules, which govern the calculation of stability. However, it can be stated that:

- making decision process can be efficiently supported by providing more details about the addressed subject(s) at the time of making the decision. This will help the responsible engineer (designer, manager, etc.) to make the right decision.
- during the design process, the success in shifting the right decisions to the early design stage will result in significant benefits regarding the decreasing of the potential design failures/problems, which can be raised in the following design stages.

The diagram depicted in Figure 2.5, shows the positive effects of increasing the details about the designed ship at the early design stage on the efficiency of the decision-making process and on the successful of shifting the decision-making process from the advanced design stages to the early ones. Therefore, more details should be associated with the design elements earlier in the design process in order to enhance this important aspect. Thus, Gale [101] pointed out that to support the trade-off studies in the early stage, it is an essential issue to provide the important differences between the different alternatives in order to support project mangers to adopt the right decisions.
In summary, it can be stated that the success of enhancing the decision making, alternatives and trade-off studies aspects at the early design stage can support the overall design process. This

very fundamental fact serves as one guideline in the approach developed in this research.

2.2.3.2. Data-Reuse

Most difficulties faced by the design team when starting a new design project are the low level of available design-data and the time pressure in which they have to achieve all related design activities, evaluations and to make many trade-off decisions. This fact urged designers (in order to accelerate the early design process) to rely on similar previous works to re-use their data completely or partially with some modifications regarding the specifications of the new design project. Many design tools have developed functionalities to support the methodology of reuse design-data, but the main focus in most developments is the advanced stages of the ship design such as the approach introduced by Malheiros for the reuse of design and production data in operation and maintenance of warships [107]. Thus, the reuse procedures in early design phase are still faced with many difficulties and obstacles that limit their effectiveness. Some of these challenges are:

- The data of the previous design projects are not saved in a way which enables the designer to use them effectively in the early design stage.
- Often, only the data, which represents the last results of the previous studies are saved. Therefore, the designer will not be able to gain an insight into the work-flow of the processes, which have taken place to get these saved satisfactory results.
- The cooperation between different design disciplines such as system engineering, hydrodynamic, etc., which have different perspectives regarding the ship design process, to achieve the studies at the early phase, lead to save the data at this stage with respect to these different points of view. This makes the reuse of the saved data from other disciplines in future design projects difficult.
- Wrong design data at this stage are very costly, due to the fact that the freedom to make changes is reduced as the design process proceeds (see Figure 2.4).

E. Moredo et al. [111] have developed a methodology for reuse data in the tender phase. The methodology is based on the combination of a Functional Breakdown Structure (FBS) and a System Breakdown Structure (SBS) in a way that enables to structure, document and reuse design data. FBS helps the designer to get the right requirements from the client and save the design data in a structured functional database. SBS is used to enhance the documentation of design data and to improve the data-exchange between shipyards and system-integrators. They state that the implementation of this methodology will help the design team to address many weaknesses related to this stage. Other methodology has been developed by M. Toyoda et al. [144], who have tried to accelerate the generation process of a 3D-model at this stage by reusing of standard parametric structural components and assemblies stored in libraries and classified by ship type. Respecting the previously mentioned importance and difficulties, the support of a data reuse-concept (as it will be seen later) is one of the advantages, which is part of the approach developed within this research.

2.2.3.3. Data Exchange

The participation of many partners: shipyards, classification societies, design offices, suppliers and model basin in ship design projects raises the need to exchange data in order to efficiently achieve the desired design mission. In general, the difficulties associated with the data exchange are mostly emerged from the variety of data formats, which are used by the design participants. This can be attributed to the diversity of their tools, which are appropriate for their own works (tasks). However, there are some formats like IGES [91], STL [139], VRML [33], XML [1], STEP [133], etc., which have been introduced and can be considered (to some extent) as general formats to represent the design elements such as hull-form, compartments, etc. [48]. The data exchange concept regarding the early design phase can be discussed under two perspectives:

- The exchange in the same stage between design team members and tasks. This point of view can be addressed as a collaborative management issue.
- The exchange with other design stages. This raises the questions about the compatibility of the data at this stage with that in the downstream stages, and the integration level of the tools used at the different design stages.

However, in order to exchange data between different partners, "means" are required. According to J. Pradillon et al. in the design methods report of ISSC 2012 [92], emails are still the most commonly used media between different partners to exchange product model data. B. Kassel et al. [57] raised another aspect regarding the data-exchange, they stated that the association between exchanged "geometry" and "metadata" is still weak. This poses questions about the value and size of the exchanged data and raises the needs to use formats which support the relation or dependency between "shapes" or "geometry" and associated "metadata". Considering these facts the data exchange aspect is one of the advantages of the approach developed within this dissertation.

2.2.3.4. Design Constraints

Ship design in general is performed with respect to restrictions and limits, to which is referred as design constraints. In fact, the design Constraints can be issued by:

- International Maritime Organization (IMO): its regulations, which concern the safety, security and marine pollution issues [26], cover the entire ship life-cycle (from design to the disposal). Safety Of Life At Sea (SOLAS), the Load Line Convention (LLC) and the Intact Stability Code (IS-Code) are examples of IMO-regulations, which have to be satisfied during the design process.
- Classification societies: They set their technical rules for the construction and operation of ships. These rules have to be observed. Some of these rules are issued within the scope of "International Association of Classification Societies" (IACS) [25] as "Common Structural Rules" (CSR) (for tanker and bulker) e.g. the CSR for double hull oil tankers.
- Flag state: Ships have to satisfy the regulations (related to inspection, certification, safety, pollution prevention) issued by the states under whose flags they are registered.
- Ship owner: ship mission, service speed, etc., are examples of ship owner's requirements, which formulate constraints (limits) during the design process.

All constraints and regulations have to be respected in order to create feasible design solutions, which meet the owner's and other requirements. The great diversity of ship types impose a significant expansion on the special requirements in order to take into account the particularities of the different ship types, such as Container-, Roll-On/Roll-Off-ships, etc. [102]. However, there are some requirements which can be considered as common requirements relevant to almost all ship types. The clarification and the correct understanding of the requirements at the beginning of the design process is an essential issue [60]. This can be supported through a good relationship and interaction between the owner, classification society and the design team. The design requirements, as it has been mentioned previously (see section 2.2.3.1), represent the criteria on which the decision-making in the design process will be based. Indeed, the perception of these requirements is not uniform, C. Koenig et al. [101] classified them in technical like endurance, service speed, manning, accommodations, etc., and commercial such as operating agreements, ship financing, etc. Whereas B. Oers [117] introduced another point of view, he grouped the requirements into "negotiable" and "non-negotiable". He pointed out that some requirements, especially in the case of designing warships, must be complied with whereas other requirements can be simply adapted to enhance other requirements. However, these requirements are not isolated from each other, they are directly or indirectly interdependent and interrelated and can be contradictory or incompatibly. This imposes the need to reach a consensus on these requirements in order to create satisfactory design solutions. In the approach developed within this dissertation, design requirements represent the specifications, which form the constraints of new design projects.

2.3. Design Methodologies

Many papers, which generally address the ship design process and particularly the early design phase have been published. The purpose of the following survey is to give an insight into most published ship design approaches and methodologies in order to serve as a background for the approach developed.

- Oers et al. from the Delft university of technology [147] have developed an approach to help the naval architect to identify the suitable solutions at the early ship design stage. Their approach is based on two algorithms, a "packing-algorithm" and a "search-algorithm". These two algorithms will automatically generate a large number of "feasible ship designs", which support the naval architect to make a decision while selecting the preferable designs. Six types of 3D objects (design elements) are used to form the design-models [117]. Some spatial and logical rules such as "overlap-rules" are used to pack these objects together in sequential manner, which can be changed enabling the designer to control the packing process. Some criteria are used to assess the resulted "packed" objects in order to generate the set of feasible ship designs.
  Since the generating process of the feasible 3D designs requires a lot of time, Oers et al. tried in the advanced approach presented in [118] to reduce the computational effort by simplifying the packing approach to '2.5D'. They state that this simplifying procedure increases the generating speed by at least a factor of three.
- The Intelligent Ship Arrangement (ISA) [67] is another approach, which has been developed for the U.S. Navy by the University of Michigan. It has been developed in a way that enables the user to capture U.S.Navy design rules, regulations and other important naval architectural resources. ISA aims to help the designer in the early design stage by providing a number of trade-off studies for the ship general arrangements. Multiple arrangement solutions will be generated, compared and optimized in a rational process. The methodology adopted by ISA to create the general arrangements can be divided into two stages. The first is to allocate the design elements to the predefined structural zones and the second is to arrange these spaces within the zones to which they have been allocated [53]. Gillespie et al. [77], [78] introduced an approach to increase the efficiency of the "allocation"-stage of ISA. The main objective of their approach, which is based on a "network theory-based partitioning method", is to decrease the number of the detrimental relationships between the design elements allocated to the same structural zone.
- In the literature many papers can be found, which illustrate the Design Building Block (DBB) approach like [36], [37] and its applications such as [34], [58]. The DBB approach has been intended to improve the early ship design studies. It has been developed by University College London and implemented as an incorporated part of the design system

GRC's PARAMARINE [29]. The methodology of this approach depends on a "functional" breakdown of the design into blocks. The designer can then aggregate blocks together to form which called "Master Building Block". These Master Blocks can be used directly by PARAMARINE to perform the necessary naval architectural calculations. DBB helps the designer to conduct early ship design studies by means of the functionalities offered by the PARAMARINE software system and based on 3D integrated design representations.

- A holistic ship design optimization methodology has been introduced by Papanikolaou [121], [35]. The methodology is based on the addressing of the ship regarding the system approach. In this context, the ship is considered as a complex system composed of many subsystems and their components such as the subsystems for the energy generation, accommodation for the crew, etc. Papanikolaou argued that the ship design optimization must address the whole ship's life-cycle from the concept/early design to the scrapping/recycling. In this context, he defined the "optimal ship" as the outcome of the system ship for its entire life-cycle. Therein, the design objectives, constraints and optimization criteria represent conflicting requirements. This can be attributed to the different points of view and interests of the design partners: shipowner, ship builders, classification societies, etc. and must be regarded by the optimization procedures.
- The formulation of the ship design problem presented by Nowacki [115], [116], represents the roots of the previous design optimization methodology introduced by Papanikolaou [121]. Nowacki [115], as it is shown in Figure 2.6, defined the elements of the ship design process in general (horizontally) as well as by taking into consideration the modern computer modeling strategies (vertically). In general, he specified the input and output of the design process as follows:

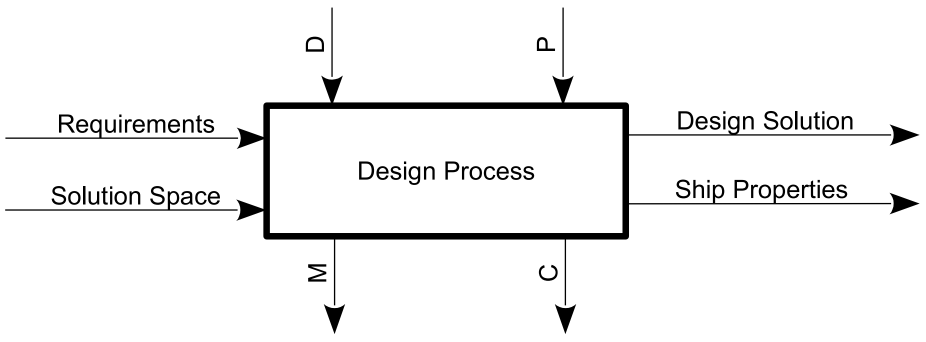


Figure 2.6.: Ship design Process with respect to [115]

  - Requirements: such as desired performance, safety constraints, etc.
  - Solution Space: it specifies the limits of the design solutions
  - Design Solution: it should meet the all "Requirements" in addition to the respecting of the "Solution Space"
  - Ship Properties: the description of the design properties, which is the basis for the production.

By respecting the computer modeling strategies, he defined the design process elements as follows:

  - D: the design parameters, which can be controlled by the designer, e.g. main dimensions
  - P: the design parameters, which can not be controlled by the designer, e.g. port water depth
  - M: "measure of merit", like, economic criterion.
  - C: Constraints, regarding e.g. intact and damage stability

Therein, the ship design process mission is to assess M and C at a given state of P and at a set of states of D. In this context the "Optimal Design" is the one with the best value of M.

2.3.1. Integration Strategy

The integration strategy aims mainly to bridge the gap between ship design tasks to be efficiently performed. It has been adopted from almost all software systems concerned with ship design. The integration of all phases including the early stage is the main objective of the following reviewed design methodologies and consequently to the approach introduced in this dissertation.

- An integrated design and multiobjective optimization approach to ship design has been introduced by Papanikolaou et al. [120] and Harries et al. [84]. In addition to the adopting of the previously described optimization methodology (introduced by Papanikolaou [121], see section 2.3) in the presented approach, the simultaneously nature of the ship design process has been taken into consideration, see Figure 2.3. Therefore, a synthesis model of Computer Aided Engineering (CAE), which represents an integrated design software platform, has been developed. Therein, an integration of many methods and tools for the design, calculation, optimization and data-exchange has been conducted. The implemented approach has been applied to the design of an Aframax tanker. An integrated software platform of the Friendship-Framework [24], Poseidon [28], NAPA [27] and Shipflow [30] tools has been achieved. Parameters related to the payload, steel weight, strength, oil outflow, stability and hydrodynamics have been simultaneously examined for many variants against determined requirements.
- An integrated approach to manage the ship general arrangements has been introduced by A. Cebollero et al. [49] by developing a Foran General Arrangement tool. They tried to overcome the lack of information details at the early design stage by building a product model for the ship general arrangements from the beginning of the design process. This product model has been used as a basis while creating compartments and other general arrangements. To facilitate these definitions, a combination of two different environments "2D" and "3D" has been adopted. A. Cebollero et al. [49] stated that this combination enables the user to design the general arrangements in "2D" or "3D" models in accordance to the available data at the time of creation.
- It is almost the same approach proposed by Eriksson et al. [71], they also show how it is useful to use a total model of a ship from the beginning of the design process. This model is called a 'common reference model' and should be used as a reference model and updated within all phases of the ship design project.
- J. Charles et al. [66] argued the benefits of the integration of ship design stages in order to support the "Design For Production (DFP)" concept. Therefore, an integrated platform has been developed. This has been achieved by the integration of two software systems Paramarine [29] and FORAN [49] in a way enabling each system to get the data from the other system's database. The integration has been made in three steps, in which the designer is able to transfer design data between the two systems and use the appropriate system regarding its capabilities with respect to the performed design task.
- An integrated ship design system "E4" has been developed by the Technical University of Hamburg-Harburg (TUHH) with the cooperation of the Flensburger Shipyard (FSG) [98], [45]. E4 has been especially intended to be used in the early design phase. Therefore, it offers many tools to perform early design tasks, such as hull form development, intact and damage stability, etc. All these embedded tools are based on a common data base. Its features take into consideration the team-work nature of the early design process by

providing the multi-user functionality [114].

- Mizutani et al. [70] developed an approach to integrate and optimize the design process based on a single NAPA 3D Model [65], and their own shipyard's design tools. They state that the usage of a single 3D product model will improve the consistence of the data of various design disciplines, it will at the same time support the efficiency and accuracy of data at the early design stage, and thus will improve the overall design process.

In summery, it can be stated that the common denominator between almost all methodologies, which adopt the integration strategy, is the need to build upon a ship product model. This raises the question about the ship product data structuring, which is discussed in the following.

2.4. Ship Product Data Structuring

Firstly, it is necessary to distinguish between data and information. Data themselves have no meaning. Data are raw, unorganized and useless material. Data become information when they are structured in order to be useful and have a meaningful description of the addressed domain [123]. For example the number 200 as a data can be become an information when it represents the length between perpendiculars of a ship. With the main objective of supporting the integration strategy within the ship design process, many standards and additional generic resources can be found in the literature, which concern the structuring of data related to the ship product. Developments primarily relevant to the topic of this thesis are: ISO 10303, SFI and HCM.

2.4.1. ISO 10303

The STandard for the Exchange of Product model data (STEP - ISO 10303) is an international standard which has been developed to provide a neutral means to exchange product data within the product life cycle independent of any specific software [133]. Shipbuilding is one of the different industrial areas covered by STEP applications protocols (APs). They offer a detailed description of a product (such as the ship) suitable for originating a product information model, appropriate to be used as a basis to establish an integrated database, such as the database developed by Doese [54] to support the concurrent engineering concept within the ship design. These APs were the basis for many developments regarding the ship design and production process, such as the integrated data model developed by Atlantec Enterprise Solutions [23] depending on the information descriptions provided by these APs. The developed data model by Atlantec serves as a neutral storage for the different data retrieved from different CAD-systems. These data are the basis for many validation and quality control procedures, which can be performed in the ship design and production process [96].
According to STEP, ship product data are divided into a number of domains like ship structures, distribution systems, etc. as shown in Figure 2.7. Each of these areas are described by one or more application protocols. For example, three APs, which are AP 215, AP 216 and AP 218, are developed to characterize the ship structural envelope.

AP 215: Ship Arrangements

This application protocol has been developed to support the downstream design stages (and not the early stage) in the product (ship) life cycle: detail design, production engineering, etc., for both commercial and navy ships. It provides a detailed definition (property, function, geometric representation, etc.) of the internal subdivision of the ship into compartments, zones in addition to the relationships between these spaces. Furthermore, it offers a detailed description of many aspects associated with the ship arrangements like the definition of cargo, damage stability analysis, loading conditions, etc. [113].

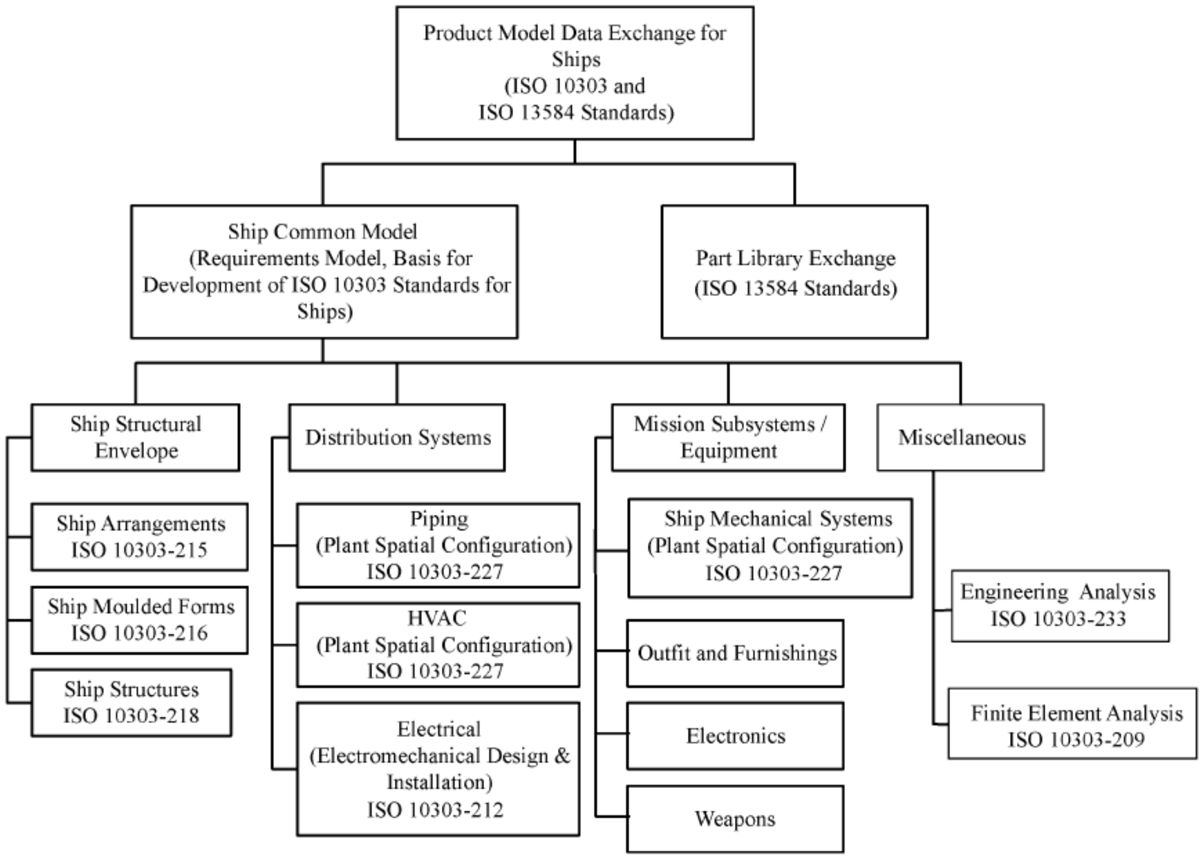

Product Model Data Exchange for Ships (ISO 10303 and ISO 13584 Standards)
Ship Common Model (Requirements Model, Basis for Development of ISO 10303 Standards for Ships)
Part Library Exchange (ISO 13584 Standards)
Ship Structural Envelope
Ship Arrangements ISO 10303-215
Ship Moulded Forms ISO 10303-216
Ship Structures ISO 10303-218
Distribution Systems
Piping (Plant Spatial Configuration) ISO 10303-227
HVAC (Plant Spatial Configuration) ISO 10303-227
Electrical (Electromechanical Design & Installation) ISO 10303-212
Mission Subsystems / Equipment
Ship Mechanical Systems (Plant Spatial Configuration) ISO 10303-227
Outfit and Furnishings
Electronics
Weapons
Miscellaneous
Engineering Analysis ISO 10303-233
Finite Element Analysis ISO 10303-209

Figure 2.7.: Initial Ship Application Protocols, [113]

AP 216: Ship Moulded Forms

Ship moulded forms and hydrostatic properties are the subject of this application protocol. It fosters the geometrical representation of the different ship moulded forms like: ship hull (mono- and multi-hullforms), rudder, propeller and other appendages, by addressing their shape and design parameters. AP 216 has been developed especially to enhance the geometry and hydrostatics exchange of the hull moulded form, from the beginning of the design process and between different design partners [112].

AP 218: Ship Structures

This part of ISO 10303 has been developed to improve the data exchange of ship structures between different organizations such as classification societies, shipyards, etc. in the design, production, maintenance and inspection phases during the ship lifecycle. General characteristics relevant to ship structures, coordinate system (global, local), weights and centers of gravity of the structure parts and assemblies, etc. are defined and addressed in the AP218 [5].

Summarized, the detailed descriptions of ship product data within the relevant STEP-APs provide generic resources to develop ship product models independent of any software system. Therefore, the definitions introduced within these APs and additional generic resources have formulated the basic background of the ship information model developed within this research as it will be seen later.

2.4.2. Senter for Forskningsdrevet Innovasjon (SFI) Group System

SFI group system is a decimal coding system. It is used as a classification system in the maritime and offshore industry [2]. SFI has been developed mainly to support shipyards and other

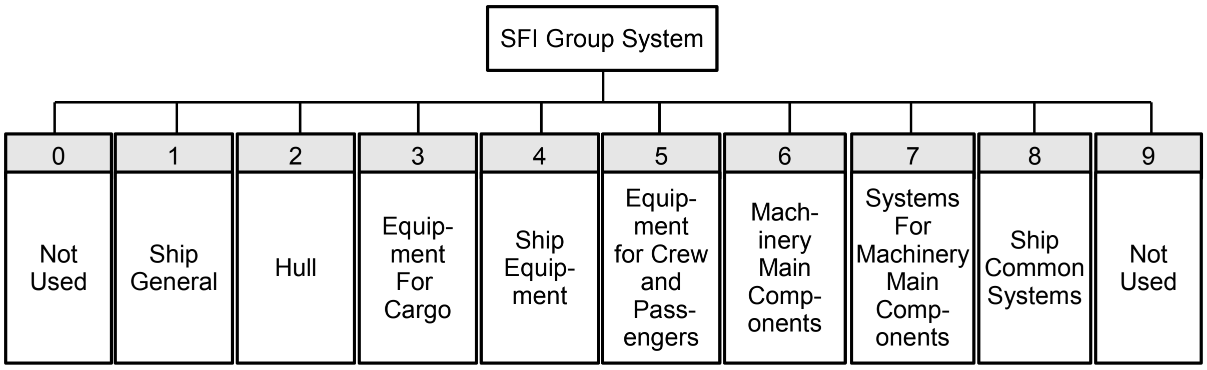


Figure 2.8.: Main Groups of the SFI [153]

shipping companies to manage their operations like estimates of cost of materials and work hours production, purchases, maintenance and repair planning, etc., during the product lifecycle. The information associated with the ship or offshore structure are grouped with respect to functional aspects, like the ship's systems, machinery, etc. are given numbers respecting their functions (purposes). According to SFI the ship/offshore components are classified in ten main groups, from which just eight (from 1 to 8) are used. This enables the companies to group disregarded components (user defined components) in the unused main groups (0 and 9). Figure 2.8 shows the main eight groups of the SFI group system. Each main group is divided into ten groups. Each one of these groups in turn consists of ten subgroups [17]. In order to give an insight into the coding system within SFI, the numbers associated with some groups and subgroups of the second main group 'Hull' are listed in Table 2.1.

SFI-Num	*Component Name*	*SFI Class Type*
2	Hull	Main Group
20	Hull Materials, General Hull Work	Group
201	Hull Materials	Subgroup
202	Transportation, Sorting and Storage of Hull Materials	Subgroup
203	Blasting, Shop-Priming, Rolling and Cleaning of Materials	Subgroup
204	Testing of Tanks, Bulkheads	Subgroup
205	X-Ray and Ultrasonic Testing of Hull Parts	Subgroup
206	Template and Mould Loft Work	Subgroup
207	Joining of Hull Parts Afloat	Subgroup
21	Afterbody	Group
211	Shell Panels, Separate Shell Plates	Subgroup
- -	- -	- -
22	Engine Area	Group
221	Shell Panels, Separate Shell Plates	Subgroup
- -	- -	- -
23	Cargo Area-Hull Small Vessels	Group
- -	- -	- -
29	Miscellaneous Hull Work	Group
- -	- -	- -

Table 2.1.: Groups and Subgroups of the Main group 'Hull' in SFI

2.4.3. Hull Condition Model (HCM)

The HCM has been developed within the EU project ‘Condition Assessment of aging ships for real-time Structural maintenance decision’ (CAS) [47]. During the operation of ships, the structure is periodically checked by classification societies according to the international regulations. Therefore, measurement campaigns are periodically performed (typically by specialized companies) to record the steel thickness, coating conditions, etc. These measurements are issued in reports, which are checked by the class surveyor in order to request repairs or not. Jaramillo et al. [61] stated that this condition assessment process is typically performed manually, through which the measurements reports can only be exchanged in tables, pictures and text forms. The difficult, time consuming, error-prone and the lack of Information Technology (IT)-support nature of this process are the main motivations of developing the HCM. HCM is a structured data model, which has been implemented by means of the Extensible Mark-up Language (XML) [1] as a standard electronic exchange format. It has been mainly introduced to support the integrated electronic nature of the condition assessment process. Therefore, the data structures (Components) of the HCM are strictly confined to the requirements of this process [62]. HCM contains the following top level data components, each of which has been implemented in a XML-module:

- Generic Product Data: The general data of a product, which are not shipbuilding specific e.g. the data constructs, which describe the geometry properties.
- Generic Ship Data: It includes the data constructs, which define the general shipbuilding information e.g. frame spacing tables, principal dimensions, etc.
- Thickness Measurements Data: Its data constructs concern the definition of the data collected during the measurement campaigns e.g. plate strake based measurements, cross section based measurements, etc.
- HCM Model: It provides the ability to assemble the data sets required to define the Hull Condition Model (HCM) of the vessel after the measurement campaign.

2.4.4. Tool-Specific Ship Product Models

In the ship design process, software systems are utilized to perform a great diversity of design tasks. In fact, these software systems have their own points of view regarding the structuring of the ship product data. These data structuring perspectives are dependent on the design tasks they have developed to serve. As an example Figure 2.10 shows the grouping of the ship product

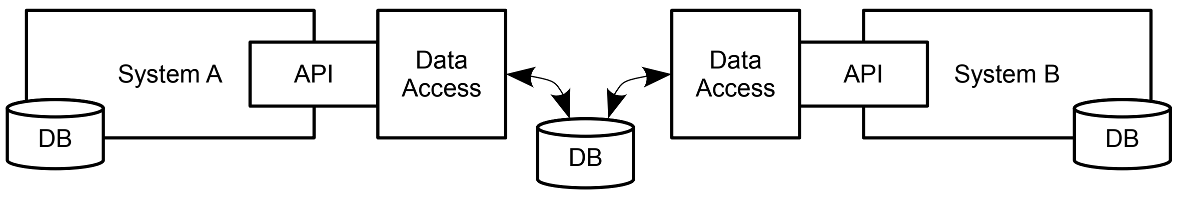


Figure 2.9.: Application Programming Interface (API)

data within the AVEVA CAD system [108] regarding the description of the ship’s steel structure. The ship represents a "root element", which has five "child elements" (Material, BarSection, Block, NotchDefinition, HoleDefinition). These "child elements" have their own "sub elements" (see Figure 2.10) and by this a hierarchy is built. These "elements" have their own properties, which are used to describe the ship’s steel structure. On the other hand, the integrated nature of the design tasks, imposes the data-exchange between different software systems as it has been

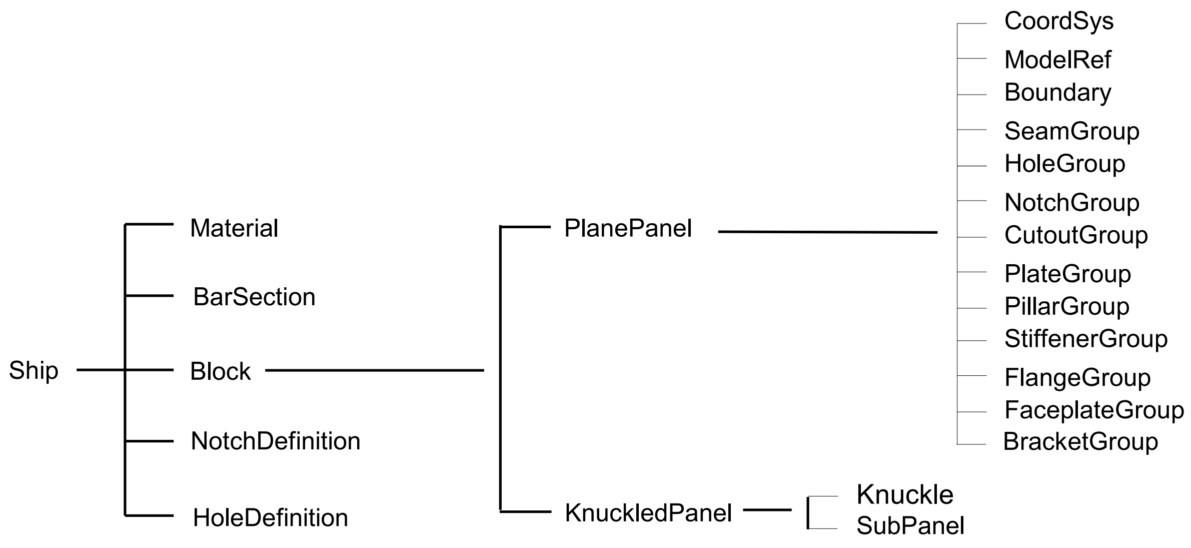


Figure 2.10.: Steel structure product data in AVEVA system, [108]

described above. This fact gives a reason for the growing needs of Application Programming Interfaces (APIs) [31], which is used as a means to transfer the data between different systems. As it is shown in Figure 2.9, an API technique can be used to transfer data between the two systems A and B. Thereby, the API plays an intermediary role to achieve this goal.

2.5. Need for Research and Objective Target

By taking into consideration the importance and effects of the early design stage on the whole design and production process, it can be argued that this phase has not acquire sufficient attention and research. This conclusion has been stated by many authors. According to Andrews, as he mentioned in his 'state of the art report' [59], this can be attributed to the reason that most of the shipyards or design offices depend on their previous designs to build the new ones. However, the necessity to continue researches regarding the early stage can be interpreted as follows:

- Most of the design methodologies (see section 2.3), which have been developed to enhance the ship design process, are used or they have been introduced to be used in the downstream design phases and not in the early stage.
- A literature review reveals that most of the approaches which have been developed specially for this phase (some of them has been presented previously in section 2.3), are intended to be implemented for:
  - Specific ship types (mostly navy ships)
  - Specific problems related to specific shipbuilding companies

  This raises the need to develop solutions regardless the type of the ship and the specific problems of companies.
- Many of the published developed approaches, which adopt the integration strategy, have been implemented for specific purposes such as the optimization of the design, see section 2.3.1. Therefore it is seen that the applying of the integration strategy with more insight into the early design specifications (complexity, drawbacks, etc.) can lead to significant benefits regarding the entire design process.

2.5.1. Ship Early Design Process

Considering the previously mentioned conclusions of the necessity to continue research concerning the ship early design and before clearly defining the main objective of this thesis in the next section, a brief description of the early ship design process regarding its properties, capabilities and tasks (as it is seen in this research) is illustrated in the following:

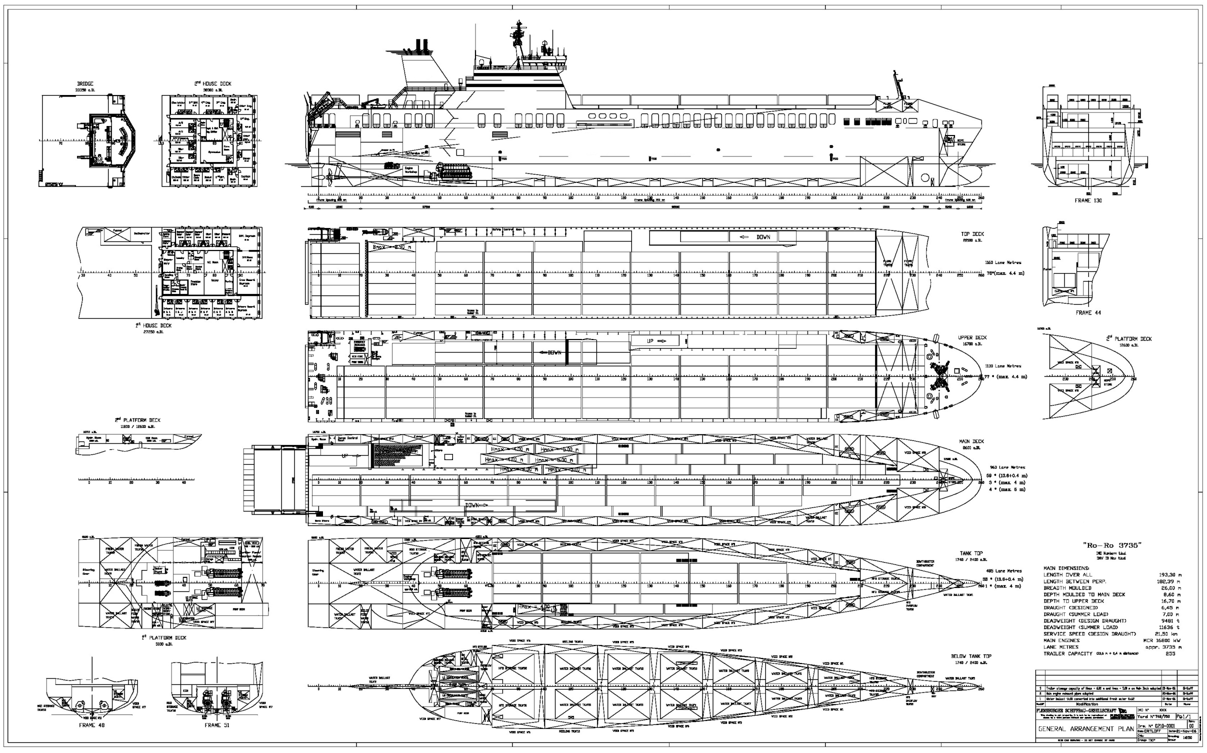


Figure 2.11.: 2D General Arrangement of a RoRo-Ship (Courtesy of FSG)

- Design constraints (see section 2.2.3.4) whether they are issued by IMO, classification societies, flag states or ship owners should be respected within the studies performed very early in the ship design process.
- Early ship design process should provide several design solutions, which are used by a shipyard to get new contracts. The increasing of data-detail level as well as the efficient definition of these design solutions is an essential issue to be respected in the early design process. Thereby, the possibility to address design solutions with the definition of the exact difference between them can lead to significant benefits regarding the data-reuse aspect in future design projects.
- The above mentioned design solutions should be based on results gained from an efficient performing of several design tasks within the early ship design process. The tasks, which have to be performed, can briefly be mentioned as follows: generate hull form, hydrostatics, compartmentation, loading conditions, check intact and damage stability, weights- and capacities-calculations, freeboard, hydrodynamics/powering, longitudinal strength.
- In addition to the above outlined tasks, ship components such as machinery, outfitting, etc. have to be defined and taken into account within the early design studies. Considering ship components properties (weight, location, etc.) at the beginning of the design process helps to increase the accuracy of the studies and to decrease inconsistent problems (which can be raised in the next design stages) between the components and their related ship spaces.

- The results of early design tasks (such as defined components, spaces into which a ship-form is subdivided, etc.) are traditionally represented in 2D-general arrangement drawings in the early phase (such as the one depicted in Figure 2.11 for a RoRo-ship), and in 3D-models in the downstream stages. Therefore, it is seen that the study environments (2D and 3D) in the early design stage have to be compatible with the other stages in order to increase the efficiency as well as to support the integration and data-reuse aspects between the early and the following design phases.
- The efficient definition of design elements such as ship-form, space elements, components, etc., i.e. the identification of their properties and relationships at the early ship design process leads to significant advantages regarding the overall design process. For example, the space elements (compartments, rooms, zones), which play a central role in the design process have to be defined by means of their properties such as volume, weight, surface area, boundaries, etc. in addition to the definition of the association between them.
- Early ship design tasks are interdependent, i.e. the performing of some tasks is totally or partially based on the results of other tasks. Therefore, the interaction between the different design tasks should be addressed carefully. The dynamic nature of the early design process has to be insured.
- The above mentioned dynamic nature issue has to be managed effectively. Therefore and regarding the significant developments in the management concepts within the system design engineering fields, it is seen that the applying of these concepts like configuration management, change management, etc. to the ship design process especially in the early phase can support the dynamic nature of the design process and improve many of its drawbacks.
- The overlapping nature of the design elements and the interdependencies between different early design tasks imposes needs on the efficient management of changes, which can be requested and performed during the design process. The management should be able to externalize the implications of changes to the right persons at the right time.
- The early design task properties (input, results, applied tools, etc.) have to be combined (related to each other) effectively in order to support the consistence as well as data-reuse aspects.
- Several concepts such as approval, status, etc. can be applied to enrich the design properties and accordingly to support the early ship design process.
- The teamwork nature of the ship design process is another topic, which has to be supported by an efficient management system of the ship product data. It is seen, that the teamwork nature can be improved effectively by applying rights management as well as collaboration management concepts.

2.5.2. The Research Objective

The developed approach within this dissertation aims to improve the ship design process by focusing on the early stage. This respects the previously extracted fact that the supporting of the early design can lead to considerable benefits regarding the entire design process. Therefore, the main objective of this research can be stated briefly as follows: the accomplishment of the studies related to the early design stage in an efficient and dynamic manner by taking into consideration all previously mentioned drawbacks, difficulties, reviewed approaches as well as derived conclusions. In order to achieve this target, the following interactions and dependencies have been seen as the basic subjects, from which the development should be proceeded:

- Data $\longleftrightarrow$ Data
- Data $\longleftrightarrow$ Process

- Process $\longleftrightarrow$ Process

The more the supporting of these aspects within the early design phase itself and between it and the downstream design stages, the more the benefits can be gained regarding the entire design process. Therefore, system design engineering concepts have been employed in order to serve this objective target as it will be seen in the following.

3. System Design Engineering

System engineering is a general framework through which large and complex projects can be addressed. It focuses on the design and management of a system [3]. It guarantees effectively the interactions between technical knowledge and human disciplines of complex projects. However, many aspects like configuration management, work management, etc. should be taken into consideration in the application of the system engineering concepts on a large system like an enterprise [140]. The applying of these concepts to the ship design process can lead to significant advantages and improve many drawbacks related to the traditional ship design process. In the following sections some of the system engineering aspects in addition to the enterprise modeling approach are presented.

3.1. Enterprise Modeling

Enterprise modeling is the description of principles, methods and goals of a collection of organizations in coherent models in order to externalize the enterprise knowledge [103]. The main

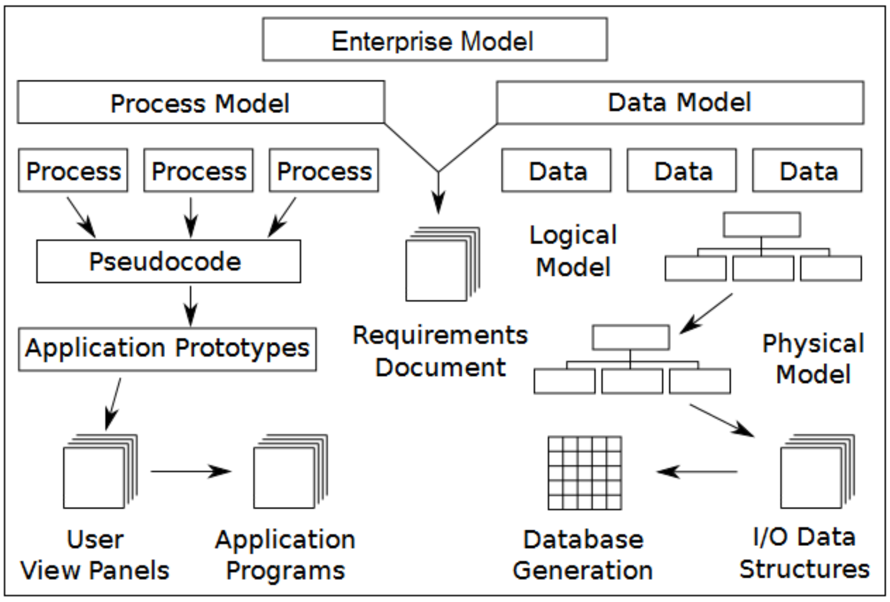


Figure 3.1.: Integrated Enterprise Model, [136]

objective of modeling an enterprise is to help persons to communicate, understand and develop efficient solutions of their business problems [141]. According to ISO 19439:2006 [7], four issues should be taken into consideration by the modeling of an enterprise:

- information: the data used and obtained
- function: the operations through the enterprise

- resource: the required sources for the operations
- organization: the responsibilities authorizations and associations through it

These different concepts can be handled in different abstraction levels regarding the needs of the designer. The modeling approach is based on the integration of two aspects: data and processes. The interaction of data and processes leads to considerable benefits like the improving of consistence, control capabilities, etc. especially for modeling complex distributed software systems [128] [85]. An example of an integrated enterprise model built up of data and process models (which are defined by taking into consideration the requirement documents of the addressed subject) is shown in Figure 3.1. In the following sections 3.1.1 and 3.1.2, these two models ("data" as well as "process") will be described in detail.
Within this dissertation the four principle concepts have been taken into consideration while keeping the main focus on preserving the efficient interaction between data and process.

3.1.1. Data Modeling

The Data Model (DM) of an Enterprise, as an information modeling technology, represents the static part of the enterprise model [150] as it is shown at the right side of Figure 3.1. The DM is the development of a conceptual representation of data elements by means of symbols and text. It describes the addressed data by giving its properties and relationships which help the reader to get a detailed impression and understanding of the addressed subject [88]. The DM forms the basis of any database application [138]. As it is shown in Figure 3.1, a conceptual representation of a DM (regardless the number of defined elements) can be developed until a database is configured. This is achieved by passing through logical-, physical- models (which can be evolved from the conceptual DM by providing more and more details about the defined elements, attributes, relationships in addition to ‘primary and foreign keys’, which describe the DM regarding database management systems) and by applying techniques like Input/Output (I/O) data structures which provides the ability to translate the physical data model to the information processing systems and accordingly to build up a database. As a result, the coherent building of DM-objects, relationships, etc. supports the data-consistency-concept and as a result different applications covered by an enterprise can be enhanced.
However, within this research, the term "Information Model" refers to the organized product, management and support data. They form a model, which on one hand includes all defined information-objects together with their relationships and on the other hand serves to be configured within a database by means of appropriate information processing systems.

3.1.2. Process Modeling

Enterprise processes are collections of activities involved within an organization. They are structured and designed to work together in order to achieve the required customer's aims and to fulfill certain goals [134]. The process model represents the dynamic part of the enterprise model. The left side of the Figure 3.1 shows that a process model can be divided into several sub-processes complementing each other to accomplish the intended objectives of an organization. The processes which have been built regarding the enterprise goals analysis can be developed, translated to the software applications, executed and controlled by users by applying programming techniques as it is depicted in the Figure 3.1.
The activities, which are related to the process addressed within the ship early design process have been modeled in a predefined-form, interacted and interdependent to each other in order to increase the dynamic nature of the process.

3.1.3. Modeling Methods

The main objective of modeling enterprises is to facilitate the understanding and analyzing of their nature, structure and potential connections with external applications [106]. This requires a proper description and employment of the concepts relevant to the modeled entities like functions, information, resources, etc. [148]. Moreover, the more the consistence and integration concepts are realized during the modeling, the higher the effectiveness of the resulted models can be achieved. This assists to efficiently provide the needed data to conduct operations as well as to manage the results. However, this raises the question about the mechanisms, notations, formalisms and methodologies through which the enterprise is modeled. Many modeling methods have been developed in recent years. Depending on Najeh et al. [106], these methods can be classified according to the mission of the modeling into: functional, decisional, organizational, informational, resources. A brief overview of some modeling techniques is given in the following, covering SADT, CIMOSA, ARIS and UML.

Structured Analysis and Design Technique (SADT)

This method has been developed by Ross [127]. It is intended to be used for data as well as process modeling objectives. It employs symbols like boxes and arrows to relate the objects to each other. In addition to SADT, this method is characterized as ICOM by taking into consideration its methodology to model processes (Input, Output, Control and Mechanism) as it is shown in Figure 3.2. It is based on the Top-down analysis concept to represent large

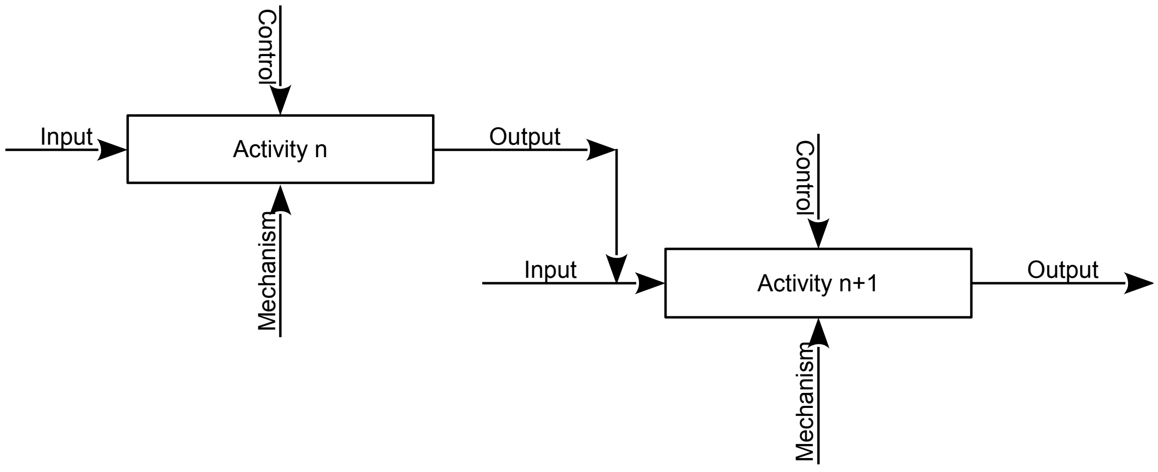


Figure 3.2.: Process Model, SADT notation

systems as depicted in Figure 3.3 [94]. The system is divided into modules, each one represents a component of the basic modeled system. These modules in turn are modeled in detail and can be divided into smaller modules until the required degree of detail is achieved. Thus, SADT provides a methodology to model systems like an enterprise in different levels of details regarding the formulated requirements. Within this dissertation, the ICOM methodology is used to model the ship early design tasks. This is due to its features, which enable one to model the activity details (input, output, etc.). Thereby, the difference between input and control aspects (which is principally defined as follows: inputs are modified within an activity whether controls are not changed) is defined in a special way regarding the ship design process as it will be seen later.

Computer Integrated Manufacturing Open System Architecture (CIMOSA)

CIMOSA was developed as an Open System Architecture to analyze and to design Computer Integrated Manufacturing (CIM) systems. It has been developed by the AMICE Consortium as

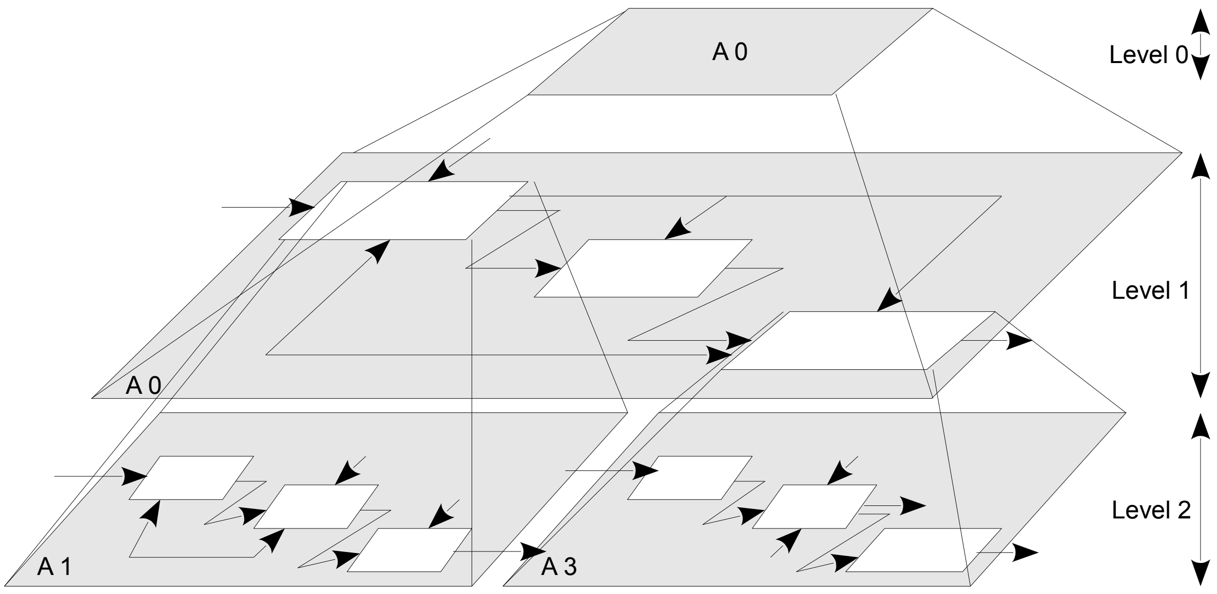


Figure 3.3.: SADT Model Structure [106]

a series of ESPRIT Projects [97]. CIMOSA main objective is to integrate the enterprise operations and information [109]. The cube shown in Figure 3.4 represents the modeling framework of the CIMOSA approach [32]. The modeling approach is based on the four concepts: function, information, resource and organization. The CIMOSA reference architecture model is structured

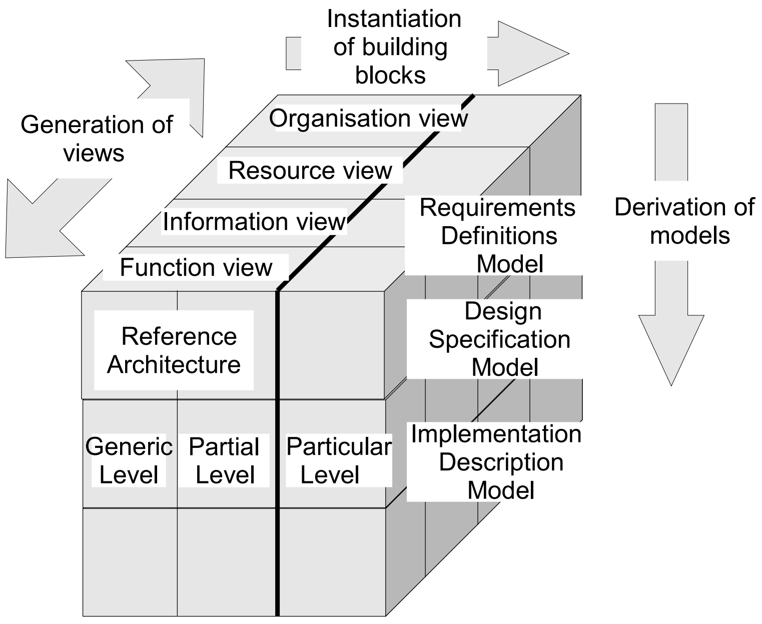


Figure 3.4.: CIMOSA Cube [32]

into generic and partial components in order to allow users to identify a particular model to work with, which can be the complete or a subset of the enterprise model. CIMOSA supports the enterprise lifecycle by covering its different levels (requirements definition, design specification and implementation description) [149]. The approach is intended to describe enterprise operations by taking into considerations different views (function, information, etc.) but it is not capable to schematize these processes or the interaction between them.

Architecture of Integrated Information Systems (ARIS)

A modeling methodology for the Architecture of Integrated Information Systems (ARIS) has been developed based on a research of Prof. Scheer [130]. The ARIS framework has been introduced to support the definition of complex business models. The basic concept is the interconnection between the different aspects while modeling an enterprise. Figure 3.5 shows the views which are taken into consideration: data, functional, organizational and the control view by means of which the integration and the relationships between different aspects of the enterprise model are established [103].

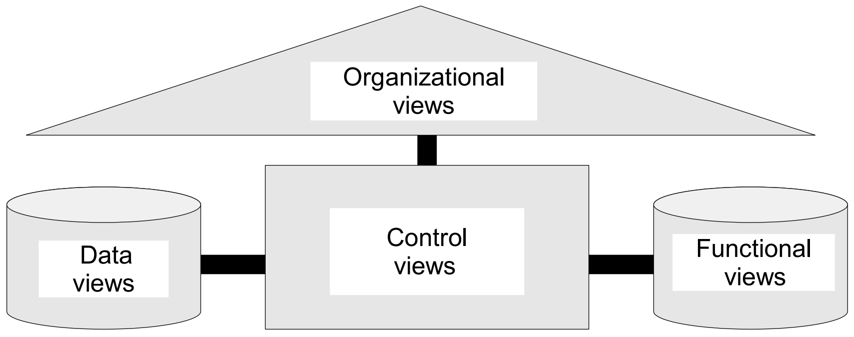


Figure 3.5.: ARIS framework [130]

Unified Modeling Language (UML)

UML represents a multi-purpose modeling language. It has been developed to meet the needs of Computer Aided Software Engineering (CASE), which can be concluded in standardizing and facilitating the means of communication between different developers of CASE-Tools [93]. UML was officially released in 1997, it has been founded as a result of the unification of three modeling techniques: Object Modeling Technique (OMT), Booch-method and Object-Oriented Software Engineering (OOSE) [46]. The diagrams offered by UML can be divided as shown in Figure 3.6 into structural types like: class-, package-diagrams, etc. and behavioral types such as: activity diagram, etc. UML-diagrams in addition to the above described ICOM-methodology are used within this dissertation to model the ship enterprise model as it will be seen later. Some UML

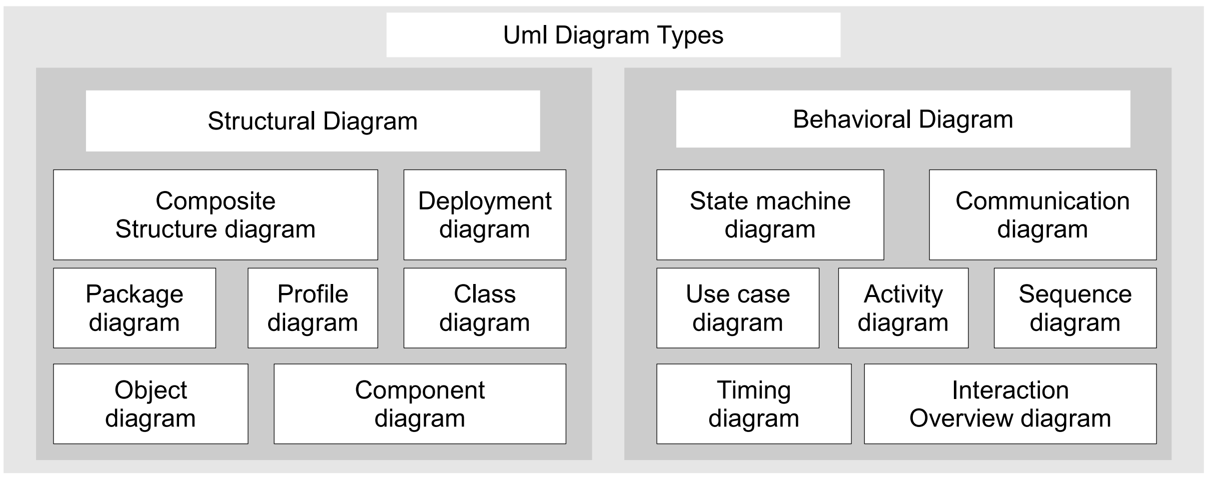


Figure 3.6.: UML Diagram Types

features which make it one of the preferred languages regarding the modeling of complex systems like an enterprise are described in the following [55], [129]:

- The great diversity of its diagram types,
- Its ability to cover both static and dynamic perspectives of systems: in addition to its capabilities to model the operational nature of the system, UML offers functions to design models consisting of objects, attributes to objects, relationships and interactions between objects.
- Object-oriented modeling technique independent of any programming language.
- UML is used as a standard modeling technique, managed by the Object Management Group (OMG) since 1997 and accepted by ISO/IEC 19505 (Part-1 [12] and Part-2 [13])
- The possibility of functional division of the modeled organizations by using UML leads to the advantages of representing the processes and the interactions between these processes within the systems.

3.1.4. Enterprise Integration

The technical perspective of modeling an enterprise is denominated as Enterprise Integration (EI) [99]. EI represents the tasks for improving the system performance by addressing aspects like: product data interchange and exchange, distributed systems interconnection and user-machine communication [122]. Petrie et al. [122] mention that the attempts to mitigate the complexity of large organizations like ship-, car- and airplanes- companies can be supported by applying EI-technique. They argue that the overall process among these comprehensive and probably distributed organizations can be fostered by EI. It can reduce the needed time, money, and resources and at the same time increase the productivity through better utilization of data and resources. This in turn leads to an improved quality of the product. The EI concept is based

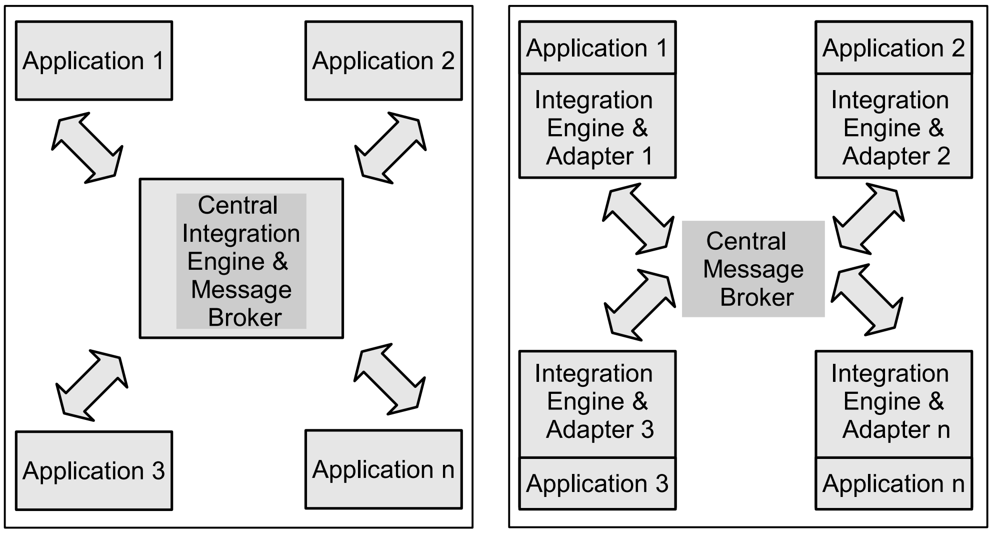

Application 1
Application 2
Central Integration Engine & Message Broker
Application 3
Application n
Application 1
Integration Engine & Adapter 1
Application 2
Integration Engine & Adapter 2
Central Message Broker
Integration Engine & Adapter 3
Application 3
Integration Engine & Adapter n
Application n

Figure 3.7.: Integration patterns: Hub/Spoke (left), Bus (right) [79]

on the abstract idea of linking all (among an enterprise) participating and involved persons, processes, systems, technologies and resources in an integrated single unit, ensuring that the right person will use the right information and right process at the right time [44]. This concept implies many topics which have to be handled carefully and correctly in order to ensure the performance improvement of any built unit [74]:

- Product data model which is a collection of structured organized data associated and related

to each other in a consistent manner. This requires a precise, in-depth knowledge of the product in addition to the administrative information which have to be addressed by the modeled enterprise.
- Process model of the operations implemented within the enterprise. It should be able to describe the enterprise processes, their interactions and combinations with each other, with information, resources and with external organizations.
- An integration platform provides a means to transfer information across the different participants and update the produced information during the internal operations or through external means and resources.

These needs can be fulfilled by utilizing database systems [76] which offer the ability to handle information transaction across different distributed sub-systems. A database, within which the required structured data are configured, can be built by applying techniques like Object Relational Mapping (ORM), which is dependent on any applied Object-Oriented Programming (OOP) language and provides the capability to convert data between incompatible systems [83]. The configured database can be connected with a variety of different applications, interacted with users, managed and updated by using an underlying Database Management System (DBMS) [125].
The main two methodologies utilized by the integration of the enterprise associated applications are depicted in Figure 3.7 according to Goel [79]:

- Hub/Spoke architecture (left sub-picture): in this architecture the integration is centralized by using a central integration engine (Hub) which is connected with the different applications within an enterprise by means of adapters (spoke). These adapters convert the variety of formats used by the different participating applications into a format understood by the Hub and vice versa.
- Bus architecture (right sub-picture): the integration in this architecture is not centralized but distributed. Each associated application has its own adapter and integration engine, which will take the information from the central messaging broker and convert it to a format understood in the related application.

Having a single central Hub in the first pattern makes the management of the related applications easier. But for complex systems with a large number of applications to be managed, the efficiency of this pattern can be decreased. Since the associated applications have their own adapters and integration engines in the second pattern, its flexibility is increased but at the same time questions about the complexity and maintainability issues are raised. These facts have been taken into consideration within the developed approach as it will be seen later.

3.2. Enterprise Management

The organization and coordination procedures applied to the enterprise activities and data like the authority and responsibility restrictions in order to achieve desired goals and objectives is termed as enterprise management [87]. This definition implies management disciplines which together incorporate to foster an enterprise model by supporting the efforts of storing, capturing, reporting, processing, visualizing and controlling the available resources efficiently and effectively [80]. Management disciplines which will be discussed in the following provide technologies and strategies to meet customer requirements like the assignment of roles, handling of changes, including versions and requirements treatment. These disciplines are not isolated but rather complementing each other as it is shown in Figure 3.8. Regarding the integration concept of "data" and "process" by modeling an enterprise as discussed in section 3.1, it can be stated

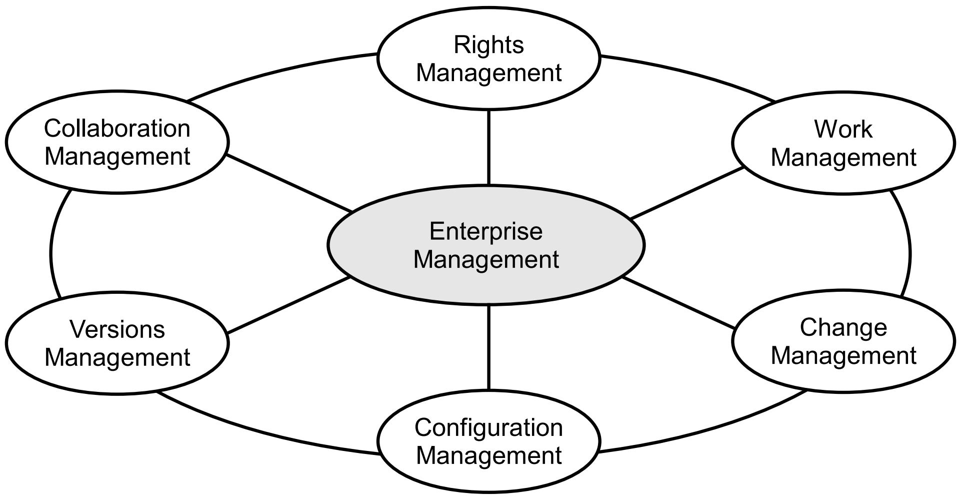


Figure 3.8.: Enterprise Management Functions

that the management should be viewed under two perspectives "data view" and "process view". These two views are interrelated and interacted to accomplish the basic objective of efficiently supported and thereby controlled product lifecycle.

3.2.1. Configuration Management

Configuration Management (CM) in terms of ISO 10303-44: 2000 [11] meets the static point of view in the data modeling. The main concern is to decide products or parts of products have to be determined as configurable items? Whereas the CM in terms of ISO 10007 [8] as well as of ISO/IEC TR 18018 [10] reflects the functional point of view by describing the activities included within the CM-process. According to A. Hass [86], this process perspective of CM has

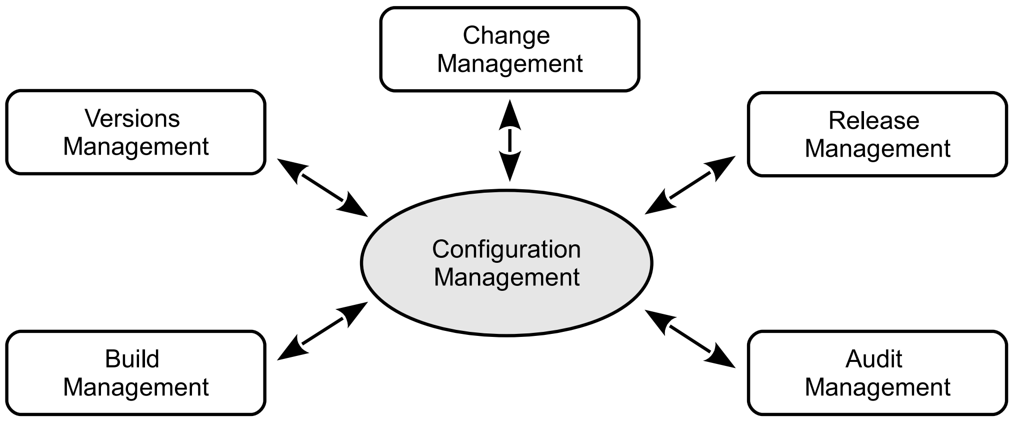


Figure 3.9.: Configuration management according to G. Marcus [81]

been emerged due to the growing demand to manage the interacted activities in the development of software applications. Activities involved in the CM process vary between different perspectives taken. Regarding ISO 10007 [8] CM process involves planning, configuration identification, change control, configuration status accounting and configuration audit activities. Therefore, CM concepts offer a stand-alone technique to control a product over its entire lifecycle from planning to evaluation passing through modification [132]. G. Marcus [81], as it is shown in Figure 3.9, stated that the configuration management plays a central role in the achievement of the traceable management of products in a mutually supportive relationship. In this dissertation and in order to consolidate the control capabilities of the formulated enterprise model, the CM term is used in a form that it acts as one of many combined, connected and interrelated management fields as shown in Figure 3.8. For example, the change management aspect is implemented in a way which enables both management fields configuration as well as change to be supported, and therefore can be used in much more flexible manner. In this context, the CM-aspect serves as a means to combine an unassigned number of data elements regarding a specific function or activity or any other identified aspect. These high-level elements-assemblies are associated with a specific status regarding a determined aspect. The example shown in Figure 3.10 represents the combination

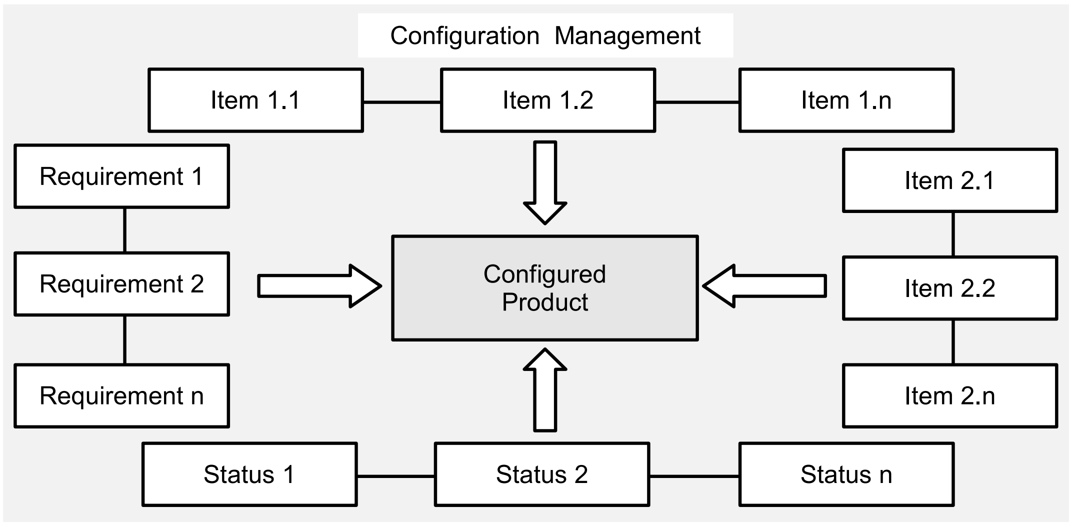


Figure 3.10.: Configuration Management

ability by utilizing the described CM-concept. Assuming that the adopted CM-concept is applied on the following existing data-elements:

- more than one version of the Item1 which can be for example hull form
- more than one version of the Item2 which can be the predicted power calculated with different methods
- more than one desired service speed specified in tender document or contract

By exploiting the adopted CM-concept it is possible to combine two versions of these two items and assigning a status of this combination by taking into consideration a specific requirement.

3.2.2. Change Management

Engineering Change Management (ECM) is the process by which a change within an enterprise is requested, evaluated regarding its impacts and attainability, implemented, verified and documented [51]. During the development of a product, changes frequently occur and must be expected at any time. This comes from the fact, that the developer's perception regarding the product is evolved during the time as well as the customer's requirements can be changed, in addition to the potential mistakes which can occur by the persons who are responsible to perform the tasks related to the enterprise [104]. Therefore, an efficient managing of changes is an essential issue and can lead to considerable benefits regarding the flexibility and homogeneity within an enterprise. However, ISO 10007 [8] identified the general features of the change control.

Considering ISO 10007 [8], it can be stated that the change management procedures should be able to:

- document the required changes: describe the proposed change, who and when it has been requested, change justification,
- evaluate the desired change: its complexity, related impacts, potential risks, technical merits and affected regulatory and statutory,
- disposition the change authority,
- implement and verify the change.

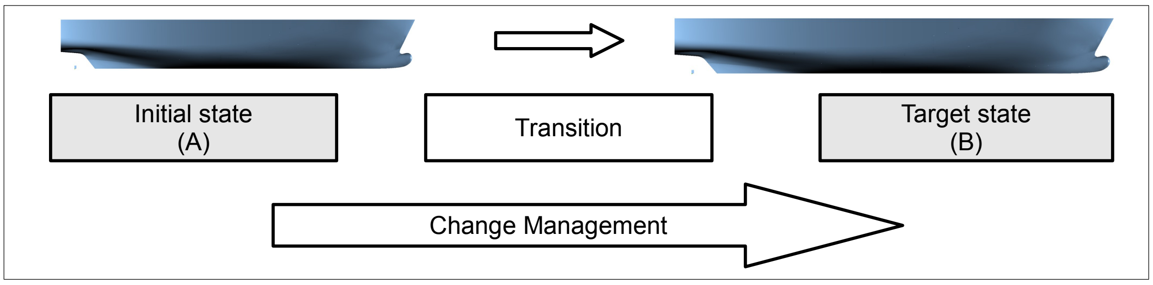


Figure 3.11.: Change management Concept

Therefore and in order to fulfill theses aspects, change management like configuration management should be treated by taking into consideration two perspectives "data"-structure to model and "process"- or "functional"-structure to execute the change. These facts have been respected by addressing the change management within this research as a control function of an enterprise management. Furthermore, it is taken into account (in addition to the previously mentioned aspects issued by ISO 10007 [8]) the numerous associations of the change management concept with other management fields. Thereby, the general objective of change management is the support of the transition from one state to other in an organized manner. As it is shown in Figure 3.11, if "A" is the current state of an item (for example ship hull) and "B" is the future (desired) state of that item, which is requested, for instance by owner's requirements or any other involved stakeholder. The change management concept is not describing the items "A" and "B" but it will document and control the procedures and associated effects which are originated (raised) by the transition from the state "A" to "B" i.e. by conducting the requested change. This is achieved by making use of additional management functions like work (activity) management.

3.2.3. Versions Management

One of the management functions which can be applied as a stand-alone technique is the Version Control System (VCS) or Revision Control System (RCS) as it is termed in many references. Version management refers to the ability to identify a unique state of a computer software, document, model or any collection of data by using a unique name or number, in addition to the possibility to keep track of the history of these states [152]. Regarding database applications [95], it is possible to apply the versioning capability on the desired objects. The software engineer is able to design the objects, or a group of objects by the definition of the database to be versionable by benefiting from the features offered by the Database Management Systems (DBMSs). Changes are always associated with new states to which is referred as versions. Therefore it makes no sense to think about changes without versions management. Version management can be embedded in other software technologies as it is done in this dissertation. In principle two general types of VCS can be implemented [119] as shown in Figure 3.12:

- Centralized (the left picture): with central repository, all users check into and check out versions from this repository.
- Distributed (the right picture): each participant has his own local repository

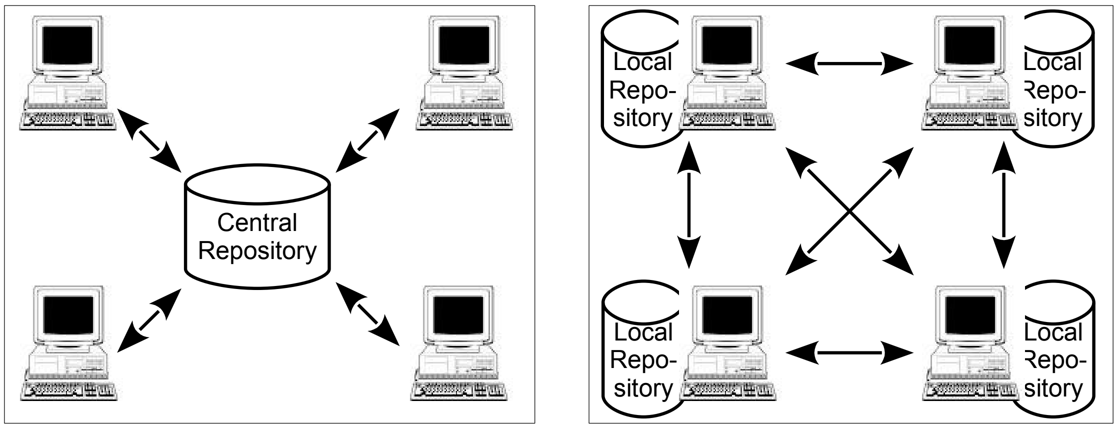


Figure 3.12.: Version Control Systems [119]

The complexity of version management is mainly due to the variety of issues which have to be addressed such as:

- Concurrency: The capability to access the same object by different users at the same time
- Merging: The integration of two versions, checked-in by two developers working simultaneously at the same object
- Delta comparison: The ability to retrieve the difference between two versions of the same object

3.2.4. Rights Management

The Enterprise Rights Management (ERM) is an information protection technique. It is intended to manage rights and usage of the data throughout their lifecycle within an organization and even if they are distributed among persons involved [75]. The ERM mainly aims to protect an

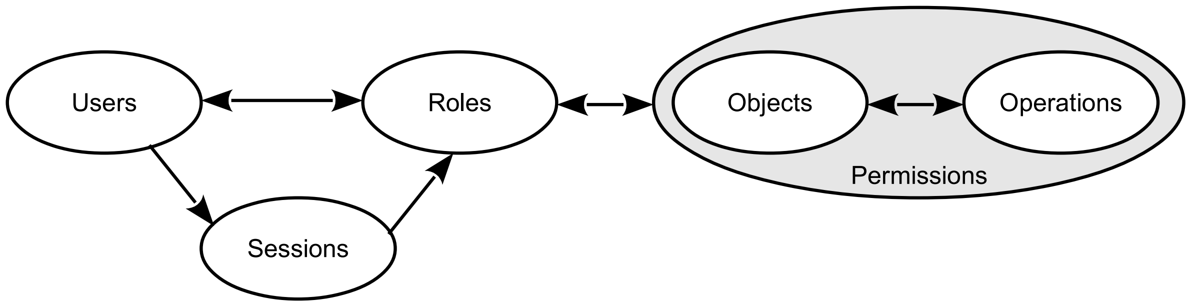


Figure 3.13.: Role-Based Access Control [90]

addressed resource by enforcing 'fine-grained' access and usage restrictions on the data content like, read only, read and copy, modify, etc. [151]. In order to efficiently achieve the ERM objectives, an appropriate access control concept is to be carefully addressed and executed. Role-Based Access Control (RBAC) represents the capability to support the rights management concept within an organization by taking into consideration the functional tasks performed by specific authorized users [145]. It is based on the fact that the roles within an enterprise are to some extent stable comparing to the users and permissions [73]. Therefore, the control of access rights regarding the roles makes the management easier and more efficient. The core model of the RBAC as it is introduced by ANSI [90] is shown in Figure 3.13. The main items are:

- Users: persons, agent, etc.
- Role: the function performed by the authorized user
- Permission: the approval of performing an operation or to access an object
- Object: can be any system element covered by the access control like file, resource, etc.
- Operation: the execution of the function
- Session: represents the ability to assign a user to many roles and vice versa

"Objects" and "Operations" are dependent on the addressed organization. They represent the two different environment aspects of the enterprise model as it was shown in Figure 3.1. RBAC can be implemented in a form of Mandatory Access Control (MAC), which is supported by DBMSs [142]. The management of the security policy by MAC is centralized. It is imposed by the manager (Administrator). The users are restricted to perform the functionalities assigned to them. Within this research, the rights management is considered in two points of view in which it is endeavored to guarantee the MAC as well as the security of data throughout its life cycle.

3.2.5. Activity (Work) Management

The product, which is anticipated from a designed enterprise model can be termed as a project. It is a temporary result with an identified beginning and end. It is gained as an integrated result of all operations taking place through an enterprise. Therefore, it can be stated, that the more the integrity and coherency of the designed enterprise model, the more the efficiency of its operation-results. This leads to the needs to apply all techniques and skills to organize the activities (works) within an enterprise to exploit all available resources and increase the achievement as well as the efficiency of the results [137]. Moreover the tracking of the tasks performed at any time in order to detect bottlenecks is required. Work Activity Management (WAM) offers a means to accomplish the objectives by organizing the associated tasks and providing the resources required by the addressed project [9]. Furthermore, it provides a technique to pursue the progress of the activities and the performance of the processes within an organization irrespective of the number of the employees and the complexity of their tasks (works). Many benefits can be exploited by applying a WAM concept such as:

- it offers the ability to record everything performed within the enterprise respecting the fact that each operation executed can be associated with information useful at any time throughout the product lifecycle,
- it supports the communication between different participants,
- it helps supervisors to observe the daily performance of the employees,
- and it facilitates the tracking of the achievement degree of the project.

In order to efficiently accomplish the desired objectives within this dissertation, an activity management concept has been designed and implemented. It ensures the utilization of all applied management functions, thereby, the activity management model plays a central role in the enterprise management as it will be shown later.

3.2.6. Collaboration Management

Collaboration management is the term under which the possible interactions of an enterprise can be addressed. In this context, it can be used to reflect the process-interaction aspect of an enterprise model. The collaboration management discipline becomes more and more essential especially in the case of large distributed organizations such as shipyards and automotive companies. Its importance comes from the needs to coordinate and organize the contributions of all participating members in an efficient, effective and dynamic manner [135]. These members, who can be geographically dispersed, are responsible to achieve the activities/tasks associated with the addressed enterprise and consequentially its temporary product "project". Thus, it can be stated that the challenge by the collaboration management is to provide all persons at the right time with the right data. Many forms of managing the collaboration relationships exist, which reflect the level of the tasks-interdependencies within an enterprise. According to Fang Chen et al. [50], three collaborative levels can be distinguished (see Figure 3.14):

- Collected work: no interactions between the enterprise-works (tasks) occur, the activities are completely separated from each other. The final product is simply the aggregation of the individual-efforts.
- Coordinated collaborative work: some tasks are dependent on each other, which impose the cooperation between the individual-efforts to accomplish the desired target.
- Concerted collaborative work: in this case, each task within the enterprise is affected by other tasks. All members have to contribute in a team effort while the performance of each individual influences all other members.

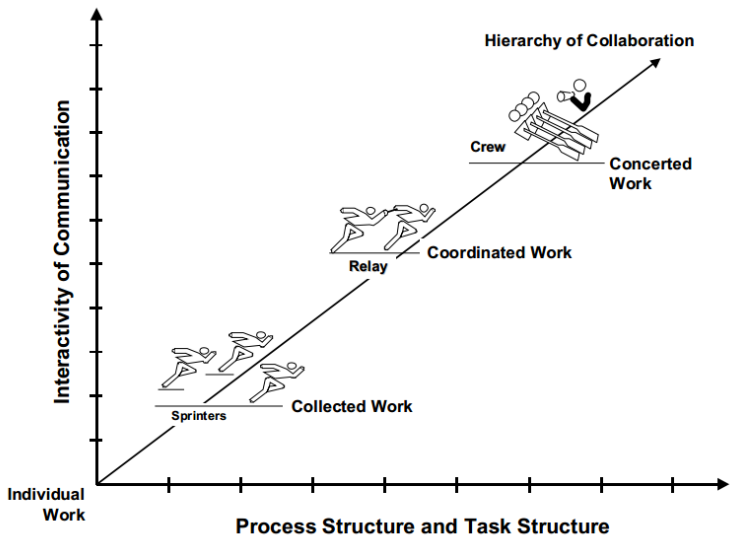

Hierarchy of Collaboration
Crew
Concerted Work
Relay
Coordinated Work
Sprinters
Collected Work
Individual Work
Interactivity of Communication
Process Structure and Task Structure

Figure 3.14.: Hierarchy of Collaboration according to Fang Chen et al. [50]

By taking into consideration the above mentioned descriptions, it can be stated, that the coordinated collaborative level is the most expressive one regarding the ship early design tasks, despite the fact, that some ship early design tasks affect (and are affected by) almost all other design tasks. An efficient management of the collaboration work within the early ship design process is one of the main goals of the developed approach within this thesis.

4. Ship Information Model

Depending on the fact that an efficient organizing of the information related to the early ship design phase can play a significant role in achieving the adopted research objectives (see section 2.5.2) and supporting the properties and capabilities characterized to the early design process (see section 2.5.1), an information model of the ship considering the early design phase is developed. Regarding their aspects and functionalities, the information is divided into three categories as it is shown in Figure 4.1: ship product-, management- and support-information. The information,

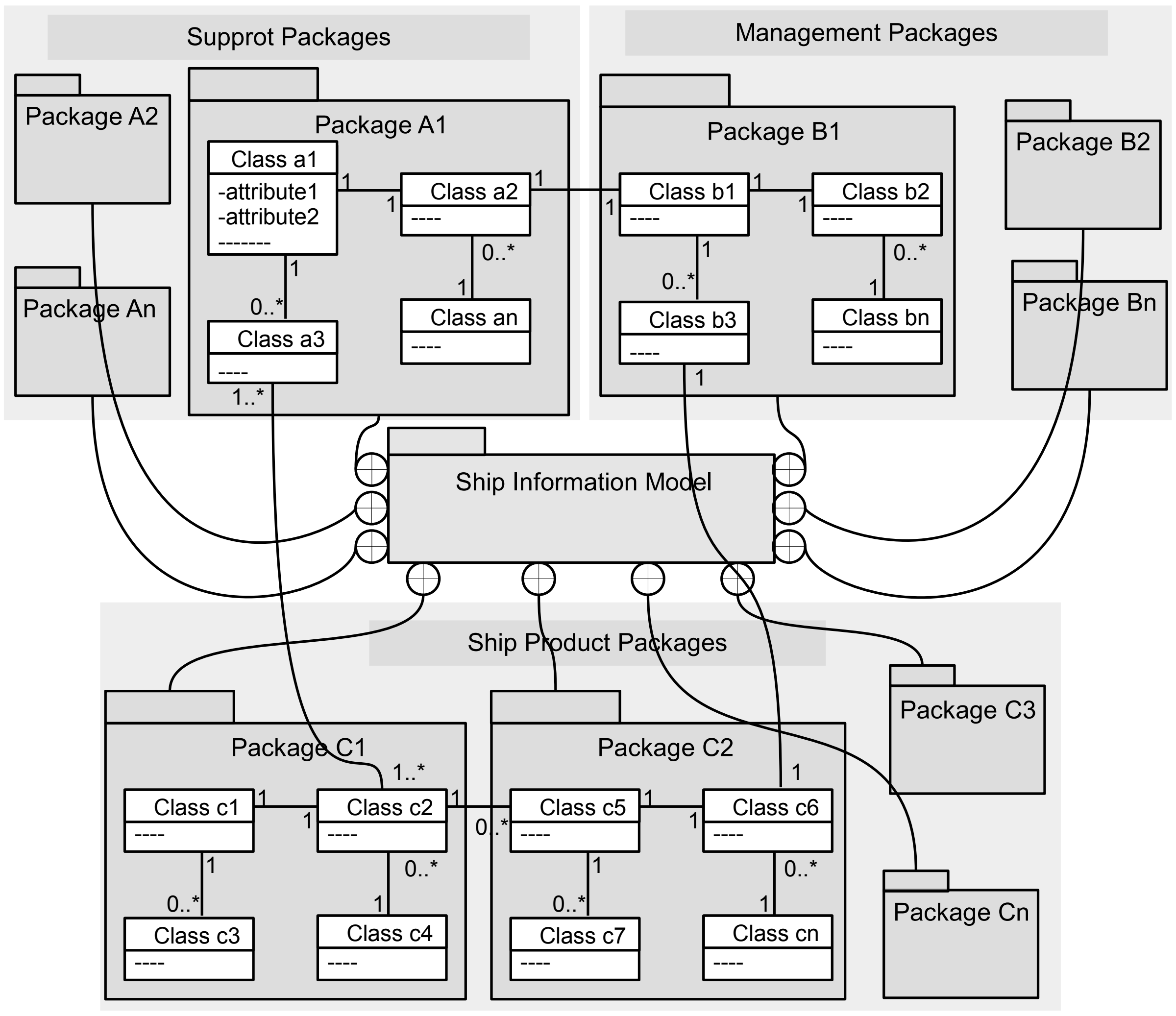


Figure 4.1.: Modeling Approach for the Ship Information Model

which are treated as objects (regardless their category) are structured, organized and related to each other reflecting the data interaction concept mentioned above. Respecting the satisfactory features and flexibilities provided by the UML method, two kinds of its diagrams (class- and package- diagrams) are applied in order to model the addressed information (More information about these diagrams are shown in Figure A.1). The UML design tool 'Visual Paradigm' [21] is utilized to model the ship information model. The ship information model is developed with the main objective of providing a neutral means (independent of any system), on which it can be based latter to configure an integrated, shareable and accessible database. This integrated database will in turn interconnected to the design applications in order to achieve the studies related to the early design phase. Furthermore, the existence of a ship information model and accordingly an integrated database will reinforce many important aspects such as the collaborative work and data exchange between different design partners. The applied modeling approach in addition to the represented information objects are described in the following sections.

4.1. Modeling Approach

The applied modeling approach is shown in Figure 4.1. The information model contains several packages reflecting the above mentioned three information categories. These packages represent the headings, under which all addressed information objects are organized. In order to clarify the modeling approach several considerations, which are adopted in the modeling, are illustrated in the following by taking into account the diagrams depicted in Figure 4.1:

- The information objects are represented as classes. Each class has a unique name and an unspecified number of attributes which reflect its properties. For example the class with the name *class a1* has the attributes: attribute1, attribute2, etc.
- The classes which serve the same aspect are assembled in a package which in this context represents a collection of information objects with a unique name. An example are the classes: *class-a1,-a2,-a3,-an* which form a package with the name *Package A1.*
- The organizing of the information objects in packages does not mean that these objects are isolated from the other package-information objects. It only means that this collection of objects represents an overriding functionality and can be connected to an unspecified number of other objects regardless their packages. In order to reflect this fact, two levels of associations are modelled, the former is represented at the information objects-level and the latter at the package-level.
- The information objects are connected to other information objects using different kinds of cardinalities offered by the UML-Class diagram: one to one, one to many, many to many. Self association, dependency in addition to inheritance (generalization), composition and aggregation are also supported. The associated objects can be included in one package or different packages. An example is *Package C1* which can be considered as *Ship Product Package* contains the *Class c2* which has these associations and cardinalities:
  - within *Package C1*: one to one with the *class c1* and many to one with the *Class c4*,
  - with information objects related to other packages: one to many with the *Class c5* which is contained within the *Package C2*, many to many with the *Class a3* from the *Package A1.*
- Many types of relationships can be established at the package-level. The packages can be related to other packages in many forms provided by the UML-Package diagram like: dependency (import, merge, etc.), inheritance, containment, etc. An example of packages-associations is the containment-relationship between the *Ship Information Model*-package and all other packages such as *Package A1*, *Package B2*, *Package C1*, etc. as it is shown in Figure 4.1.

Considering the complexity and the variety of the information associated with designing a ship, the modeling of the information objects is achieved with several objectives in mind:

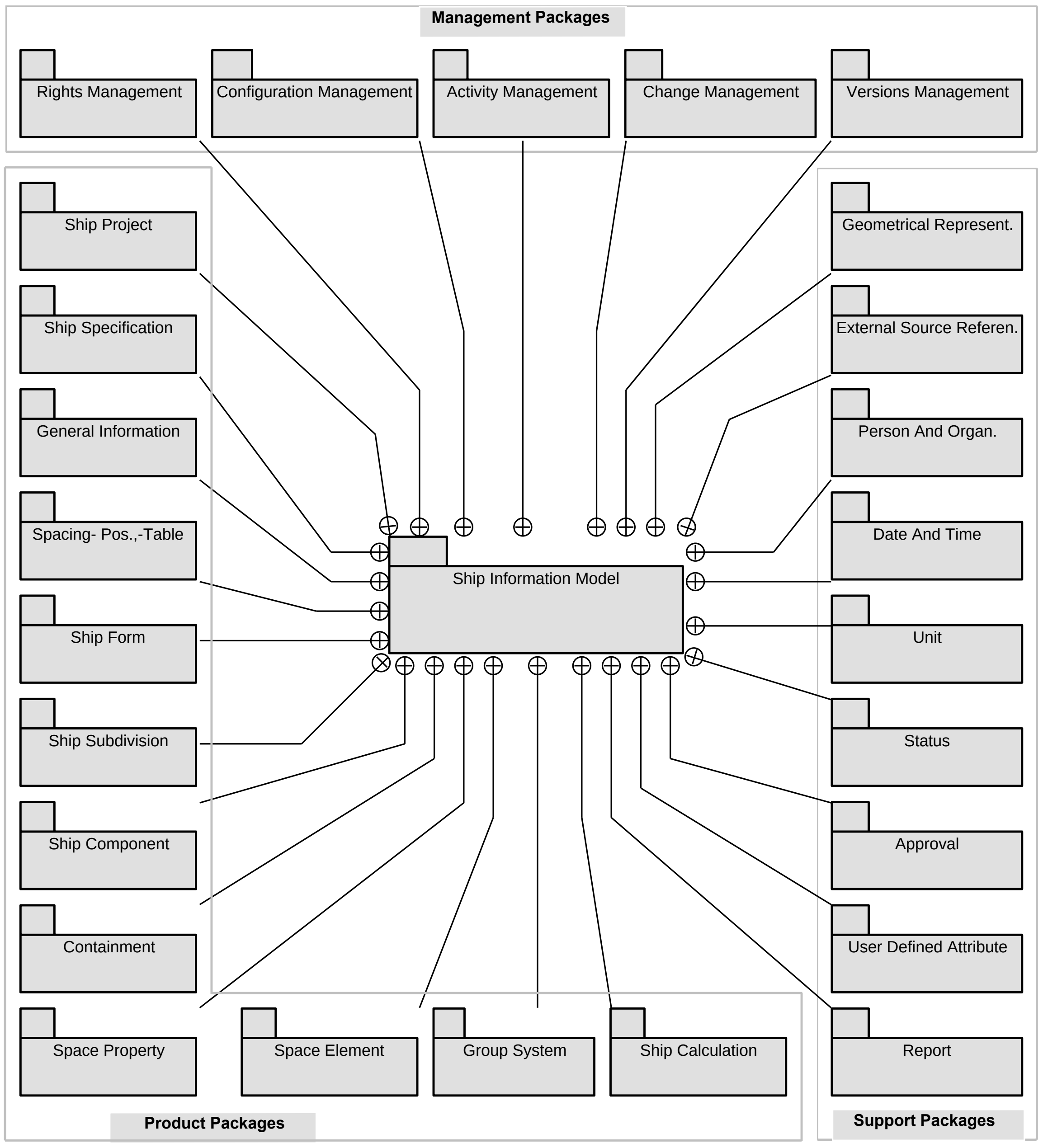


Figure 4.2.: Ship Information Model, Package Level

- The gathering of the ship product information objects which relate to each other regarding a specific aspect such as a function in the same unit (represented by package) simplifies the complexity. An example is the gathering of the information objects which represent the general information of a ship in the same package.
- The modeling of the information objects is achieved in a way that ensures the information consistency which is one of the most important aspects to be considered in order to improve the early design phase.
- Furthermore, by taking into consideration the potential needs to expand the addressed (modelled) information objects like the needs to extend the modelled information to cover more and initially not considered details (related to early or downstream design stages), the grouping aspect helps to add more packages or to accelerate the accessibility to the existed required information and accordingly to modify them. Thus, it can be stated that

in spite of the focus of this dissertation on the early design stage, the expandability of the modelled information to support other design stages, is ensured.
- Improving the management capabilities at the early design stage, regarding their importance as it is discussed in section 2.5.1, by providing several management models. These management models such as change, configuration, etc. are modelled in a way that supports the dynamic nature of the early ship design process.
- In addition to the ship product- and management-packages, the needs of support-aspects such as status, date and time, etc. (as it is previously discussed (see section 2.5.1)) is reflected within the modelled ship information model by offering these needed aspects in different support-packages.
- The grouping of the information objects in packages also has benefits regarding the implementation aspect. It facilitates the translating of the product model into the information processing system level as it will be discussed in chapter 7.

The previously described modeling approach is adopted to design the ship information model within this dissertation. In the following sections, the ship information model packages are described in detail.

4.2. Ship Information Model Package

This package represents the central package of the ship early design information model. It links to all other packages via the 'containment'-relationship at the package-level as it is shown in Figure 4.2.

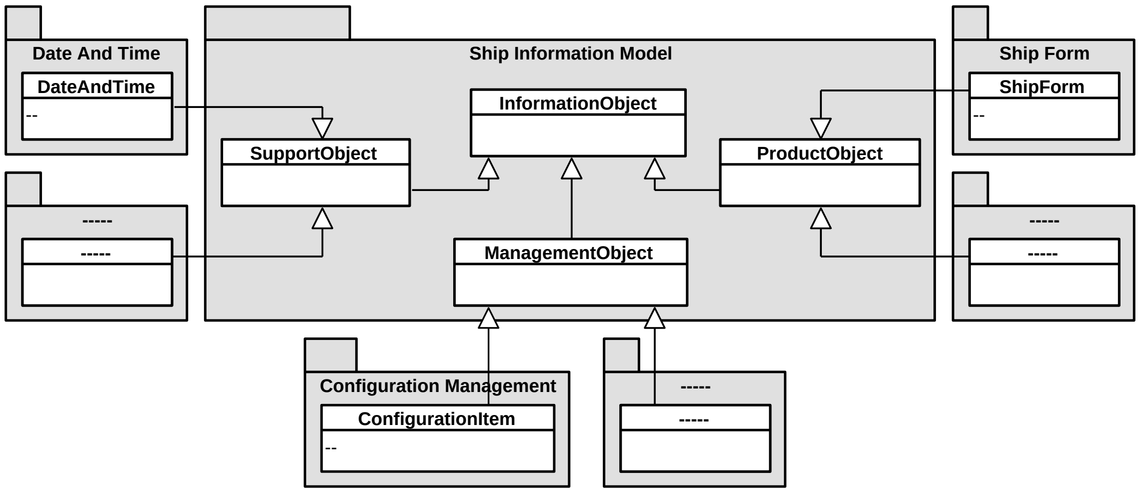


Figure 4.3.: Ship Information Model Package

This reflects the fact, that all modelled packages of the ship information model are not isolated from each other but rather they are all contained within a central one. As it is depicted in Figure 4.3, an *InformationObject* can be a *ProductObject*, *SupportObject* or *ManagementObject*. These information objects are associated with the information objects contained within other packages like *Configuration Management*-package in 'inheritance'-relationship at the information object-level as it is shown in Figure 4.3. Depending on this 'inheritance'-relationship the previously mentioned categories (either product, support or management) of the information objects are identified.

4.3. Ship Product Packages

The information objects relating to the ship as a product are organized in several packages. Each of these packages contains one or more information objects. The packages are depicted in Figure 4.2 at the left and bottom of the diagram. They offer a detailed description of the ship based on the international standards mentioned in the section 2.4.1 and by taking into consideration the requirements pointed out in literature. The information related to the ship early design are described in the following:

4.3.1. Ship Project

The model for the 'Ship Project' is depicted in Figure 4.4. Therein the information object *DesignProject*, which is related to this package in addition to its associations are represented. The *DesignProject*-object is a *ProductObject*. It represents the initial product, which should be instantiated to start design procedures. In this context the modeling of a ship begins with the foundation of a design project to which the *PredefinedActivity*(s) (see 4.5.2) are related. These activities represent the tasks, which have to be conducted within the design project.

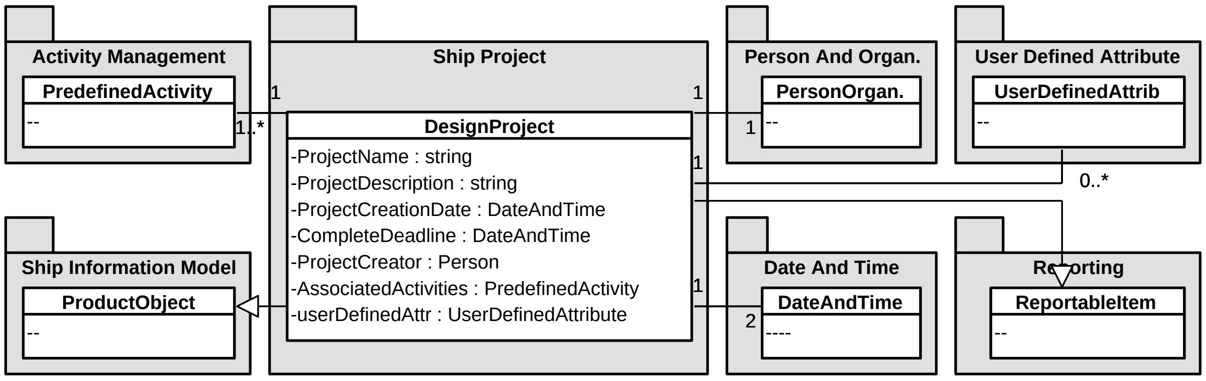


Figure 4.4.: Ship Project Information Model

By taking into account the fundamental role of the *DesignProject*-object, it is considered as the reference from which the state of the design project can be reported. As a result, the design project has to be 'reportable' in order to be covered by the reporting capabilities given in section 4.4.4. project name, project description, the person responsible for the project, the date at which the project is created and the desired completion date are properties which can be associated with the *DesignProject*-object. Moreover, the *UserDefinedAttrib* (see 4.4.7) offers the ability to associate the design project with any properties, which are considered to be important from the user's point of view.

4.3.2. Ship Specification

The owner's requirements, which are one of the most important aspects to be considered especially in the ship early design phase, are provided by *Ship Specification*-package shown in Figure A.3. The *Specification*-information object is the central object within this package. It is characterized by general properties in order to offer the ability to identify, when a specification is issued and who issued it in addition to user defined attribute. The properties can be utilized by all defined information objects within this package. The defined objects cover the following specification areas:

- Speed: as a range as well as a single value
- Endurance: in nautical miles as well as in days
- Cargo: type and capacity
- Component: to enable the definition of component specifications, which can be issued by the owner, such as the identifying of the maker of a specific component
- Manning and cabins: number of cabins of each cabin class, such as visitor cabins, etc.
- Moreover, the expandability of the model to define new specifications is guaranteed. This is achieved by defining two objects *InfoObjectRelatedSpecification* and *SpecificationSetup*. These two objects (by means of their associations together and with other information objects) enable the designer to take into consideration and define new specifications. An example of such specifications is a setting of limits or special owner's requirements regarding the *ShipForm*.

4.3.3. General Information

The information objects, which relate to the ship as a product and are considered as general characteristics, are organized in the *General Information*-package shown in Figure 4.5. The represented information objects within this package cover the following information:

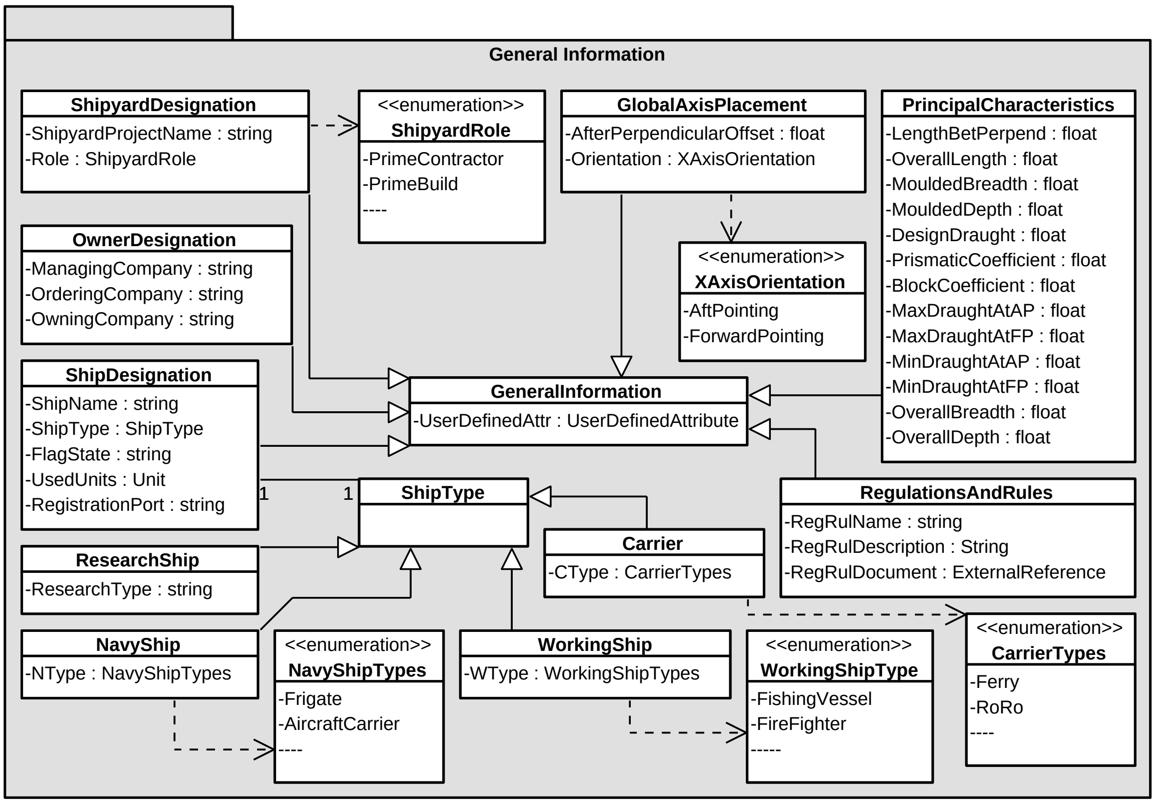


Figure 4.5.: General Information Package

- *PrincipalCharacteristics* of the ship such as block coefficient, length between perpendiculars, moulded breadth, design draft, etc., are designed as properties of this information object.

- *GlobalAxisPlacement* provides the ability to describe the global coordinate system (origin placement, orientation) which will be used during the design process.
- *ShipDesignation* provides the capability to identify a ship by offering information such as: flag state, port of registration, ship type such as navy ship; working ship, etc.
- *OwnerDesignation* respecting the owner's data like the ordering company, managing company, etc. are represented in this object.
- *ShipyardDesignation* offers a means to add information to identify the shipyard(s) like: shipyard name, shipyard role, etc.
- *RegulationsAndRules* considering the utmost importance of the commitment with the regulations and rules issued by classification and statutory in any design study, the *RegulationsAndRules*- object is defined. It offers the ability to describe these regulations in an efficient manner by means of its relationship with the *ExternalReference*-object 4.4.3.

4.3.4. Spacing- Position,-Table

This package, which is shown in Figure A.2, offers the ability to define a position in space and accordingly the spacing tables of the ship. The defined spacing positions can be used as reference positions to locate an object whether it is structural like a subdivision element or a component. The *SpacingPosition*-object, which can be used with/without an offset value as a defined distance to a specific position, is characterized by properties like name, number, etc. in addition to the possibility to assign the type of the spacing position. It can be a vertical-, longitudinal- or transversal-position. The *TableDescription*-object provides the capability to define frame-, station-tables in addition to the possibility to define tables in transverse and vertical directions. These tables are defined as a set of predefined *SpacingPositions* with an identified distance between them.

4.3.5. Ship Form

The surfaces, which form the shape of a ship are gathered in the package shown in Figure A.4. The information objects included within this package are modelled in a way that reflects the fact, that the *ShipForm* is formed by *HullForm* and *Superstructure*. Thereby, the ability to use these two objects as standalone objects as well as a part of *ShipForm* is guaranteed. The representation capabilities of these two objects, which can be in 2D or 3D format, are gained from the association with the *Representation*-object 4.4.1. Moreover, it is possible to associate the hull form with properties which are modelled in the information object *HullParticulars*. This enables the designer to add details about the hull form such as midship-, bulbous bow-properties, etc. by exploiting the capabilities offered by this object and its relations to the information objects *SAC*, *MidshipTumble*, *BulbParameters*, *BottomParameters* and *DeckParameters*.

4.3.6. Ship Subdivision

The elements, which are used to divide a ship into spaces are defined within this package. It is shown in Figure A.5. The term Subdivision Element (SE) is utilized to refer to such elements. The main information object within this package is the *SubdivisionElement*. This object is characterized by the following properties: name, representation, shape control, borders, surface area, tightness, function, location and user defined attribute. Therefore, some relationships are originated between *SubdivisionElement* and information objects related to other packages. These associations are with: *Placement*, *Representation*, *SurfaceArea*, *Tightness* and *UserDefinedAt-*

tribute. Moreover, some information objects are modelled within this package and associated with the *SubdivisionElement*:

- *SeFunction*: assignment of predefined functions to a SE such as: bulkhead, deck, floor, etc.
- *ShapeControl* as well as *Camber*: The SE-shape is controlled by utilizing a sheer and camber concept. The sheer concept, for instance, specifies the longitudinal form of a deck or a longitudinal bulkhead and the transversal form of a transverse bulkhead. The camber concept specifies for example the transversal form of a deck or longitudinal bulkhead and the longitudinal curvature of a transverse bulkhead (see Figure 7.9).
- *Border* as well as *BorderType*: The extension of a SE can be defined by the identification of the borders which represent the limits of a SE.

4.3.7. Containment

The contents, which can be assigned to the spaces reflecting the purpose for which they are designed, are assembled in the *Containment*-package shown in Figure A.6. This offers the possibility to take into account the special requirements of some contents during the design of the related spaces early in the design process. These preconditions like the carriage pressure required to be applied in a tank in order to carry, for instance methanol, can be combined with this liquid type as it is assigned to a tank. In the following are the information objects, which are included within this package and describe the contents regardless whether they are cargoes or supplies:

- *LiquidContent*: many liquid types are predefined and can be selected by the designer to be allocated to tanks like water ballast, heavy fuel oil, etc.
- *GaseousContent*: chlorine, acetaldehyde, ethyl chlorine, etc., are examples of gaseous content types which are predefined and can be used by the designer.
- *DryContent*: two types of dry contents are defined: *Unit* such as container, vehicle, etc. and *Bulk* like grain, etc.
- *Person*: in order to enable designers to assign persons whereas they are crew or passengers to the spaces, the *Person*-object is modelled. It offers the ability to deal with the persons from the space-occupation point of view.

4.3.8. Space Property

Properties, which can be associated with spaces regarding their geometry, contents, etc., are collected together in a same package named as *Property*. This package is shown in Figure A.7. The information objects modelled within this package, are:

- *WCOG*: weight and center of gravity properties
- *Volume*: total volume, net volume, center of volume and permeability
- *SurfaceArea*: the surface area by means of its value and type such as stiffened surface area, unstiffened surface area, etc.
- *Tightness*: like air tight, water tight, etc.
- *TankProperty*: the compartment spaces, which are designed to be filled with liquids can be characterized by some details respecting their specificity such as the information that describe the piping system of a tank
- *CargoSpace*: to associate the load spaces with properties by taking into consideration the cargo
- *Coating*: The *Coating*-object and its relations with *CoatingTypes*-, *CoatingLevel*-objects enable the designer to describe the required protective coating for the designed space

- *Insulation*: with insulation categories A, B, etc. defined in STEP-ISO 10303, AP-215 [113].
- *Noise*: with noise categories A, B, etc. defined in STEP-ISO 10303, AP-215 [113].
- In addition to the objects: *Illumination*, *Occupancy*

4.3.9. Space Element

Different types of spaces into which a ship is subdivided during the design process are represented within this package shown in Figure 4.6.

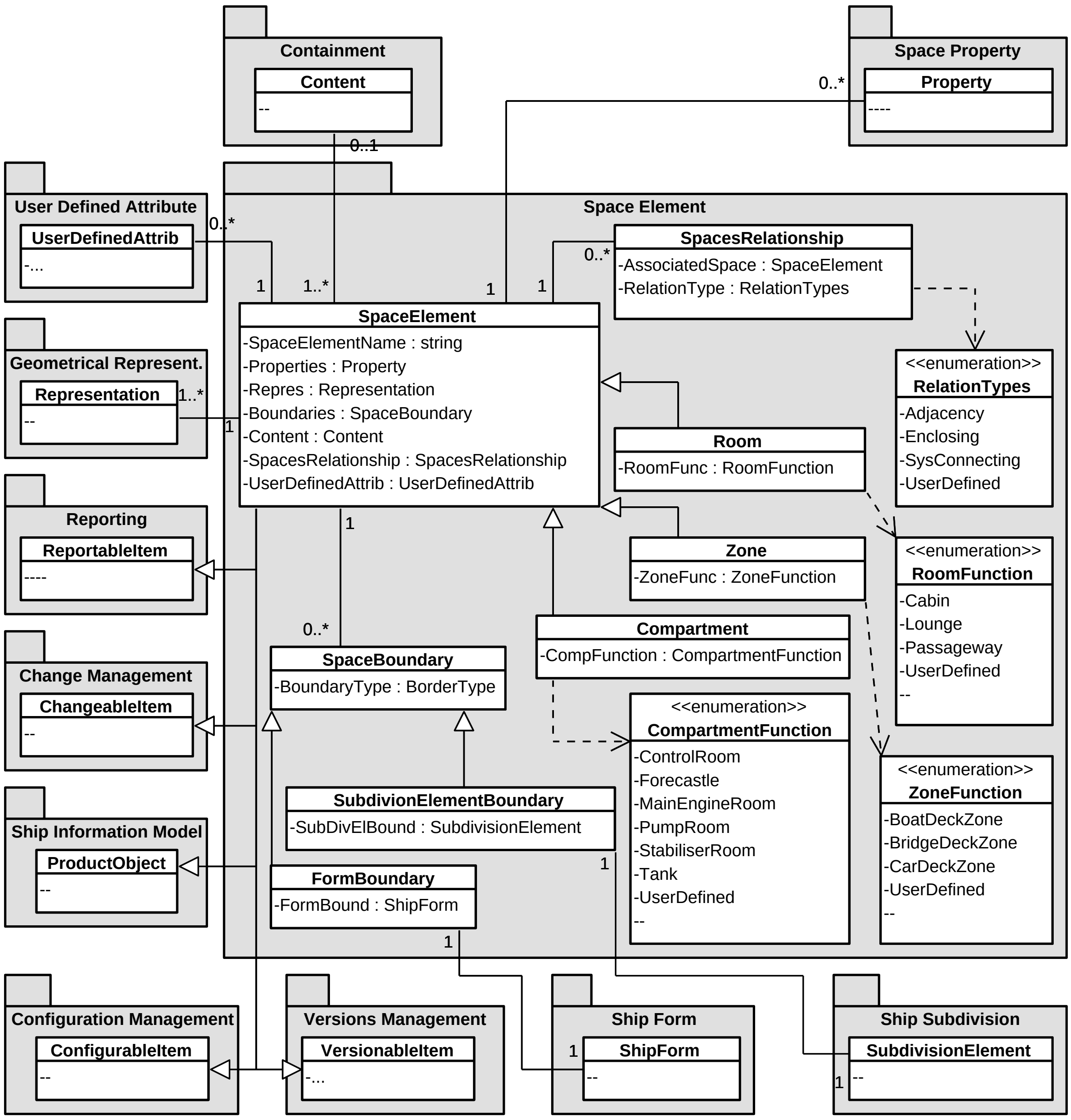


Figure 4.6.: *Space Element*-Package

It offers three types of space elements: compartment, room and zone. These spaces share many properties in addition to their own attributes. Each of them is joined with many enumerated functions enabling the designers to use predefined as well as newly defined functions. For example a compartment can be a Pump Room, Tank, etc. while a Room can be a Cabin, Mess Room, etc. The information objects of this package offer the ability to reflect the overlapping nature of the spaces. For instance, a zone can contain one or more compartments and vice versa. Considering the central role of the space elements in the ship early design process as it has been mentioned previously (see section 2.5.1), many associations between the information objects included within *Space Element*-package and different ones are modelled. These relationships depicted in Figure 4.6 are an example of the associations, which can be established within the ship information model. It shows that within this package, which contains the information objects: *SpaceElement*, *Compartment*, *Zone*, *Room*, *CompartmentFunction*, *RoomFunction*, *ZoneFunction*, *SpaceBoundary*, *FormBoundary*, *SubdivisionElementBoundary* and *SpacesRelationship*, the following associations are modelled:

- *Compartment*-, *Room*-, and *Zone*- objects are associated with the *SpaceElement*-object in a form of inheritance-relationship, which means that each of them is considered, in general, as a space element and can represents the attributes offered by the *SpaceElement*-object in addition to its own attributes.
- The function attribute of the *Compartment*-object is an example of an own attribute. The type of this attribute as *CompartmentFunction* refers to the way, in which the *Compartment*-object gets the values (instances) of this attribute. This is represented by the dependency-relationship with the *CompartmentFunction*-Object, which has a type «enumeration», the value can be assigned to the function-attribute is constrained to the predefined functions. Moreover, the designer can assign the value 'userDefined' to the attribute and then exploit 'UserDefinedAttrib'-attribute, which is gained from the *SpaceElement*-object to define a new function.
- One to many relationship between the *SpaceElement*- and *SpacesRelationship*- objects refers to the previously mentioned overlapping nature of the spaces. Spaces can be associated to each other in many relation-types such as adjacency, enclosing, etc.
- One to many relationship between the *SpaceElement*- and *SpaceBoundary*- objects reflects the fact that each space element is defined by one or more boundaries, which can be defined as subdivision element- or ship form- boundary (see 4.3.5 and 4.3.6).
- The inheritance relationship between *SubdivisionElementBoundary*, *FormBoundary* on the one hand and the *SpaceBoundary* on the other hand means, that the both objects can be considered as boundaries for the spaces.

Moreover, many associations with objects, which are defined in other packages, are depicted, such as:

- One to many association between the objects *SpaceElement* and *Property* 4.3.8. This offers the ability to associate each space element with zero or more properties.
- Many to zero/one relationship between the objects *SpaceElement* and *Content* 4.3.7: one or more spaces can contain zero/one containment
- One to many connection between the objects *SpaceElement* and *Representation* 4.4.1. reflecting the fact that each space can be represented in any number of ways.
- Zero to many association between the objects *SpaceElement* and *UserDefinedAttribute* 4.4.7: each space can be associated with zero or more attributes defined by the user.
- The *SpaceElement*-object inherits from many objects in order to be covered by the aspects they serve (to make use of their properties). It inherits from: *ProductObject*, *ConfigurableItem* 4.5.3, *VersionableItem* 4.5.5, *ReportableItem* 4.4.4 and *ChangeableItem* 4.5.4.
- One to one relationship between the objects *SubdivisionElementBoundary* and *SubdivisionElement* 4.3.6 and between *FormBoundary* and *ShipForm* 4.3.5. They offer a means to define the boundaries in order to be used in the definition of spaces.

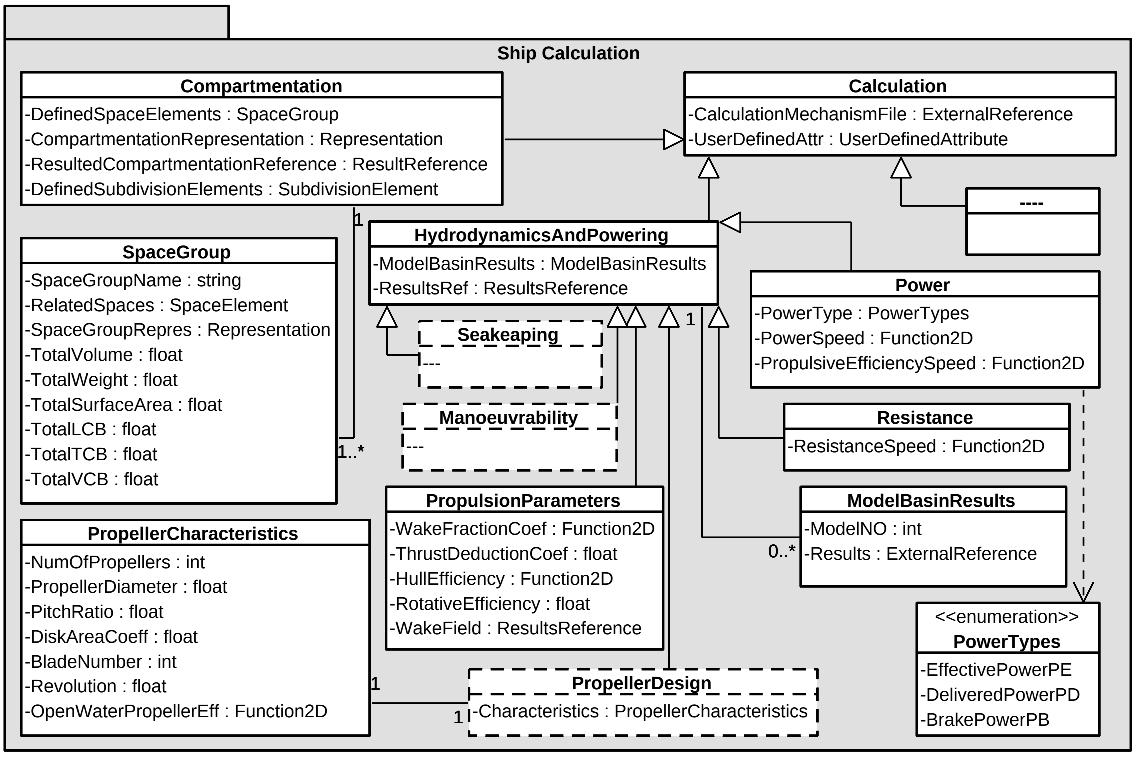
4.3.10. Ship Component and Group System

Considering the necessities and needs related to the addressing of the ship components in the early design process as it has been described previously (see section 2.5.1), a ship component package is modelled. It represents a general model to define ship components whether they are machinery, outfitting, etc. The two packages *Ship Component* and *Group System* are shown in Figure A.8. In order to enrich the ship components with the needed properties at the early design stage, three kinds of attributes are characterized to the *ShipComponent*-object:

- Properties which affect the decisions made in the early design stage, for example: weight properties, needed volume, component location, etc.
- More details about the component can be assigned, such as: component name, maker, type, in addition to the possibility to handle the component representations and catalogs offered by the maker if available.
- Identifications assigned to the components regarding a specific group system, such as SFI (see 2.4.2). In order to meet the needs of shipyards, components suppliers, etc. which use group systems to standardize their operations, a group system package, which offers a model to define group systems, is modelled. In order to enable the designer to define a group system with its details such as documents, a relationship between the information objects *GroupSystem* and *ExternalReference* is originated.

4.3.11. Ship Calculation

The subject of the information objects modelled within this package is the management of results from naval architectural calculations such as power prediction, intact and damage stability calculations, etc., which reflect the process of the ship design.


Ship Calculation
Compartmentation
-DefinedSpaceElements : SpaceGroup
-CompartmentationRepresentation : Representation
-ResultedCompartmentationReference : ResultReference
-DefinedSubdivisionElements : SubdivisionElement
Calculation
-CalculationMechanismFile : ExternalReference
-UserDefinedAttr : UserDefinedAttribute

1
HydrodynamicsAndPowering
-ModelBasinResults : ModelBasinResults
-ResultsRef : ResultsReference
SpaceGroup
-SpaceGroupName : string
-RelatedSpaces : SpaceElement
-SpaceGroupRepres : Representation
-TotalVolume : float
-TotalWeight : float
-TotalSurfaceArea : float
-TotalLCB : float
-TotalTCB : float
-TotalVCB : float
1..*
Seakeaping

Manoeuvrability

1
Power
-PowerType : PowerTypes
-PowerSpeed : Function2D
-PropulsiveEfficiencySpeed : Function2D
Resistance
-ResistanceSpeed : Function2D
PropulsionParameters
-WakeFractionCoef : Function2D
-ThrustDeductionCoef : float
-HullEfficiency : Function2D
-RotativeEfficiency : float
-WakeField : ResultsReference
ModelBasinResults
-ModelNO : int
-Results : ExternalReference
0..*
PropellerCharacteristics
-NumOfPropellers : int
-PropellerDiameter : float
-PitchRatio : float
-DiskAreaCoeff : float
-BladeNumber : int
-Revolution : float
-OpenWaterPropellerEff : Function2D
1
1
PropellerDesign
-Characteristics : PropellerCharacteristics
<<enumeration>>
PowerTypes
-EffectivePowerPE
-DeliveredPowerPD
-BrakePowerPB

Figure 4.7.: Part of Ship Calculation Model, see also A.9

This package can be considered as a bridge between the two perspectives of the ship design, which are: the 'process' represented by the ship calculations themselves and the 'data' represented by the results of these calculations, which have their foundations within this package as information objects. The objects of this package are shown in Figure 4.7 as well as in Figure A.9. The *Calculation*-object plays a central role within this package. In addition to the neutral nature of the developed information model and accordingly of the information objects included within this package for explicitly saving the key results of each calculation type such as *Compartmentation*, *HydrodynamicsAndPowering*, etc., the association to the results as stored in the applied tools (which are used to perform the calculations related to the ship early design process), is managed. The *Calculation*-object is designed in a way that enables the designer (when it is required) to define an indicator to the calculation result-stores in the applied tools. Each of the following calculation objects, which represent the results of the ship early design tasks (see section 5.2), will be described in detail in chapter 5: *Compartmentation*, *Hydrostatics*, *IntactStability*, *DamageStability*, *Capacities*, *Weights*, *LoadingCondition*, *HydrodynamicsAndPowering*, *Freeboard* and *LongitudinalStrength*.

4.4. Support Packages

By taking into consideration the complexity nature of the early ship design process and the needs of support aspects to improve some functionalities during the design process (as it has been mentioned previously, see section 2.5.1), support models are designed offering these capabilities. They are modelled based on international standards such as ISO 10303-41 [6], ISO 10303-42 [4] and are grouped in packages described in the following sections. These packages are represented at the right side of diagram shown in Figure 4.2.

4.4.1. Geometrical Representation

The purpose of this package shown in Figure A.10 is to support the geometrical representation aspect within the developed ship information model. By making use of the information objects included within this package, the following aspects can be defined:

- Placement: The ship design elements, which can be structural like SE, component, etc., can be placed in the space by means of the information objects defined within this package. Two types of placements can be defined. The former is accomplished by defining a placement coordinate system by means of the information object *Axis2Placement3D*. The latter is achieved by means of the information object *PlacementWithPositionReference*. This object enables the designer to identify the reference to which an object is positioned. The reference position can be: any defined subdivision element such as a deck (see 4.3.6), as well as it can be any defined spacing position such as a frame position (see 4.3.4).
- Transformation: Translation, rotation, mirroring and uniform scaling of design elements can be defined by means of the information object *Transformation3D*.
- Representation: The different representations (2D and 3D) of the design elements, such as the hull form, subdivision element, etc., can be treated within this package by means of the *Representation*-object. This object offers the ability to handle different formats (such as IGES, STL, dxf, etc.) as they are built or formulated.
- *BoundingBox*, *Point3D* and *Moment3D* are stand-alone attributed information-objects and can be used by any object included within the information model. Moreover, the *Function2D*-object offers the ability to handle the dependent data of design diagrams and tables such as the values of ship resistance at different ship speeds.

4.4.2. Person And Organization

This package, which is shown in Figure A.11, is modelled in order to take into consideration the variety of persons, who take place in the design process. Many properties are characterized to the *PersonOrganization*-object such as: last name, first name, organization, address, password, etc. Respecting the shipyard structure shown in the figure 4.8, which raises the role-aspect of the persons related to a shipyard, many predefined roles such as shipyard manager, head of department, project manager, etc., are modelled within the enumerated *Roles*-object and can be assigned to a person. Moreover, by taking into account the teamwork nature of the design process, a *PersonalInbox*-object is modelled. It offers a means to support the collaborative and communication concept between different design members.

4.4.3. External Source References

In the design, many types of sources are utilized. In order to enhance the ship information model by providing the ability to reference to the external sources, the objects *Tool* and *DocumentReference* which can be *ResultReference* or *ExternalReference* are modelled within a package termed as *External Source References* (see Figure A.12). By means of these objects, the designers can handle the following sources:

- Documents which can be viewed and edited, for example an IGES-file which representing a hull-form-definition. The data of these documents have the foundation within this package to be treated as it is represented (as ‘blob’-data).
- Documents which can be viewed but not edited. Examples for this are pdf-files, pictures (png, jpg,...), etc., which can represent, for instance, the regulations of the classifications societies. The designer is able to define an indicator to these documents.
- Tools: whether they are programmed by the designer, shipyards, etc. or are commercial softwares. These tools, which represent the mechanisms used by the design team to achieve the design studies, have their foundation within this package to be referenced.

4.4.4. Report

The purpose of this package, which is shown in Figure A.13, is to support the reporting and correspondingly the visualization aspects within the ship information model. Any object, which is able to be reported or visualized can be covered and accordingly use the properties offered by this package as soon as it is characterized as *ReportableItem*. *DesignProject* 4.3.1, *PredefinedActivity* 4.5.2 and *ConfigurationItem* 4.5.3 are examples of the information-objects, which are considered as *ReportableItem*s. The name, reported by, reported at, etc., are some of the properties characterized to the *Report*-object. In this context and by taking into consideration the data process interaction concept, it can be stated that the significant role of this package is, that it offers the foundation within the information model on which the reporting capabilities (which reflect the process perspective of this aspect, see the section 5.2.2.5) are based.

4.4.5. Status

A package for the status shown in Figure A.14, which is one of the most important concepts to be considered through the early design process, is modelled. This provides the ability to associate any needed information object with a predefined status such as preliminary, unspecified, etc. as well as by a user defined status. Adding the status dimension to the design data make these data more expressionistic and bring great benefits to the ship design as it will be seen later.

4.4.6. Approval

This package shown in Figure A.14 is modelled in order to enhance the design team with the ability to relate some objects such as the *ChangeControl*-object 4.5.4 with the approve, or reject aspects. In this context, this package exists to support the decision making concept within the design process. Furthermore, the *Approval*-object is characterized with properties to enable the identification of the date of approval and the person(s), who make the approval decision.

4.4.7. User Defined Attribute

In order to enable the design team to associate the information objects with their own considerations to represent aspects or properties which were not taken into account by the modelled objects, a package providing the ability to define new attributes by the users is modelled and termed as *User Defined Attribute* (see Figure A.14).

4.4.8. Date And Time, Unit

Considering the needs of unit and date/time aspects in the ship early design process, the following two packages shown in Figure A.14 are designed.

Date And Time: This package is modelled in order to enable the design team to take into consideration the date/time data within the design process.

Unit: In the design, different units can be used. This fact is reflected within this package, which enables the user to define different kinds of units

4.5. Management Packages

Considering the great necessity of the management concepts in the early ship design process as it has been previously outlined (see section 2.5.1), the developed ship information model is enriched by the following management capabilities: activity, configuration, change, rights and versions. In the following sections the description of these management areas as they are considered in this dissertation is given. The modeling is based on the principle concepts explained in the section 3.2 and take into consideration the particularities of the ship early design phase. The management packages are depicted in the upper part of the diagram shown in Figure 4.2. They represent the data perspective of the management models developed within this dissertation. They provide the foundation within the information model on which the process perspective of these management models is based. This process perspective will be described in detail regarding the early ship design process in the next chapter.

4.5.1. Rights

Shipyards are generally composed of several departments. Each of these departments has its own manager in addition to a number of designers. These designers are authorized to play specific roles within a new design project. An example of that is the shipyard depicted in Figure 4.8, which is compatible with the 'Matrix Projekt Organization' introduced by Beiderwieden et al. [41]. It

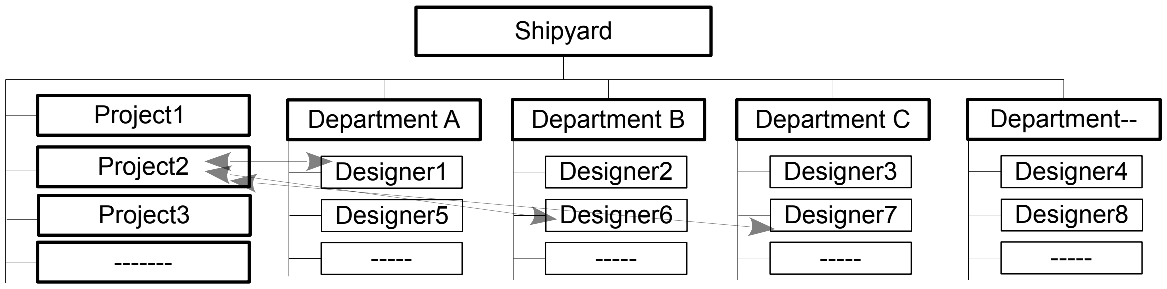


Figure 4.8.: Shipyard Structure

shows that the designer1, for instance, is related to the DepartmentA (which can be a department for ship theory) and in the same time is authorized to play a role in the Project2. This reflects the teamwork nature of the ship design process and highlights the needs to develop a means, within which the possibility to distribute the project related tasks to the responsible designers is given. Considering this fact, a model for rights management is developed as it is shown in Figure 4.9. Regarding the rights-aspects described in section 3.2.4, this model satisfies the concepts 'Role-Based Access Control' as well as 'Mandatory Access Control'.

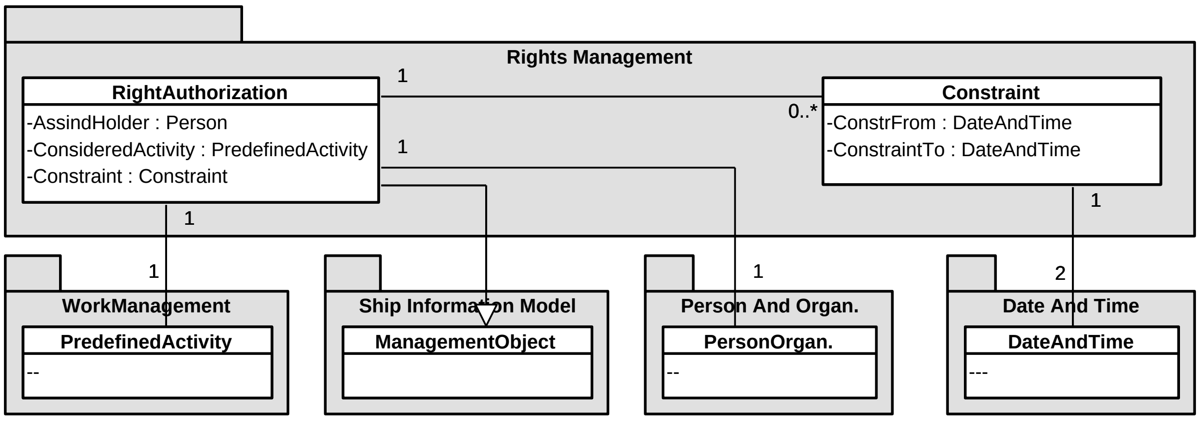


Figure 4.9.: Rights Management Model

The package of *Rights Management* includes the following two information objects:

- *RightAuthorization*: is the central object within this sub-model. The properties, which offers the possibility to distribute the responsibilities for performing specific design tasks within a design project. Furthermore, this authorization of rights can be limited by means of the *constraint*-object.
- *Constraint*: offers the possibility to restrict the rights authorized to a person. These restrictions (from-to) are defined depending on the *DateAndTime*-object.

4.5.2. Activity

This sub-model provides the information objects to handle predefined activities from the data as well as from the data-process perspectives. It is termed as *Activity Management*-package as this model is intended to represent the activities (tasks), which have to be organized in order to be performed within the early design phase. It represents a central model to which all information model objects are directly or indirectly connected. It is noteworthy to mention that this model

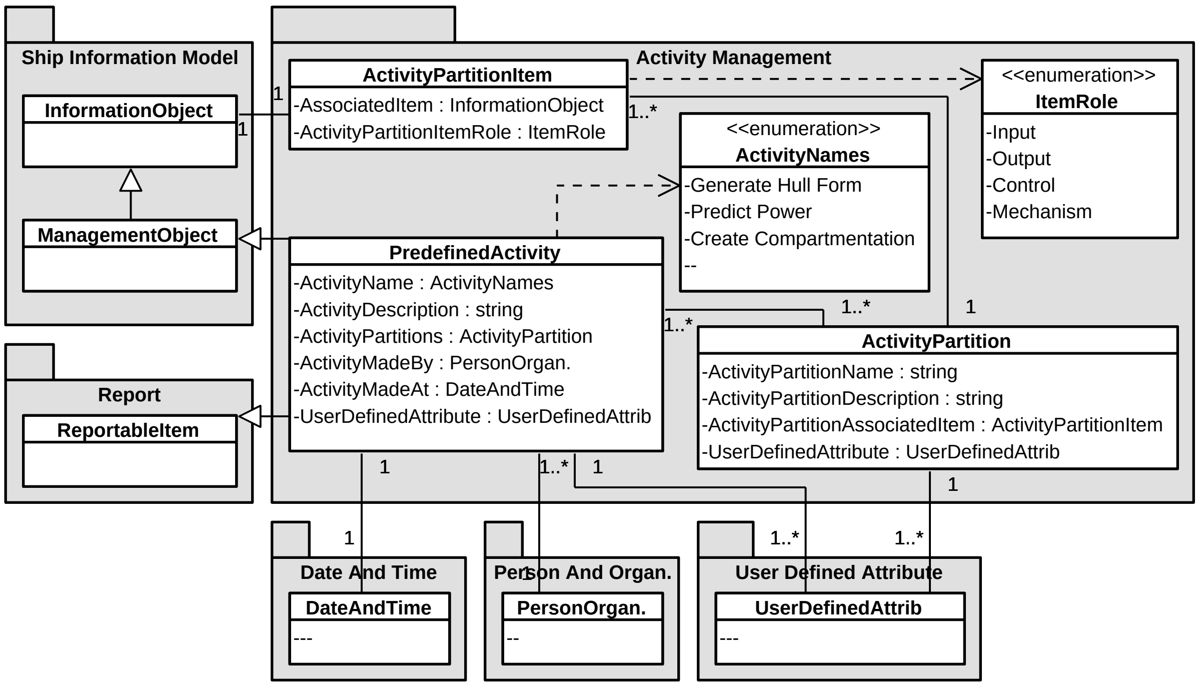

Ship Information Model
InformationObject
ManagementObject
Report
ReportableItem
Activity Management
ActivityPartitionItem
-AssociatedItem : InformationObject
-ActivityPartitionItemRole : ItemRole
<<enumeration>>
ItemRole
-Input
-Output
-Control
-Mechanism
<<enumeration>>
ActivityNames
-Generate Hull Form
-Predict Power
-Create Compartmentation
--
PredefinedActivity
-ActivityName : ActivityNames
-ActivityDescription : string
-ActivityPartitions : ActivityPartition
-ActivityMadeBy : PersonOrgan.
-ActivityMadeAt : DateAndTime
-UserDefinedAttribute : UserDefinedAttrib
ActivityPartition
-ActivityPartitionName : string
-ActivityPartitionDescription : string
-ActivityPartitionAssociatedItem : ActivityPartitionItem
-UserDefinedAttribute : UserDefinedAttrib
Date And Time
DateAndTime
Person And Organ.
PersonOrgan.
User Defined Attribute
UserDefinedAttrib
1
1..*

Figure 4.10.: Activity Management Model

is not qualified to represent the process work-flow, which is a process-modeling issue as it will be seen later in the next chapter, but rather to offer the foundation within the developed ship information model and subsequently within the database in order to bridge the gab between the data and process perspectives. It includes the following information objects as depicted in Figure 4.10:

- *PredefinedActivity*: properties, which can be characterized to the ship early design tasks as predefined activities, are modelled within this information object. It represents a *ManagementObject*. Each predefined activity can be enriched by user defined attributes. The names of these activities are gained from the association with the enumerated *ActivityNames*-object. These activity names represent the names of the tasks, which should be performed within the ship early design phase such as *Generate Hull Form*, *Predict Power*, etc. These tasks (*PredefinedActivity*)are performed at a specific date/time by a person/organization to whom the appropriate rights to perform the activity is assigned. One or more *ActivityPartition*(s) constitute a predefined activity. Considering the fact that the properties of the *PredefinedActivity*-object offer an organized details of the represented task, it is designed to be a *ReportableItem* and accordingly to be covered by reporting capabilities.
- *ActivityPartition*: early ship design tasks and accordingly the *PredefinedActivity* are represented in sets of partitions. *ActivityPartition* offers the capability to handle the information related to the partitions of a predefined activity. The many to many relationship between the objects *PredefinedActivity* and *ActivityPartition* reflects the fact, that each activity partition can be related to one or more predefined activities and each predefined activity is composed of one or more activity partitions. Each partition has a name. It can be described or even enriched by user defined attributes. Each of the *ActivityPartition*(s) is interacted with an information object termed as *ActivityPartitionItem*, which in turn is associated to many other information objects within the ship information model. As a result, the predefined activities are modelled to interact with the information objects at

the activity partition level.

- *ActivityPartitionItem*: Any of the *InformationObject*(s) of the developed information model subdivided into three categories *SupportObject*, *ProductObject* and *ManagementObject*, can be assigned to the *ActivityPartition* by means of the *ActivityPartitionItem*-object. When an *InformationObject* associates to an activity partition, it plays a specific role. Respecting the ICOM concept, an *InformationObject* can either serve for input, output, control or mechanism purposes.
  For example, when the *PredefinedActivity*-object is used to describe an early design task such as *Create Compartmentation*, one of the *ActivityPartitionItem*(s), which is used by the *ActivityPartition*-object will have a 'mechanism'-role. The system which is used to create the compartmentation can be defined as a *Tool* (see 4.4.3) and associated to the *ActivityPartitionItem* to play the 'mechanism'-role.

4.5.3. Configuration

Regarding the advantages which can be gained from the applying the configuration management aspect in the early design process mentioned in section 2.5.1, the adopted configuration management concept, as it was shown in Figure 3.10 plays a central role in the developed approach. It requires a means to combine different objects regardless their types and relate them with a specific status by taking into account a specific requirement.

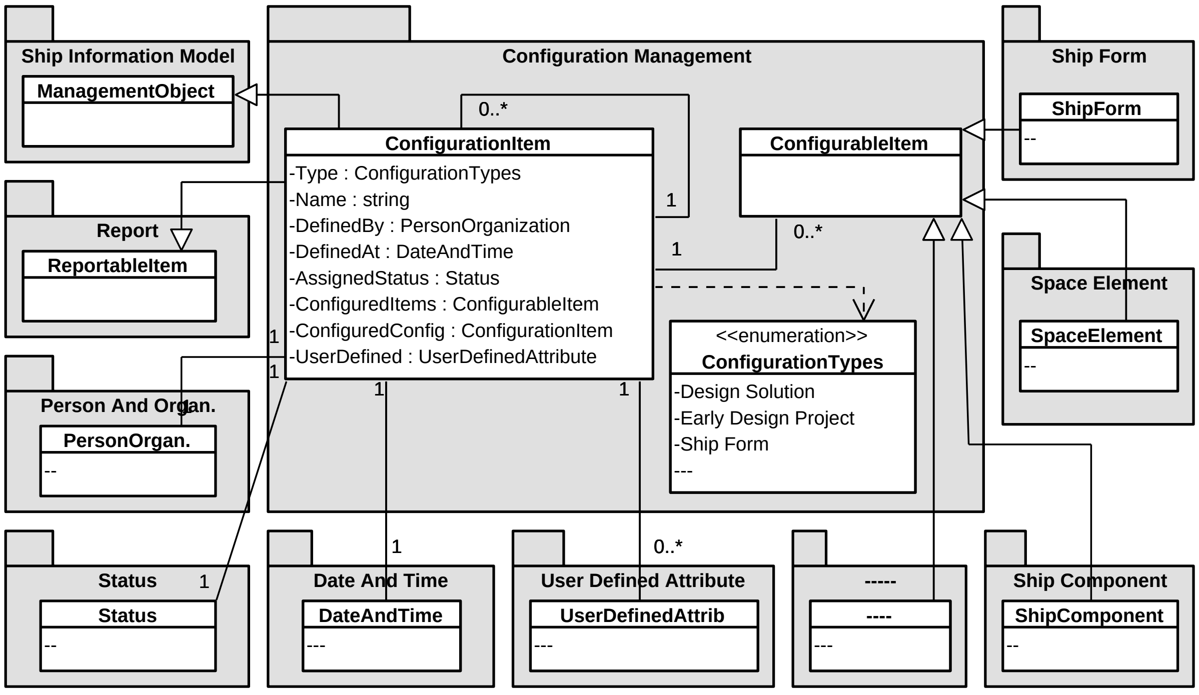


Figure 4.11.: Configuration Management Model

In order to reflect this concept and provide the capability to handle it within the developed ship information model, a package is modelled as it is shown in Figure 4.11. The information objects, which are included within the developed configuration management model, in addition to their associations are:

- *ConfigurableItem*: represents the ability to cover any information object by the configuration management concept. Any object like *ShipForm*, *ShipComponent*, etc. can be considered as a *ConfigurableItem* as it is designed to inherit from this object.

- *ConfigurationItem*: the possibility to combine an unlimited number of objects is realized by means of this information object. The information objects, which are considered as *ConfigurableItem*s, can be treated as parts of a *ConfigurationItem*. The enumerated *Type*-attribute of this information object is utilized to identify the concept reflected by the configuration. The *ConfigurationItem* can be used to reflect different concepts, such as the configuration of all objects related to the *Intact Stability*, *Compartmentation*, etc. as it will be described in the next chapter. Moreover, the 'self-association'-relationship, which relate *ConfigurationItem* to itself as it is shown in Figure 4.11, offers the possibility to use the built *ConfigurationItem*(s) as items and accordingly to be part of new *ConfigurationItem*(s). As a result, a set of *ConfigurationItem*(s) can be combined to form a *ConfigurationItem* which can represent a *Design Solution* as it is depicted in Figure 5.22. Each *ConfigurationItem*, which is a collection of *ConfigurableItem*(s) and/or *ConfigurationItem*(s) can be identified with a name, defined by a person/organization at a specific date/time, assigned by a status (see 4.4.5) in addition to the ability to be supported by user defined attributes. On the other hand, the *ConfigurationItem*, which is a *ManagementObject*, is considered as a *ReportableItem* in order to be covered by the reporting capabilities.

4.5.4. Change

The change management package which is shown in Figure 4.12 is developed respecting the utmost significance of the change concept generally relevant to the design process and especially to the early phase. It represents a means within which changes can be organized and controlled. This package includes the following information objects which form the basis of all developed change management capabilities.

- *ChangeControl*: This object, which is a child of *ManagementObject* plays a central role in the change management model. The combination between the requested change by means of the information object *ChangeRequest* and its related decision (whether it will be approved or rejected) is captured within this information object. Its features provide the ability to enrich the combination with properties such as the person/organization who control the change and the date/time.
- *ChangeableItem*: any object within the ship information model can be considered as 'changeable' when it is designed to inherit from this object, such as the *SpaceElement*, *ShipForm*, etc., as it is shown in Figure 4.12.
- *ChangeRequest*: addresses the change concept as a need. It is requested by a person/Organization at a specific date/time due to a reason which can be described by means of the associated object *RequestJustification*. Additionally requester-remarks can be captured through user defined attributes. Each change request must be associated with an evaluation which describes its implications and effects. This is achieved by means of the relation with the object *ChangeEvaluation*.
- *ChangeEvaluation*: offers the ability to describe the implications which are associated with requesting a change of a *ChangeableItem*. Respecting the overlapping nature of the design elements such as space elements, components, etc. and the interdependencies between the different early design tasks as it has been discussed in section 2.5.1, it is considered that the change implications must be analyzed at data as well as process levels in order to detect the possible affected design elements and design tasks. The properties characterized to this evaluation object are based on the addressing nature of the early design process within this dissertation which is described in detail in the next chapter.
  The addressing of the early design tasks can be briefly described as follows: each task

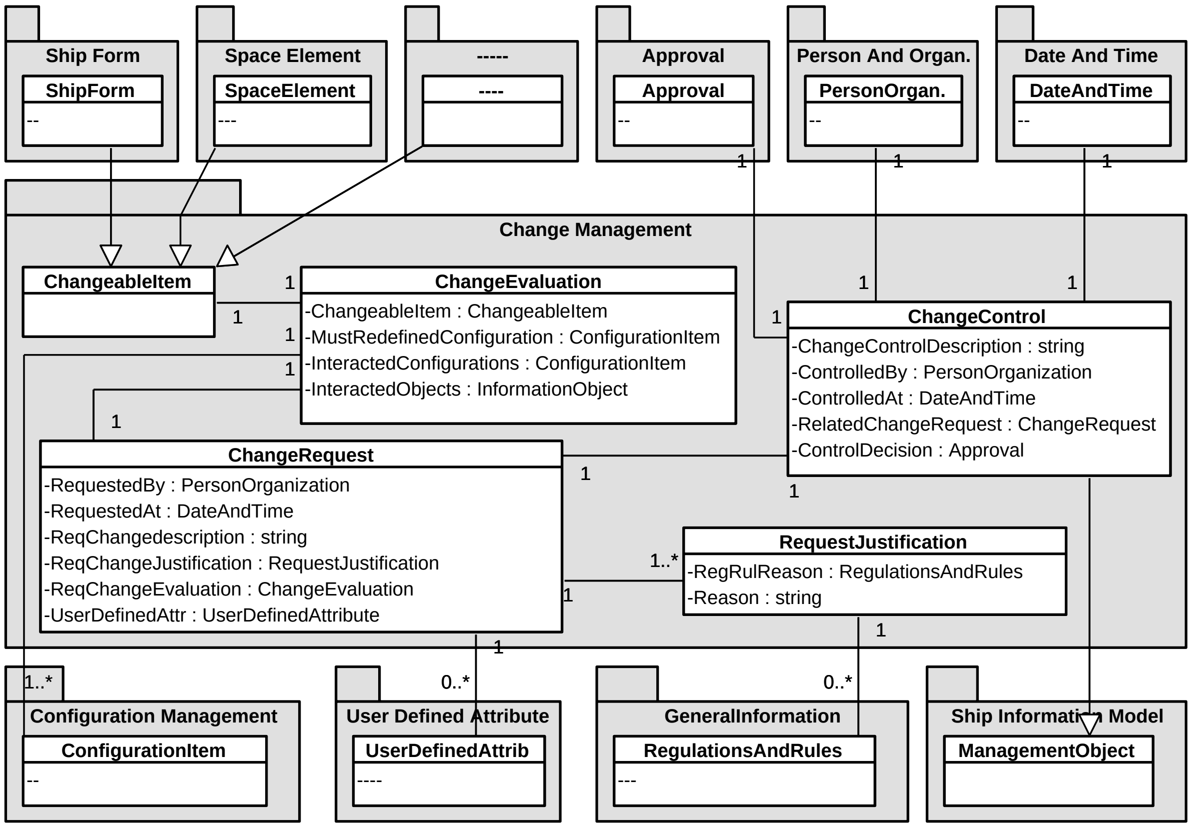

Ship Form
ShipForm
Space Element
SpaceElement
Approval
Person And Organ.
PersonOrgan.
Date And Time
DateAndTime
Change Management
ChangeableItem
ChangeEvaluation
-ChangeableItem : ChangeableItem
-MustRedefinedConfiguration : ConfigurationItem
-InteractedConfigurations : ConfigurationItem
-InteractedObjects : InformationObject
ChangeControl
-ChangeControlDescription : string
-ControlledBy : PersonOrganization
-ControlledAt : DateAndTime
-RelatedChangeRequest : ChangeRequest
-ControlDecision : Approval
ChangeRequest
-RequestedBy : PersonOrganization
-RequestedAt : DateAndTime
-ReqChangedescription : string
-ReqChangeJustification : RequestJustification
-ReqChangeEvaluation : ChangeEvaluation
-UserDefinedAttr : UserDefinedAttribute
RequestJustification
-RegRulReason : RegulationsAndRules
-Reason : string
Configuration Management
ConfigurationItem
User Defined Attribute
UserDefinedAttrib
GeneralInformation
RegulationsAndRules
Ship Information Model
ManagementObject

Figure 4.12.: Change Management Model

is modelled as predefined activity (see 4.5.2) always finalized with defining a *ConfigurationItem* (see 4.5.3), which represents a combination of all information objects used during the performing of a task to play a specific role: input, output, control or mechanism.
In this context, the *ChangeEvaluation*-properties take into consideration that the object, which is requested to be changed, can be one of the information objects, which is used by a task and accordingly is a part of a defined *ConfigurationItem*. Therefore, one or more *ConfigurationItem*(s) can be affected by a change. The *ConfigurationItem*, which is defined by a task (predefined activity) within which the object to be changed plays an output-role, is characterized as *MustRedefinedConfiguration*. It can be summarized that the above evaluation of requesting a change reflects a process level evaluation represented by the interactions between different defined *ConfigurationItem*(s). Furthermore, the data level evaluation is reflected by detecting the information objects which are related to the one requested to be changed. These objects are characterized as *InteractedObjects*.

- *RequestJustification*: realizes the possibility to describe the reasons which lead a user to request a change. The properties which are characterized to *RequestJustification*-object, enable the user to justify the request of a change by defining the reason, which can be, for example, the needs to satisfy rules and regulations (see 4.3.3).

4.5.5. Versions

Considering the previously mentioned importance of improving the versions concept in the early design process (see section 2.5.1), the developed ship information model is supported by the possibility to organize versions of the information objects. Therefore, a *Version Management-*

package is developed as shown in Figure 4.13. It is based on the concepts mentioned in section 3.2.3 and contains the following information objects:

- *VersionableItem*: represents the possibility to define an information object to be versionable. Indeed the following information objects are considered to be versionable: *ShipForm*, *SpaceElement*, *Calculation*, *ShipComponent*, *ConfigurationItem* and *DesignProject*.
- *VersionsHistory*: is the central object within this sub-model. Each *VersionableItem* may have a history of versions which can be dependent on each other. This feature provide the possibility to organize versions in an efficient manner by means of the relation with the objects *DependencyDefinition* and *VersionableItem*, in addition to the association with the support objects *DateAndTime*, *PersonOrganization* and *UserDefinedAttrib*.
- *DependencyDefinition*: offers the possibility to define the dependency between two versions. These two versions can be organized as 'predecessor' or 'successor'. Moreover, the definition of the difference between the both is possible by means of the relation with the *DifferenceDefinition*-object.
- *DifferenceDefinition*: provides the capability to describe the differences between two versions in order to allow a description of their dependency.

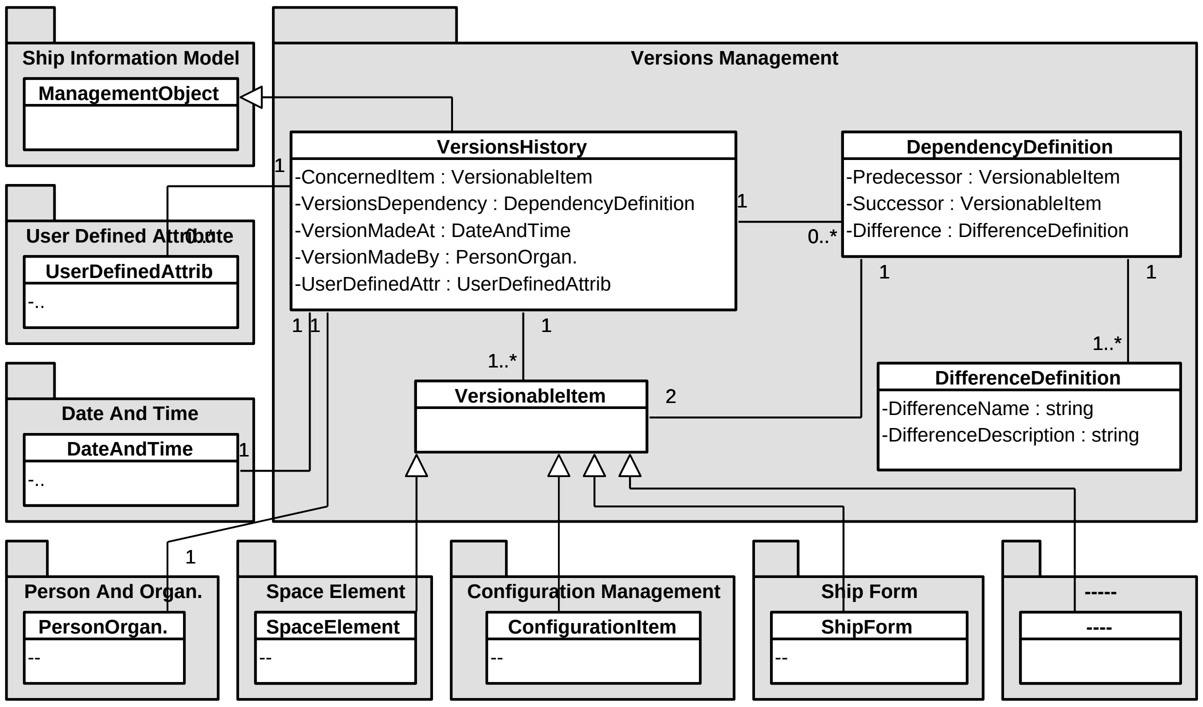


Figure 4.13.: Versions Management Model

5. Ship Early Design Process Model

A general model, which represents the addressing nature of the ship early design process, is depicted in Figure 5.1. The modeling approach is based on the addressing of the activities contained within a process. The activities are analyzed and organized in order to be modelled in predefined forms. These predefined activities interact with each other and with the information objects described in the previous chapter. The predefined activity concept is enhanced in an efficient manner by utilizing the management concepts described in section 4.5.

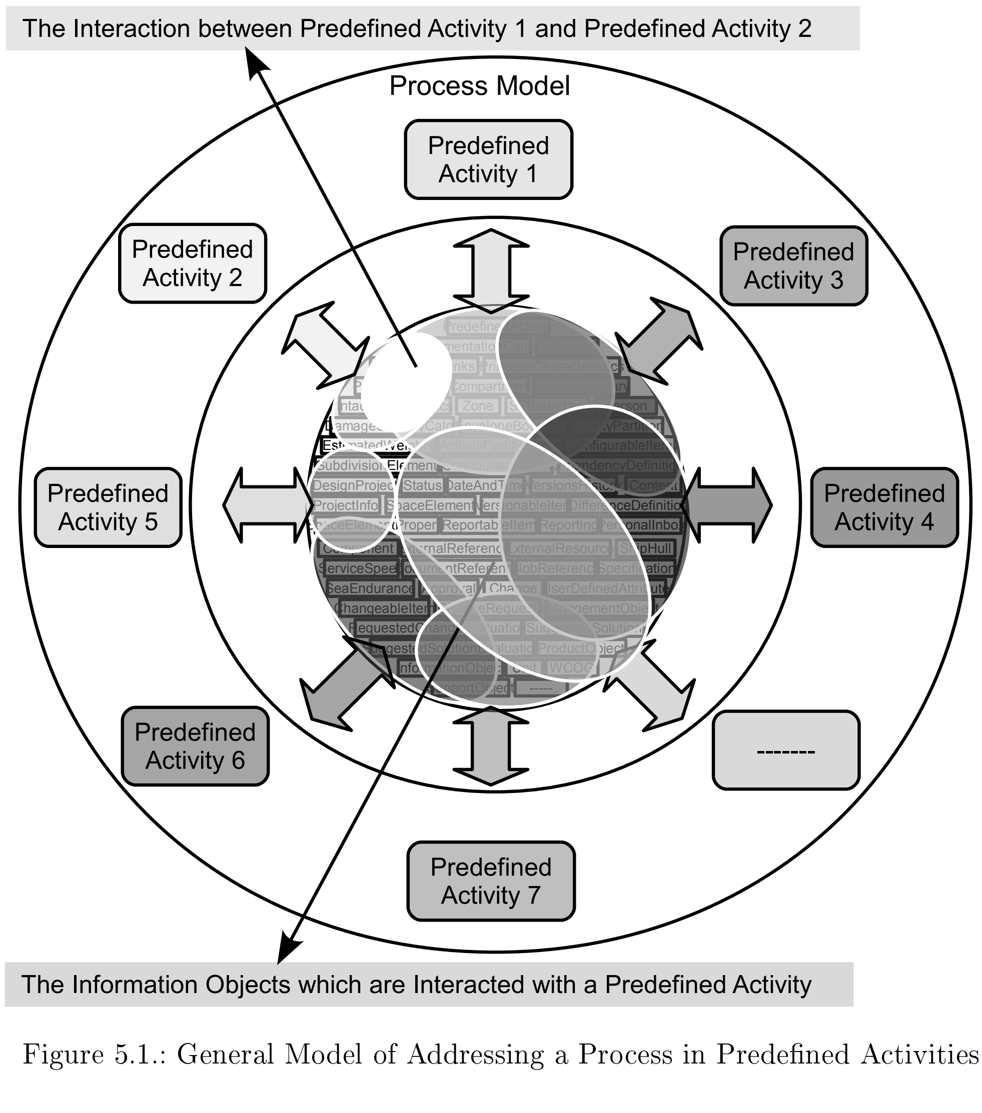


Figure 5.1.: General Model of Addressing a Process in Predefined Activities

In order to gain insight into the methodology applied within the developed model, the following sections are intended to describe:

- the process modeling approach, which is based on the modeling of the activities in predefined forms. Therein, the following aspects are further described: the interaction between

the predefined activities and the interaction between the predefined activities and the information objects in addition to the supporting of the predefined activities by applying the configuration management concept.
- the applying of the developed approach on the ship early design process. Thereby, it is distinguished between two categories of predefined activities: 'Definition/Analysis' and 'Management/Reporting'.

5.1. Process in Predefined Activities

The process, which represents the ship early design, is seen as a collection of predefined activities. These predefined activities reflect the tasks which should be achieved in order to complete the studies of the early design phase. The modeling of these activities from the process point of view is accomplished by utilizing two types of methods: the 'activity-diagram' from the UML-method (such as the right part of Figure 5.2) and the 'ICOM'-method (such as the left part of Figure 5.2). These models have their foundations in the developed ship information model (see chapter 4) as it is discussed by describing the 'Acivity Management'-model in section 4.5.2. During the

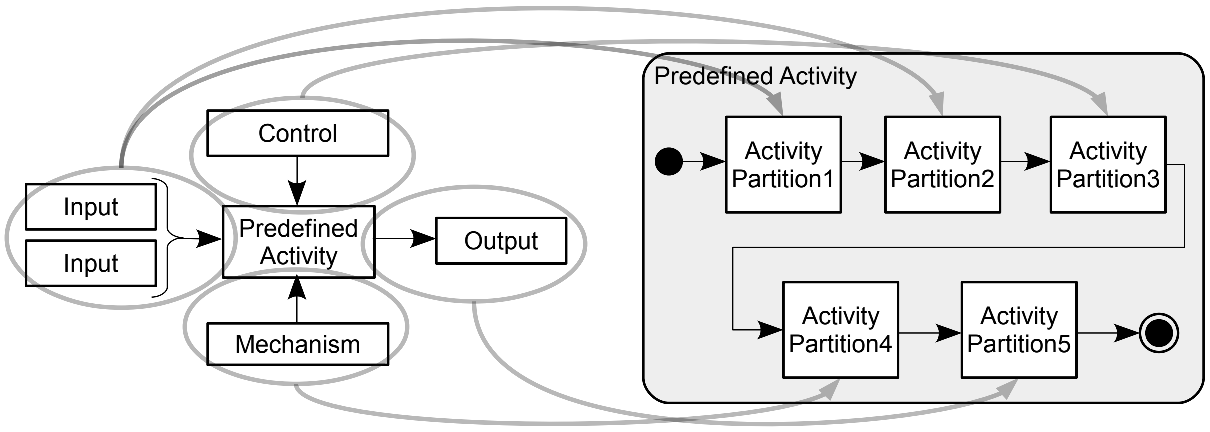


Figure 5.2.: Predefined Activity Consisting of Activity Partitions

modeling, the following three points are taken into consideration:

- Each early design task is represented as a predefined activity. It includes a set of activity partitions, which can be determined from analyzing the task respecting the ICOM-concept. The example depicted in Figure 5.2 shows a *Predefined Activity* composed of five *Activity Partitions.* Two of them reflect the input aspect (regarding the ICOM-diagram) while each of control-, mechanism- and output- aspects is reflected by one partition. In this context, the activity partitions represent the consecutive steps which have to be conducted in order to accomplish the task.
- The representation of the predefined activities and accordingly their partitions by means of ICOM- as well as UML-diagrams offers the ability to exhibit the work-flow aspect within the addressed activity, as it is shown in Figure 5.2, the work-flow can be deduced from the arrows between the related activity partitions. Figure 5.2 shows that the work-flow of performing the *Predefined Activity* starting with *Activity Partition1* and finishing with *Activity Partition5.* It is noteworthy to mention, that the above mentioned utilized diagrams enable the representation of more complex work-flows (which are not always represented as a sequence as it is the case in the simple example depicted in Figure 5.2) within the modelled activities as it will be seen in chapter 7.

- Considering the management and reporting necessities in the early ship design process and the needs of activities which reflect these aspects, the predefined activity concept is used in this dissertation to represent the following two activity-categories:

  Definition/Analysis: i.e. early design tasks which have to be performed within the early design process for definition and analysis purposes, such as *Generate Hull Form*, *Calculate Intact Stability*, etc.

  Management/Reporting: i.e. activities reflecting management and reporting capabilities offered by the developed approach, such as *Create Version*, *Request Change*, *Create Report*, etc.

  These two above mentioned categories of predefined activities will be described in detail in the following (see section 5.2).

5.1.1. Data-Process Integration

Considering the significant benefits which can be obtained by taking into account the integration of data and process within the ship early design phase, a special importance is accorded to this aspect by developing the modeling approach. The activity management model as it is shown

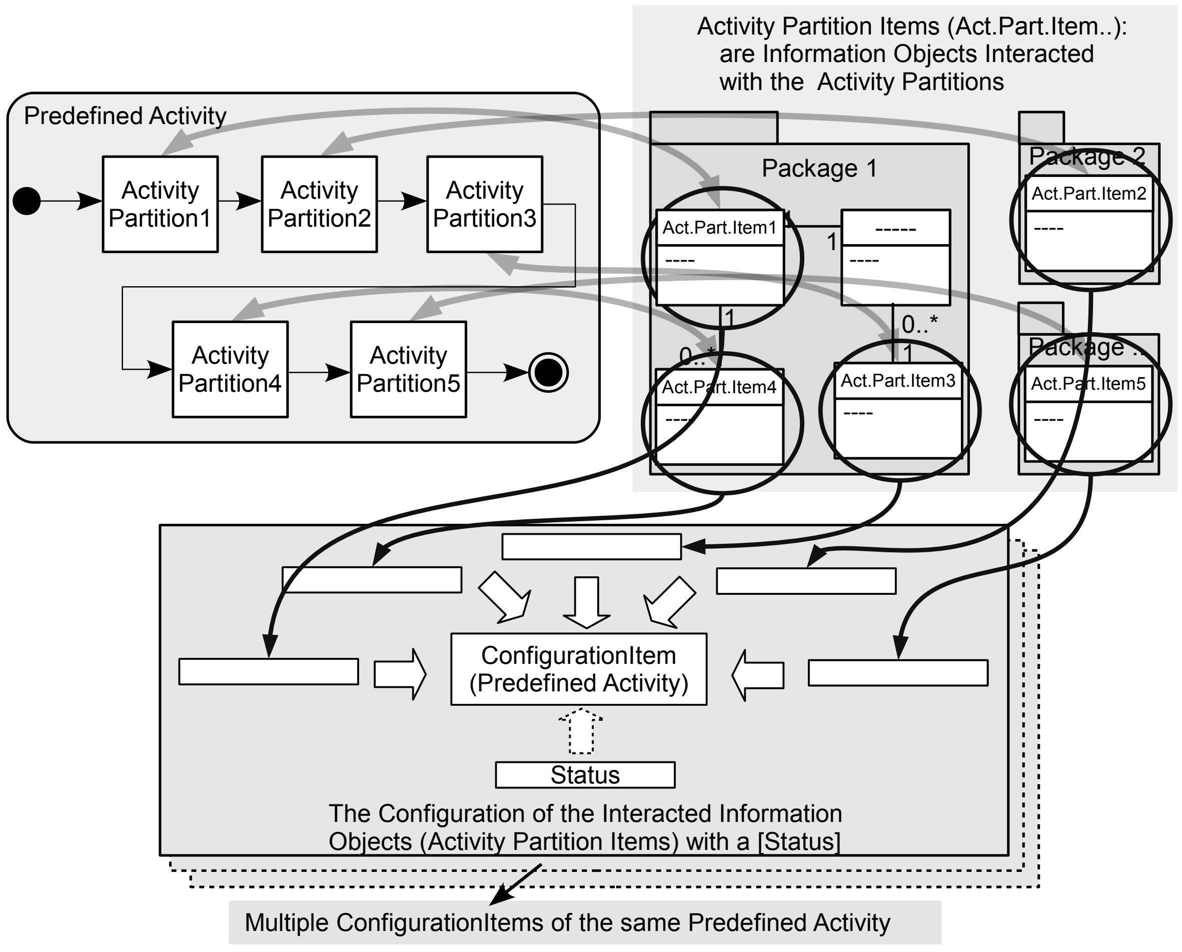

Activity Partition Items (Act.Part.Item..): are Information Objects Interacted with the Activity Partitions
Predefined Activity
Activity Partition1
Activity Partition2
Activity Partition3
Activity Partition4
Activity Partition5
Package 1
Package 2
Package ..
Act.Part.Item1
Act.Part.Item2
Act.Part.Item3
Act.Part.Item4
Act.Part.Item5
1
0..*
ConfigurationItem (Predefined Activity)
Status
The Configuration of the Interacted Information Objects (Activity Partition Items) with a [Status]
Multiple ConfigurationItems of the same Predefined Activity

Figure 5.3.: Data-Process Integration

in Figure 4.10 represents the environment within which the data-process integration concept is achieved. The interaction between the predefined activities and the information objects as it is shown in Figure 5.1 is designed to be achieved at the activity partition level. This is depicted in the upper part of Figure 5.3. It can be seen, that each of the *Activity Partitions*, which are included within the *Predefined Activity* is linked with an information object, which in turn has its own associations with other information objects of the ship information model. By taking the activity management model into consideration, the interacted information objects are characterized to the *Activity Partitions* as *Activity Partition Items* having specific roles such as input, control, etc. As it is shown in Figure 5.3 the *Activity Partition2*, for instance, is interacted with the *Act.Part.Item2*. *Act.Part.Item2* is an information object represents (is characterized as) the *Activity Partition Item* of the *Activity Partition2* from the *Predefined Activity.*

5.1.2. *ConfigurationItem*s in Predefined Activities

The configuration management model (see section 4.5.3) offers the ability to combine different information objects in order to represent a specific concept. It is utilized in a way that fosters the data-process as well as the data-data interaction aspects. As it is depicted in the lower part of

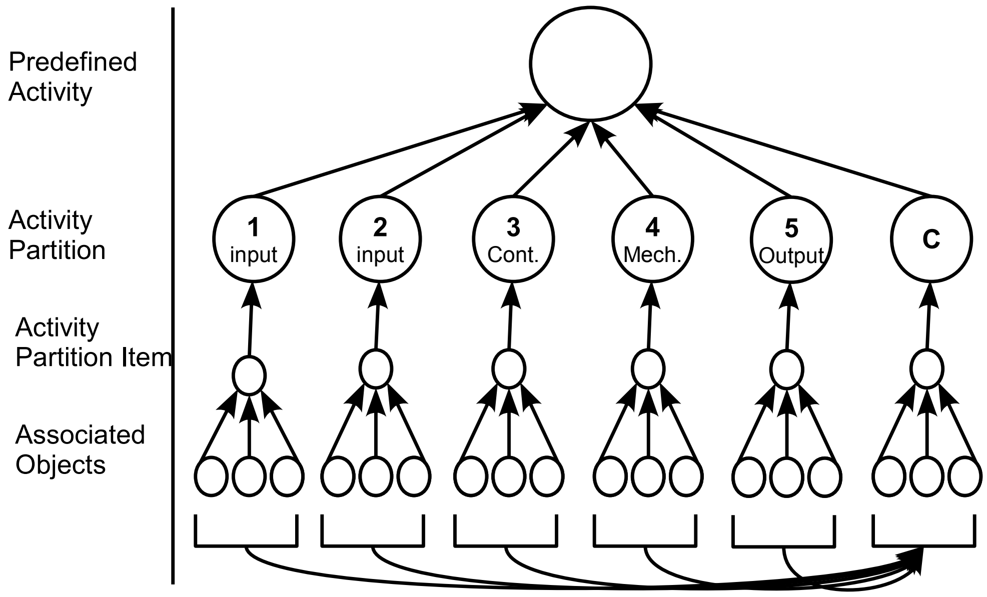


Figure 5.4.: Structure of Predefined Activities, see also Figure 5.2

Figure 5.3, the *Activity Partition Items*, which are information objects interacted by the *Activity Partitions* that are included within a *Predefined Activity* in order to perform a certain role (input, output, control or mechanism), can be related to each other, combined and attributed by a status. The status in this context can be used to reflect and assign a decision-dimension to these combined information objects. It is worth mentioning that the status property is modelled to be an optional one. As a result, in the case that the designer does not have a specific status for the combined information objects, they are combined with the default status which is 'Unspecified'. Furthermore, the possibility to assign a status at any later stage of the process is guaranteed by means of the activity: *Update Status* as it will be described later (see section 5.2.2.2). The transformation of these issues into the information processing system is achieved in the implementation level, see chapter 7. In order to utilize the configuration-concept within the predefined activities, an activity partition, which offers the configuration ability, is added to the partitions of the predefined activities. This partition is named 'C' in Figure 5.4. Considering the above described two predefined activities categories (see section 5.1), 'C' is always added to the partitions of a predefined activity in the case of representing the 'Definition/Analysis' early design

tasks. Respecting the previously mentioned data-process interaction concept, the 'C'-partition is linked to the *ConfigurationItem*-object (see 4.5.3). In this context, if the first five *Activity Partitions*, which are depicted in Figure 5.4, describe the input, output, control, and mechanism of a ship early design task represented as a *Predefined Activity* (see also Figure 5.2), the 'C'-*Activity Partition* is added to the *Activity Partitions* in order to combine all *Activity Partition Items* (interacted information objects) with a known or unspecified status. The term 'Associated Objects' shown in Figure 5.4 refers to the information objects (from the developed ship information model) associated with the information object, which represents the *Activity Partition Item.*
Employing the configuration-concept within the predefined activities has many advantages regarding the ship early design process. These advantages are:

- Efficient combination of the early design task details within a *ConfigurationItem*-object, i.e. input, control, mechanism and output, which are in this context the interacted information objects (*Activity Partition Items*) having the 'input'-,'control'-,'mechanism'- and 'output'-roles.
- The above mentioned combination is effectively used within the developed approach to support:
  - the efficient usability of the combined information objects. As a result, when one of the *Activity Partition Items* of a predefined activity (such as the *LoadingCondition*-object of the activity: *Define Loading Condition*) is used to play an 'input'- or 'control'-role within another predefined activity (such as the activity: *Calculate Intact Stability*), it will be possible to use the different combined *Activity Partition Items* (such as *ShipForm*) of the first predefined activity within the second one (see sections 5.2.1.5 and 5.2.1.6).
  - the interaction concept between different early design tasks represented as predefined activities as it will be seen in the following section,
  - the decision-making concept, i.e. the ability to enrich the combination with the decision-dimension by utilizing the 'status'-concept (more details in section 6.1). Moreover, the 'status'-concept can be used to identify the conformity of a calculation such as intact and damage stability with the relevant regulations.
  - the data-reuse concept by providing an insight into the entire design task details and not just the task-results (more details in section 6.2)
  - the change management concept by providing the possibility to detect the implications of a required change as it will be described in detail in section 5.2.2.4,
  - the version management concept by providing the ability to version the entire design task details represented by the *ConfigurationItem*-object, which will be described in detail in section 5.2.2.3.
- It facilitates the addressing of the repeating-nature of the ship early design process. In this context, multiple *ConfigurationItem*s can be defined for the same early design task represented as predefined activity (see Figure 5.3). These *ConfigurationItem*s have the same name but differ in their 'ID' (which is an implementation issue, see chapter 7)) and may differ in their attributed 'status'.

5.1.3. Predefined Activities Interaction

The process-interaction concept, which reflects the fact, that some tasks within the ship design process and especially early design phase are interdependent and interrelated to each other, is realized within the developed modeling approach. The predefined activity concept, which is used in the next section to represent the ship early design tasks, offers the ability to analyze the represented tasks in order to detect their dependencies. It can be seen that the early design tasks, which are represented as predefined activities, can be:

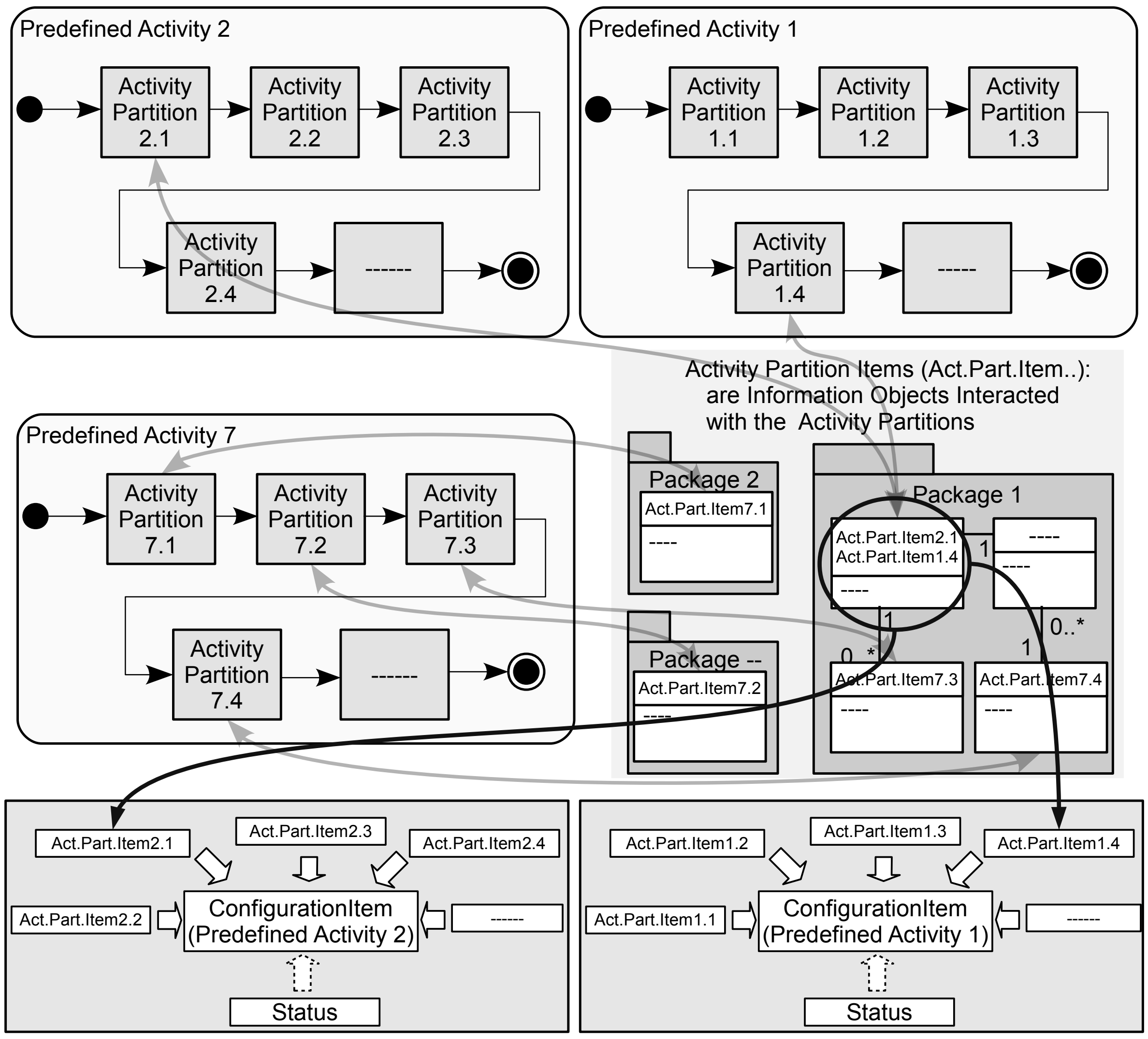


Figure 5.5.: Process Interaction

Dependent: Two predefined activities can be considered as ‘dependent’ when at least one of the ship information model objects is interacted with at least one of the activity partitions from any considered predefined activity, to play input, output or control role. As a result, the same information object can be interacted to represent different *Activity Partition Items* and accordingly to play different roles in different predefined activity partitions. The *Predefined Activity1* and *Predefined Activity2* are dependent to each other as it is shown in Figure 5.1. This dependency, as it is depicted in Figure 5.5, results from the fact, that both activity partitions- 1.4 from *Predefined Activity1* and 2.1 from *Predefined Activity2*- are interacted to the same information object, which represents the *Activity Partition Item 2.1* as well as *Activity Partition Item 1.4*. These *Activity Partition Items* may have, for example, an output-role in the *Predefined Activity1* and an input-role in the *Predefined Activity2*.

Independent: Independent predefined activities can be performed parallel to each other. Two predefined activities can be considered to be ‘independent’, if the activity partitions included within the former predefined activity are interacted with information objects different from those interacted with the activity partitions of the latter one and vice versa. Therefore, as it is shown in Figures 5.5 and 5.1, the *Predefined Activity1* and *Predefined Activity7* can be considered as 'independent' and thus can be performed simultaneously.

By taking into consideration the adopted methodology, which finalizes each predefined activity with a definition of *ConfigurationItem*, it can be stated, that predefined activities interaction is automatically reflected within the built configurations. By following up the example of the interacted predefined activities 1 and 2, the interaction is reflected within the built configurations as it is shown in Figure 5.5. The both *Activity Partition Items* -2.1 and 1.4-, which are represented by the same information object, are parts of the defined *ConfigurationItem*(s) of *Predefined Activities* -1 and 2- but with different roles. This classification of the predefined activities is taken into consideration in the implementation of the early design tasks (see chapter 7) in order to increase the efficiency of achieving the ship early design phase as it will be seen later.

5.2. Ship Early Design Predefined Activities

The previously described process modeling approach (see section 5.1), which is based on the modeling of the activities in predefined forms, is applied to the ship early design process. As a result (and as it has above mentioned), ship early design process is represented in two categories of predefined activities: 'Definition/Analysis' and 'Management/Reporting'. In the following sections a detail description of these predefined activities can be found. The activity partitions in addition to the interacted objects (*Activity Partition Items*) are listed in the tables in Annex A.3.

5.2.1. Definition/Analysis Predefined Activities

Predefined activities which represent early design tasks for definition and analysis purposes are listed within this category of predefined activities. The multiplicity of the techniques that can be applied to achieve the tasks related to the ship early design stage raises difficulties with their modeling as predefined activities. This is based on the fact that the shipyards or design offices have as a matter of fact their own design strategies, which are based on their own software foundations, to perform the early design tasks. Therefore, at this addressing level and by taking into consideration the ICOM-concept, the early design tasks are analyzed by means of their input, control, output and regardless their mechanism. This leads to the possibility to apply the developed modeling approach on the early design process independent of any company specific software environments. As a consequence the early design tasks are represented as predefined activities composed of a number of partitions, which are interacted with the information objects characterized as activity partition items and these predefined activities are always finalized by employing the configuration concept. The mechanism aspect, which reflects the diversity of the tools used within the design process, is taken into consideration in the implementation level. Different scenarios, which represent different strategies, are implemented in order to represent the applicable nature of the developed approach as it will be described in chapter 7.
Most of the design tasks, which have to be performed in the early design phase, are represented as predefined activities in the following sections. Figure 5.6 shows a sketch of these predefined activities which are, *Setup Early Design Project*, *Generate Hull Form*, *Calculate Hydrostatics*, *Create Compartmentation*, *Define Loading Condition*, *Calculate Intact Stability*, *Calculate Damage Stability*, *Estimate Weights*, *Estimate Capacities*, *Calculate Freeboard*, *Predict Resistance*, *Specify Propeller*, *Predict Propulsion*, *Predict Power*, *Define Component* and *Calculate Longitudinal Strength*.
Each of these predefined activities is shown by means of the ICOM-method. Therein the

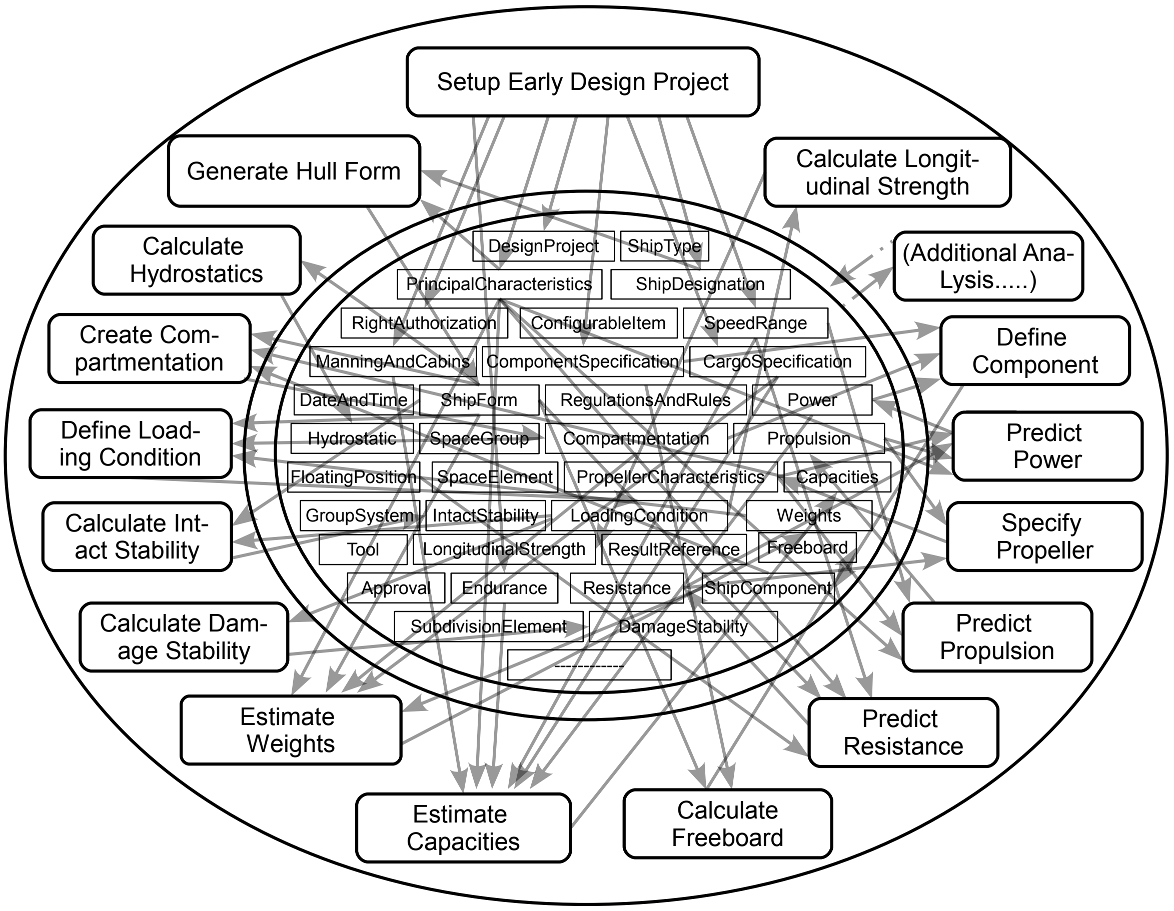


Figure 5.6.: Definition/Analysis Predefined Activities Interacted With Information Objects

'Control'-aspect is utilized to reflect the following issues: specifications, rules and regulations, information identified by the designer to control a calculation, such as the identification of floating positions to control the hydrostatics calculation in addition to the information, which can be gained from other tasks and affect an addressed task. Thus, this information has a controlling role if they exist, but their absence does not prevent the performing of an addressed task. Finally, it is noteworthy to mention that the following predefined activities representing the early design tasks are not capable to manage their results in the case, when the results of a calculation are/are not in conformance with the owner's requirements or other design constraints. These management capabilities within the developed approach are attributed to the 'Management'-predefined activities described in section 5.2.2.

5.2.1.1. Setup Early Design Project

In principle, establishing a design project must be based on owner requirements and the preliminary general information, such as the length between perpendiculars, breadth, etc. The data related to these two aspects are determined through the meetings between the ship owners and shipbuilders, taking advantage of the historical data of the previously built (similar) ships, in addition to the applying some parametric design methods [101] that are used to estimate the ship parameters very early in the design process. These two aspects are represented in gray color in Figure 5.7 in order to refer to the fact, that their data are considered within the developed approach, but the methodology through which these data are gained is not in scope of this research. The obtained data are the starting point for setting up a new design project. Therefore, the main objective of this activity is to organize/store the obtained data within the corresponding information objects in order to be available as a basis for the early design tasks. Proceeding from

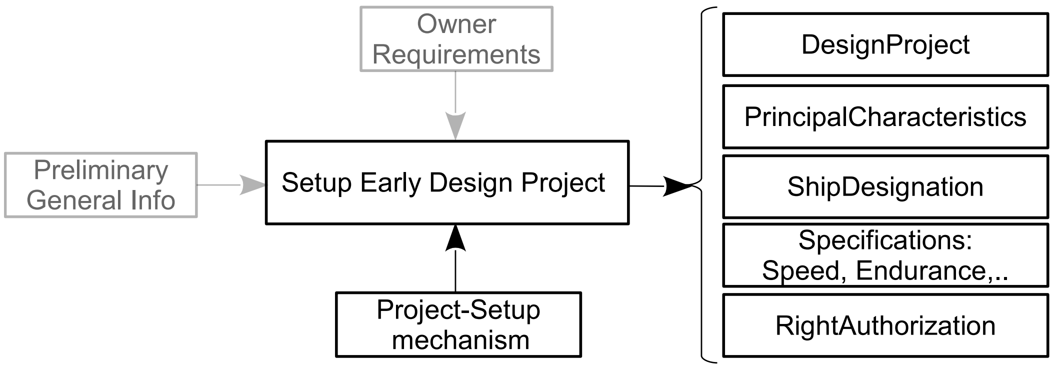


Figure 5.7.: Activity: *Create Early Design Project*

the previous description, the setting up of a new design project is being considered in many partitions, which interacted with the corresponding information objects. These partitions together with their interacted objects are listed in Table A.1. Many information objects like *DesignProject*, *RightAuthorization*, *PrincipalCharacteristics*, *Specification*, etc. are interacting with the activity partitions included within this predefined activity. The partitions can be summarized as follows: At first, a mechanism, which is capable of setting up the design project must be applied. This is followed by four partitions, which are used to specify data regarding:

- *DesignProject*, such as project name (see section 4.3.1)
- *PrincipalCharacteristics*, such as length between perpendicular (see section 4.3.3)
- *ShipDesignation*, such as ship type (see section 4.3.3)
- *Specification*s, such as speed (see section 4.3.2)

The next partition is to specify the tasks responsibilities. As a result each of the early design tasks has to be the responsibility of Person/Organization. The last partition is used to define the configuration with the name `Design Project` (see section 5.1.2), within which all information objects characterized as *Activity Partition Item*s are organized.

5.2.1.2. Generate Hull Form

By taking into consideration the process-interaction concept represented in Figure 5.5, the creation of a preliminary hull form for the ship at the very beginning of the design process can be considered one of the most important tasks during the early design stage. Its importance comes from the needs of a hull-form as a precondition in order to perform many early design tasks, such as to create compartmentation, to predict resistance/power, to calculate hydrostatics, etc. The generating of a hull-form can be seen as a predefined activity. The activity partitions, which are included within this predefined activity, in addition to their linked information objects and their roles regarding the ICOM-concept (see Figure 5.8) are listed in Table A.2. In the following is the description of input, control, mechanism and output concepts:

Input: The input aspect of this predefined activity is indicated in two activity partitions. The former is to get the *PrincipalCharacteristics*, such as the L_{pp}, B, etc. The latter is to get the *ShipDesignation*, such as ship type.

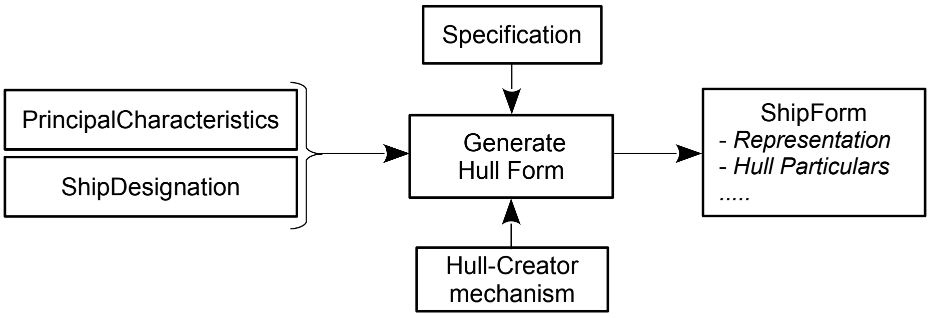


Figure 5.8.: Activity: *Generate Hull Form*

Control: Any special requirements, which can be organized as *Specification* and characterized to the hull form, should be known in order to be taken into consideration. These specifications, which control the generation of the hull form are obtained by means of an activity partition.

Mechanism: the predefined activity is performed by a mechanism. Therefore, after getting the input and control information by means of the previous activity partitions, a partition to apply a hull creator mechanism is performed. This can be achieved in different scenarios as it is discussed in section 7.4.1.2

Output: Two activity partitions are defined to reflect the output of this predefined activity. The first one is to save the resulted ship form within the corresponding information objects (*ShipForm* and its relations with the objects: *Representation*, *HullParticulars*, etc. (see 4.3.5)). It is important to mention that the storing of the results is achieved by exploiting the properties of these objects, which are in fact the objects of the *Ship Form*-package. The second one is to close the activity by defining the `Ship Form`–configuration.

5.2.1.3. Calculate Hydrostatics

Center of buoyancy in longitudinal as well as vertical directions, longitudinal center of flotation, displacement, etc. are hydrostatics data as functions of floating position, which have to be calculated very early in the design process due to the needs of these properties especially in stability calculations. The analysis of this task by means of the ICOM-concept, as it is depicted in Figure 5.9, leads to the possibility to describe it as a predefined activity.

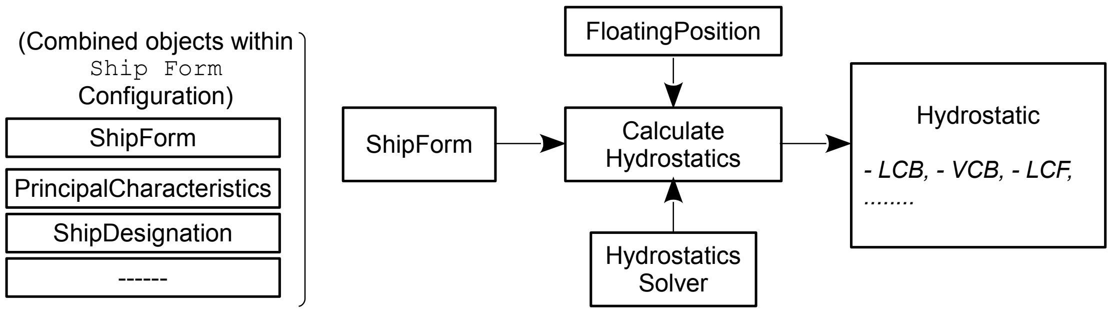


Figure 5.9.: Activity: *Calculate Hydrostatics*

The activity partitions, which are included within this predefined activity, in addition to the information objects which are interacting with them as well as their roles are shown in Table A.3. What follows is the description of input, control, mechanism and output concepts:

Input: Considering the fact, that the ship form, which is created within a previous task, with its characteristics are needed in order to calculate the hydrostatics, one activity partition is defined, whose purpose is to provide the *ShipForm* and accordingly the input data of the calculation. Regarding previously described advantages of the configuration management concept (see section 5.1.2), it is worth mentioning that the usage of the *ShipForm*-object to provide the input of this predefined activity (see Figure 5.9) refers to the possibility of using any of the interacted information objects such as *PrincipalCharacteristics* combined within the defined `Ship Form`–configuration.

Control: One activity partition, which enables the definition of the *FloatingPosition*(s) i.e. the heel angle, trim angle, amidships draft, etc., is applied to reflect the control-aspect of the calculation. It specifies the boundary conditions at which the hydrostatics have to be calculated.

Mechanism: After providing the input and control information, the next partition is to apply a hydrostatic solver to perform the calculation. It saves the applied solver within the *Tool*-object (see 4.4.3) with the mechanism-role.

Output: Two partitions are added to reflect the output-aspect: The former is to save the resulted hydrostatics by making use of the properties of the *Hydrostatic*-object, such as LCB, LCF, etc. in addition to the possibility to define a link to the hydrostatics properties within the used hydrostatic solver (see Figure A.9 and section 4.3.11). The latter is to define a `Hydrostatic`–configuration (see section 5.1.2) in order to close the calculation.

5.2.1.4. Create Compartmentation

The 2D-representation of the general arrangements document and accordingly the ship-compartments at the early stage, has a non optimal impact on the subsequent design stages. This is due to the simple consistence between the compartments and other design elements such as components as well as between the compartments themselves, the low level of detail of properties associated with compartments, the lack of possibility of integration as well as usefulness of these properties by the subsequent design tasks, for example for the naval architectural calculations (Calculate stability, estimate weight, etc.). Therefore, the *Create Compartmentation*-task is

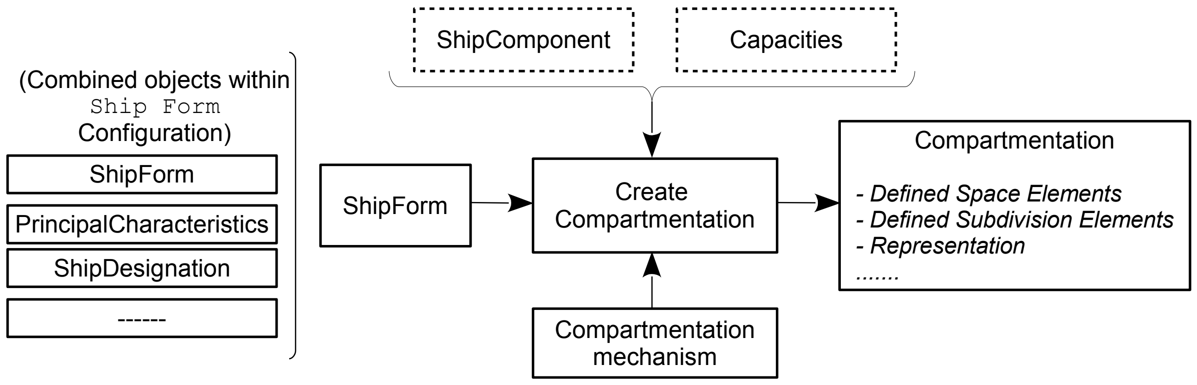


Figure 5.10.: Activity: *Create Compartmentation*

addressed and analyzed from the 3D-representation point of view. This task, which is highly related to the modeling methodology, is analyzed regarding ICOM-concept (see Figure 5.10). This leads to the ability to be modelled in a predefined form regardless the applied compartmentation-approach (mechanism). The activity partitions together with their partition-items are listed in Table A.4. Input, control, mechanism and output concepts are described in the following:

Input: The input-aspect is indicated in one activity partition. It is capable to provide the *ShipForm* created by the *Generate Hull Form*-task in order to be subdivided.

Control: Two activity partitions are modelled to reflect the control-aspect of this task. The purpose of these two partitions is to check and obtain the requirements, which are issued by other tasks and affect the creation of the compartmentation. One of them is to handle the required *Capacities* (see section 5.2.1.9) and the other one is to deal with the required *ShipComponents*-spaces (see section 5.2.1.12). The modeling of these two objects as dashed lines boxes in Figure 5.10 means, that they represent details, which, if they are considered within the creation (according to their availability), will increase the compartmentation efficiency and accuracy. The absence of these details does not prevent the creation of the compartmentation which can be performed, in this case, with respect to the designer-experience, history data, etc. but the accuracy level of the compartmentation-results will be decreased.

Mechanism: After providing the input as well as control data, an activity partition to apply a compartmentation mechanism is added. This mechanism should be capable of providing an efficient generation of a 3D-representation of the compartmentation in the early design stage.

Output: The last two activity partitions are to reflect the output-aspect. The first is to save the compartmentation results within the corresponding objects and the second is to close the task with the definition of the `Compartmentation`-configuration. The *Compartmentation*-object from the *Ship Calculation*-package (see 4.3.11) shown in Figure 4.7, together with its associated objects are used to save the results of this task. Making use of their properties offers the ability to save:

- the *SubdivisionElements* (together with their properties, see section 4.3.6), which are defined in order to create the compartmentation.
- the defined *SpaceElements* in groups, such as the ballast water tanks, heavy fuel tanks, etc. Furthermore each of these groups can be characterized by properties, such as volume, weight, LCB, VCB, representation, etc., in addition to the *SpaceElements* individually (with their properties, see 4.3.9), which are included within a space group.
- the compartmentation shape, representation or any other details, which can be obtained from the compartmentation mechanisms in formats such as XML, VRML, etc., (see *ResultReferenc* in 4.4.3 and *Representation* in 4.4.1).
- Moreover, it is possible to define a pointer to the data of the compartmentation-results within the mechanism's own data management component.

5.2.1.5. Define Loading Condition

In order to identify the loading of a ship, which describes the cargos and commodities such as fuel, ballast water, etc. at a specific time, loading conditions are defined. Specific rules as e.g. defined in the IMO 'IS-Code' and 'SOLAS' have to be observed in order to define the loading conditions. These specific defined conditions such as departure condition, arrival condition, etc., are applied to check the conformity with criteria in ship design, such as the intact and damaged stability SOLAS. In Figure 5.11, the defining of loading conditions is depicted. The activity partitions together with the interacted objects, which are counted among this task as predefined activity are listed in Table A.5. Two activity partitions are needed to identify the definition rules (thereby, the possibility to take into consideration special *Specifications* is insured) and accordingly to reflect the control aspect, as well as to apply a mechanism which offers the capability to define loading conditions. The input and output concepts are:

Input: Three activity partitions are added to reflect the input aspect. They are used to check

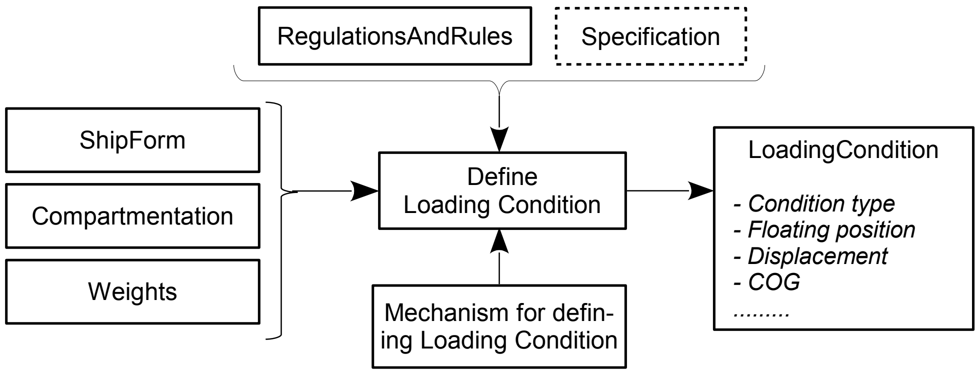


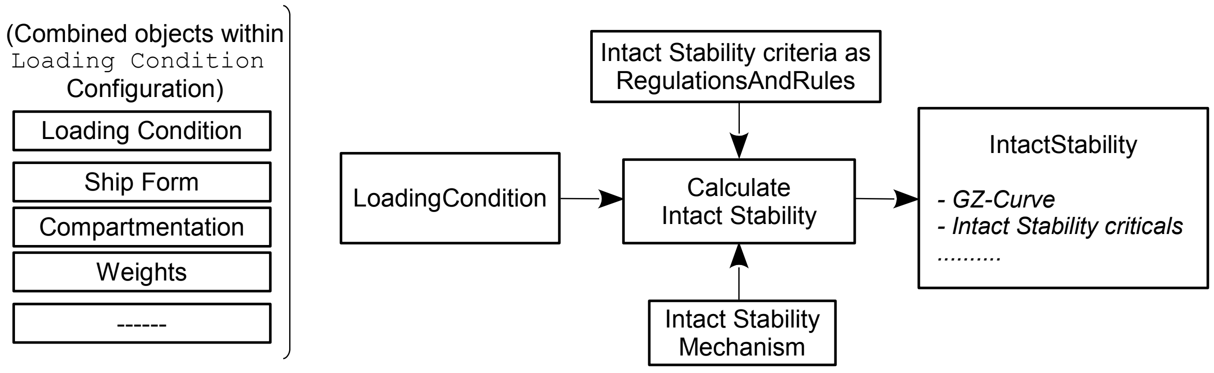
Figure 5.11.: Activity: *Define Loading Condition*

the availability and consequently to obtain the defined *ShipForm*, *Compartmentation* and *Weights*.

Output: The output is represented by means of two partitions. They are capable of saving the results of this task within the suitable information objects and defining the `Loading Condition`-configuration. The condition type such as departure condition, arrival condition, etc., floating position, displacement, COG, etc., are properties characterized to the *LoadingCondition*-object, which is part of *Ship Calculation*-package (see Figure A.9).

5.2.1.6. Calculate Intact Stability

The purpose of this task is to check the designed ship with respect to the IMO intact stability criteria. In order to achieve this task, relevant data have to be provided. The depiction of this task by means of ICOM-concept is shown in Figure 5.12. The activity partitions, which have to be performed linked to the interacted information objects are listed in Table A.7. In addition to the partitions to represent the input and output aspects described below, two partitions to reflect the control and mechanism aspects are modelled to allow an integration into the shipyards software network.


(Combined objects within Loading Condition Configuration)
Loading Condition
Ship Form
Compartmentation
Weights

LoadingCondition
Intact Stability criteria as RegulationsAndRules
Calculate Intact Stability
Intact Stability Mechanism
IntactStability
- GZ-Curve
- Intact Stability criticals
....

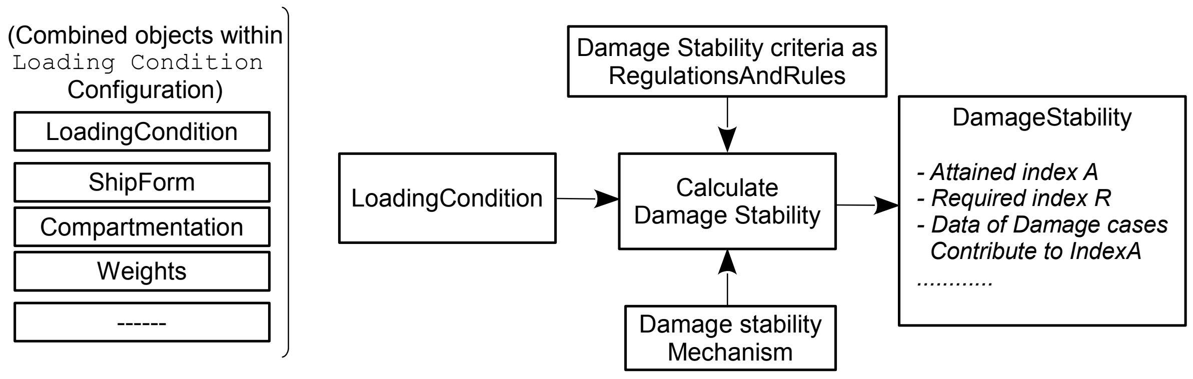
Figure 5.12.: Activity: *Calculate Intact Stability*

Input: Considering the fact, that the design should be examined at specific loading conditions with respect to the IMO intact stability criteria, an activity partition to check the availability and to provide these loading conditions (in the case the definitions of relevant loading condition exist (see predefined activity: *Define Loading Condition*)) is modelled. By taking into account the advantages of the configuration management concept described previously (see section 5.1.2), it is noteworthy to mention that the usage of the *LoadingCondition*-object in order to provide the input of this predefined activity (see Figure 5.12) refers to the possibility of using any of the interacted information objects such as *ShipForm*, *Compartmentation*, etc., combined within the defined `Loading Condition`-configuration.

Output: In addition to the activity partition to define the `Intact Stability`-configuration, a partition to save the results of this task within the *IntactStability*-object (see section 4.3.11 and Figure A.9) is modelled. The properties of this information object are capable to handle the results of this task in different ways. In addition to the possibility to define an indicator to the results stored within the applied mechanism and the possibility to handle the results as they are gained from the applied mechanism, it is possible to save the key details of the analyzing results explicitly. As a result the data of the GZ curve, required and resulted intact stability criticals such as 'Area under GZ curve up to 30 degrees', 'Area under GZ curve from 30 to 40 degrees or downflood', 'IMO Weather Criterion (Maximum Initial Angle Of Heel)', etc., are stored. It is worth mentioning, that the conformity of the calculation with the IMO intact stability criteria can be identified by making use of the status concept within the defined `Intact Stability`-configuration.

5.2.1.7. Calculate Damage Stability

The limitations of the floatability of the designed ship in damage conditions have to be determined in the early design stage. Therefore, checking the designed ship against the related rules and regulations of SOLAS is an essential early design task, see Figure 5.13. The description of


(Combined objects within Loading Condition Configuration)
LoadingCondition
ShipForm
Compartmentation
Weights

Damage Stability criteria as RegulationsAndRules
LoadingCondition
Calculate Damage Stability
Damage stability Mechanism
DamageStability
- Attained index A
- Required index R
- Data of Damage cases Contribute to IndexA
....

Figure 5.13.: Activity: *Calculate Damage Stability*

this task as a predefined activity is shown in Table A.7. The information, which are required to check the damage stability, are supplied by means of an activity partition. In addition to the previous one which reflects the input aspect, two activity partitions to reflect control and mechanism aspects are added. Reflecting the output aspect, the results of this task are saved by means of two activity partitions. The first one is to interact with the information object *DamageStability* in order to save the results and the last one is to close the task by means of applying the configuration concept. The properties characterized to *DamageStability*-object, which is part of *Ship Calculation*-package offer the ability to save the key results of checking the damage stability: 'Attained index', 'Required index' and data of damage cases, which contribute to the attained index. By making use of the status concept within the defined `Damage Stability`-configuration, it is possible to identify the conformity of the calculation with the SOLAS-regulations. Moreover, it is possible to define a pointer to the stored data of damage stability within the applied mechanism.

5.2.1.8. Estimate Weight

The reasonable estimation of the ship weight throughout the design process is a most important task due to its considerable impact on the overall ship design [40]. The ship-weight is divided into two categories: deadweight, which refers to the payload and all consumables including ballast water and the lightship weight[113]. This task, which is shown in Figure 5.14 is modelled as

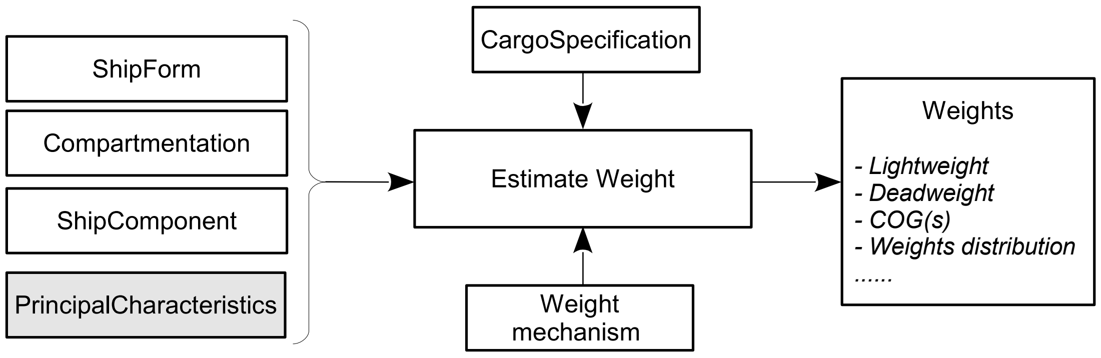


Figure 5.14.: Activity: *Estimate Weight*

a predefined activity composed of several activity partitions listed in Table A.8. Considering the developed process modeling approach, it is noteworthy to mention, that the purpose of this predefined activity is to estimate the ship weight while the capability to control the weight is attributed to the 'Management'-predefined activities described in section 5.2.2. In addition to an activity partition to apply a suitable mechanism, activity partitions to reflect input, control and output are modelled:

Input: The method, which is used to estimate the ship-weight, is dependent on the available data. Therefore, the input of this task takes into consideration the fact of low level data in the early design phase and a steadily increasing data volume as the design process proceeds. Thus, activity partitions to provide available data of *ShipForm*, *Compartmentation* and *ShipComponent*s regarding weights estimation are added.
In case these data are not available, which is the case at the beginning of the design, shipyard's data of previously built ships or empirical methods are utilized to estimate ship-weights in the very beginning of the design process [42]. Considering the fact that these methods are based on the main characteristics of the addressed ship in order to estimate the weight, an activity partition to provide the *PrincipalCharacteristics* is added. Its purpose is to supply the *PrincipalCharacteristics* data in case, that the previously outlined data: *ShipForm*, *Compartmentation* and *ShipComponent*s are not available. Therefore, the *PrincipalCharacteristics*-object is depicted in a different color in Figure 5.14.

Control: *CargoSpecification*s are identified within the setting up of the design project (see activity: *Setup Early Design Project*) and should be used by the estimation of the deadweight. Therefore, an activity partition to provide these data is added.

Output: The properties of the *Weights*-object, which is a part of *Ship Calculation*-package, allows for saving the lightweight, deadweight, COG and weights distribution explicitly in addition to the possibility to define a pointer to the weights results stored within the applied mechanism. Therefore, two activity partitions to save the results to this object and to close the estimation by employing the configuration concept are modelled.

5.2.1.9. Estimate Capacities

The purpose of this task is to estimate the volumes of needed tanks such as HFO, FW, etc. as well as the volumes of some spaces such as the volume of the main engine room. The estimated volumes can be used to control the compartmentation in order to increase the accuracy, decrease the mistakes and accordingly the potential changes within the design process. The capacity management issues, such as the comparing of the needed and existing capacities and accordingly making a decision e.g. to request a change, can be performed by making use of the 'Management'-predefined activities described in section 5.2.2. An example of such management capabilities will be described later. The depiction of this predefined activity is shown in Figure 5.15. In addition

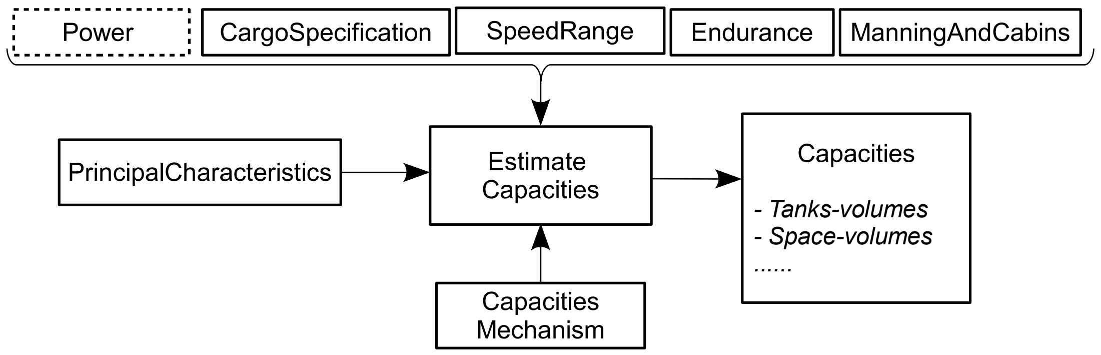


Figure 5.15.: Activity: *Estimate Capacities*

to an activity partition to reflect the mechanism-aspect, several activity partitions are included within this task to reflect the following input, control and output aspects. The activity partitions are listed in Table A.9.

Input: Paying attention to the fact that the estimation can be performed by empirical methods based on the ship main characteristics, an activity partition to provide the *PrincipalCharacteristics* as input data is added.

Control: Several partitions are modelled to provide the specifications for *Endurance*, *SpeedRange*, *ManningAndCabins* (to identify the number of crew as well as passengers) and *CargoSpecifications* (to control estimating the volume of payload spaces). In addition to an activity partition to provide the predicted *Power* (in the case that it exists) for estimating the volume of HFO-tanks is modelled.

Output: Two activity partitions are added to reflect the output aspect of this task. The first one is to save the estimated capacities within the *Capacities*-object, which is a part of *Ship Calculation*-package. The second one is to close the estimation by defining the `Capacities`-configuration.

5.2.1.10. Calculate Freeboard

The purpose of this task is to calculate the freeboard characteristics of the designed ship in accordance with the International Load Line Convention (ILLC). The determination of these characteristics is an essential issue in the design process. Its conditions must be satisfied by any designed ship. The depiction of this task depending on ICOM-concept is shown in Figure 5.16. The included activity partitions within this task as a predefined activity can be summarized as follows (see Table A.14): each of input, control and output aspects is reflected by one partition in order to (respectively) check the availability and retrieve the *ShipForm* (which

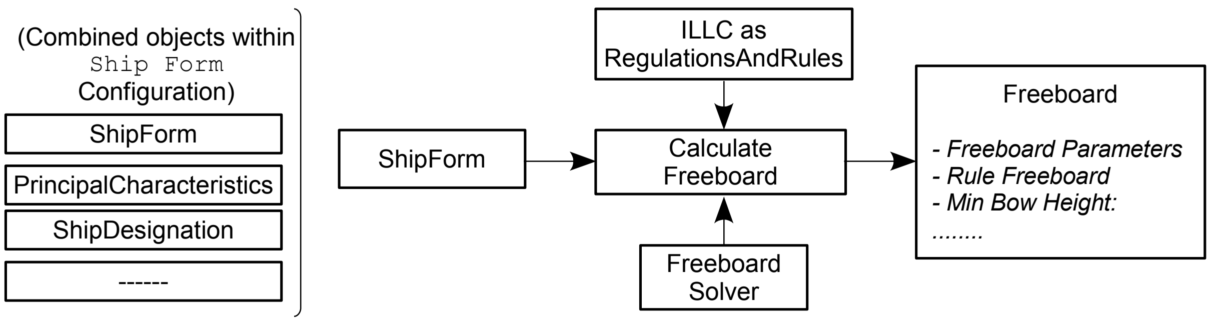


Figure 5.16.: Activity: *Calculate Freeboard*

is formed by *HullForm* and *Superstructure*, see section 4.3.5) and accordingly to retrieve the *PrincipalCharacteristics* and ship type from *ShipDesignation*-object (see Figure 5.16), to make the ILLC-*RegulationsAndRules* (on which the calculation is based) available, and to apply a freeboard solver. These are followed by two partitions (to reflect output aspect). The first is to store the calculated freeboard depending on the *Freeboard*-object, which is a part of *Ship Calculation* package. This object allows for the saving of many characteristics such as rule freeboard, freeboard length, minimum bow height, ship type according to ILLC (A, B, B-60), etc. The second is to finish this task by defining the `Freeboard`-configuration.

5.2.1.11. Hydrodynamics/Powering

The prediction of hydrodynamic and powering properties very early in the design process is an essential issue for the design studies. These properties, which comprise of resistance, propulsion, propeller, power, seakeeping and maneuverability, are estimated and specified by means of several tasks strongly interrelated to each other. Most important hydrodynamics/powering tasks: *Predict Resistance*, *Predict Propulsion Parameters* and *Predict Power* are represented in the following sections in predefined forms. Moreover, the developed ship information model is capable to store the propeller parameters resulted from the design propeller task. As it is shown in Figure 4.7, the *PropellerCharacteristics*-object offers the possibility to store the propeller data such as number of propellers, propeller diameter, pitch ratio, open water propeller efficiency, etc. in order to be used by other tasks. The *HydrodynamicAndPowering*-object is shown in Figure 4.7. It is a part of *Ship Calculation*-package (see 4.3.11) and represents the parent object of all other hydrodynamic and powering objects. This object offers the possibility to also store the model basin results of any of its child objects.

5.2.1.11.1. Predict Resistance

The estimation of the total ship resistance is an essential issue in the early design phase. This task is depicted in Figure 5.17. The activity partitions related to this task as predefined activity are listed in Table A.11. Regarding the risk associated with a wrong estimation, it ought to be determined with the highest possible accuracy. The accuracy is highly dependent on the applied prediction methodology: empirical methods, potential theory, etc. which in turn depends on the level of available details. Therefore, two activity partitions are modelled to reflect the input aspect of this task. One is to check the existence and accordingly to select and obtain a *ShipForm*, which is the basis for the advanced prediction methods. The second partition is to provide the *PrincipalCharacteristics* in the case when no ship forms are existing at the time of prediction the resistance. These characteristics are the basis for the prediction by means

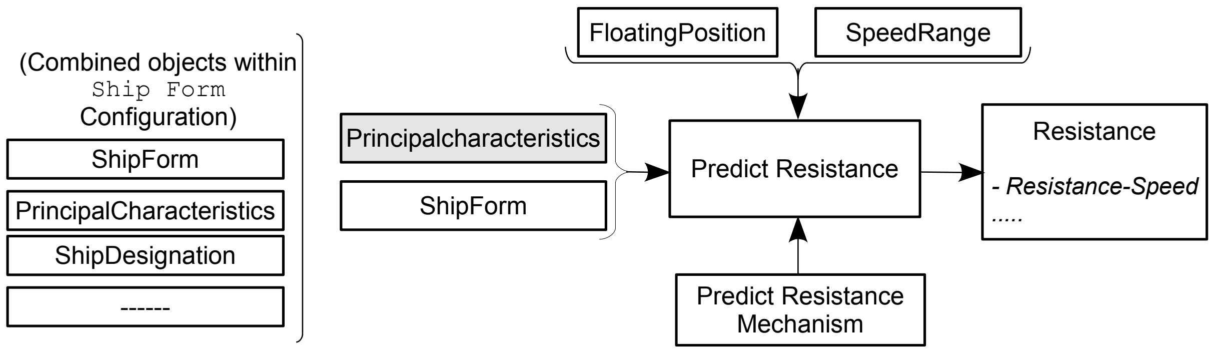


Figure 5.17.: Activity: *Predict Resistance*

of empirical methods. The control-aspect is represented by two partitions, which are used to provide the specified *SpeedRange* as well as to specify *FloatingPositions* at which the calculation should be performed. In addition to the activity partition to apply the prediction mechanism and the partition to finalize the prediction with the configuration definition, a partition to save the resulted resistances within the *Resistance*-object shown in Figure 4.7 is modelled.

5.2.1.11.2. Predict Propulsion Parameters

The purpose of this task, which is shown in Figure 5.18, is to predict the propulsion properties: wake fraction coefficient, wake field, thrust deduction coefficient, hull efficiency and rotative efficiency. These properties, which are needed to specify the propeller as well as to predict the power, are influenced: by the number of ship propellers (gained from *PropellerCharacteristics*), by *FloatingPosition* and by the *SpeedRange*.

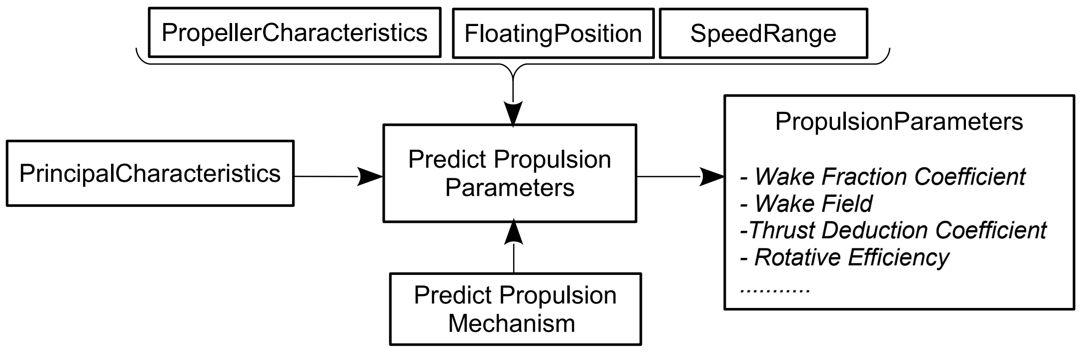


Figure 5.18.: Activity: *Predict Propulsion Parameters*

They can be determined by means of model basin tests or estimated depending on the ship *PrincipalCharacteristics*. This description is reflected within the modeling of this task as a predefined activity composed of several activity partitions (see Table A.12).

5.2.1.11.3. Predict Power

The modeling of this task considers the fact that, during the design process and especially in the early stage, the power should be predicted efficiently and accurately due to its great influences on many decisions made within this stage such as the identification of the main engine system. This task is shown in Figure 5.19. What follows is the description of the activity partitions, which are included within this task as a predefined activity and listed in Table A.13. The partitions,

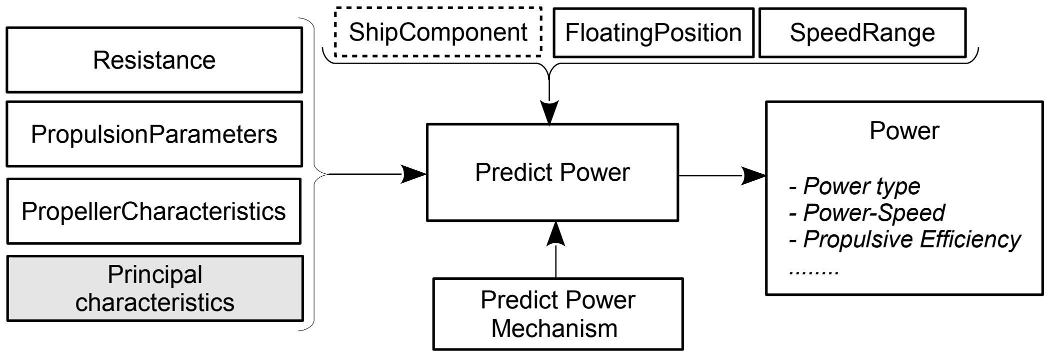


Figure 5.19.: Activity: *Predict Power*

which reflect the input-aspect take into consideration that the methodology which can be applied to predict the required power, is depending very much on the available detail level of data at the time of prediction. Therefore, three partitions to check the availability and accordingly to obtain data about the *Resistance*, *PropulsionParameters* and *PropellerCharacteristics* are introduced. Moreover, an activity partition to retrieve the *PrincipalCharacteristics* of the ship (in the case, when the above mentioned information are not available), which are used by the empirical methods to estimate the power, is modelled. The control aspect is represented by three partitions: to provide the *SpeedRange*, *FloatingPosition* and the shaft efficiency (if it is available from *ShipComponent*-object). One partition is modelled to apply a mechanism capable of predicting the power depending on the provided data. The *Power*-object (see Figure 4.7) offers the ability to store the output of this task: the values of the predicted power (whether it is effective (PE), delivered (PD) or brake power (PB)), the data of the power-speed diagram in addition to the data of the propulsive efficiency at different speeds. This is achieved by means of an activity partition, which reflects together with the final partition to define the corresponding configuration, the output aspect of this task.

5.2.1.12. Define Component

The purpose of this task is to define major ship components specially important already in the early ship design regardless their type like machinery or outfitting as it has been discussed previously (see section 4.3.10). This task, which is shown in Figure 5.20 respecting the ICOM-

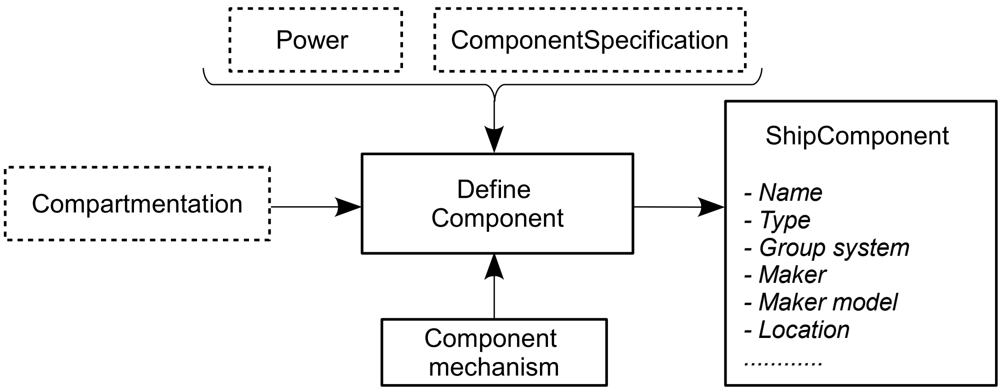


Figure 5.20.: Activity: *Define Component*

concept, is modelled as a predefined activity composed of a number of activity partitions (see Table A.10). As it is depicted in Figure 5.20, any available data which can affect the decisions made during the definition of ship components should be provided. Therefore, three partitions (one to reflect input and two to reflect control aspects) are modelled to retrieve data about *Compartmentation*, predicted *Power* in addition to any potential specifications regarding the ship components like e.g. formulated in the makers list being part of the contract specifications. One partition is responsible for applying a mechanism, which is capable of defining the ship components. Two activity partitions are introduced to reflect the output aspect of this task, the first is to store the defined component within the *ShipComponent*-object, and the second is to close the definition by applying the configuration concept.

5.2.1.13. Calculate Longitudinal Strength

The calculation of longitudinal strength in different load conditions can be considered as a bridge between ship early design stage and the following stage. The purpose of this task is to calculate sheer forces and bending moments, which act on the ship as a result of the weight and buoyancy distribution in the calm water condition. This task is represented in Figure 5.21 making use

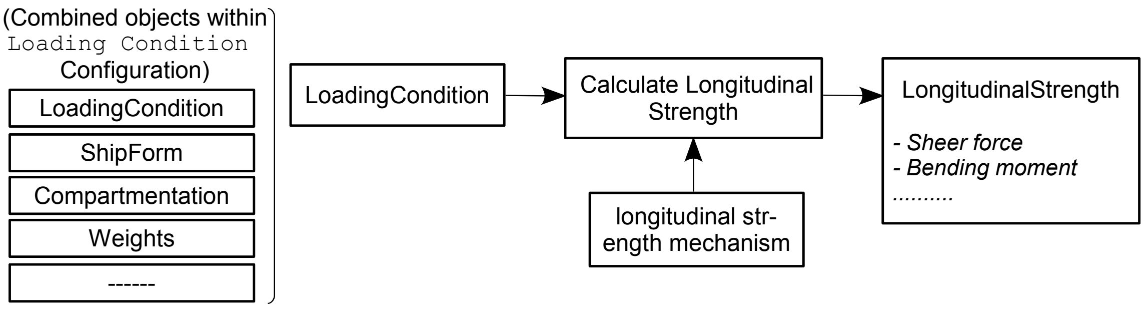


Figure 5.21.: Activity: *Calculate Longitudinal Strength*

of ICOM-concept and in Table A.15 as a predefined activity. In order to provide the input of this task, an activity partition to obtain the *LoadingCondition*, at which the calculation will be performed, is modelled. This is followed by three other partitions, which are used to apply mechanism, store the resulted sheer forces and bending moments in an efficient way by means of the *LongitudinalStrength*-object (see section 4.3.11 and Figure A.9) and finally to define the corresponding configuration.

5.2.2. Management/Reporting Predefined Activities

Activities represented in predefined form and intended to reflect/support management and reporting capabilities in the ship early design process are listed within this category of predefined activities. In the following is the description of these predefined activities: *Create Design Solution*, *Update Status*, *Create Version*, *Check Change Influences*, *Request Change*, *Approve/Reject Requested Change* and *Create Report*.

5.2.2.1. Create Design Solution

As it has been described previously (see section 5.2), several tasks have to be performed in an iterative process in order to achieve a design goal meeting the specifications and formulated objective functions. These tasks are interrelated to each other in a complex network and often in

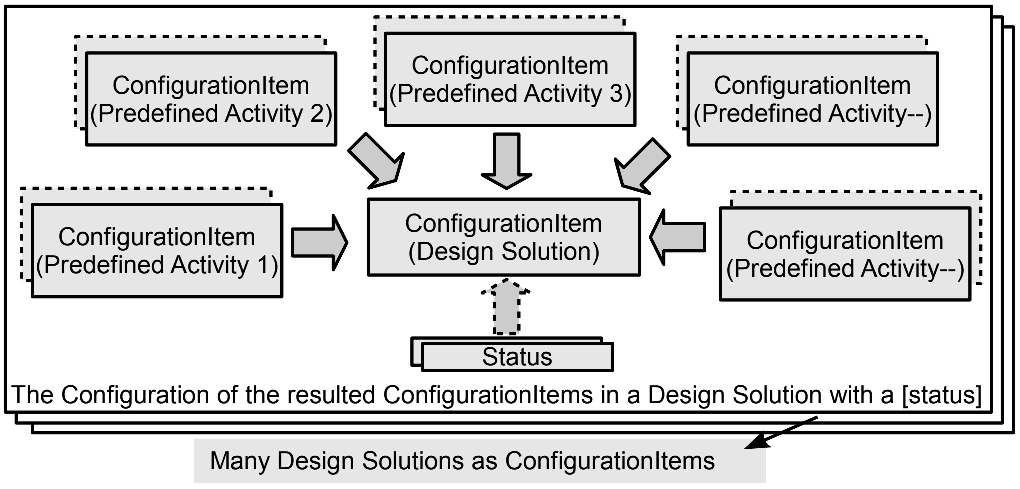

ConfigurationItem (Predefined Activity 2)
ConfigurationItem (Predefined Activity 3)
ConfigurationItem (Predefined Activity--)
ConfigurationItem (Predefined Activity 1)
ConfigurationItem (Design Solution)
ConfigurationItem (Predefined Activity--)
Status
The Configuration of the resulted ConfigurationItems in a Design Solution with a [status]
Many Design Solutions as ConfigurationItems

Figure 5.22.: Design solutions as *ConfigurationItems*

conflict to each other. Reaching a compromise solution of the design is the main objective of the design process. Regarding the developed approach and as it is depicted in Figure 5.22, a design solution at a certain stage can be regarded as a combination of several *ConfigurationItem*(s), which in turn represent a combination of the tasks-details. As a result, a design solution is represented in a *ConfigurationItem* by making use of the capabilities offered by the configuration management model (see section 4.5.3).
Considering the fact that each design task will be performed several times in the iterative design process and accordingly a variety of *ConfigurationItem*(s) of the same task will exist, design solutions can be conducted from the combination of different tasks-*ConfigurationItem*(s). These design solutions, depending on the objectives of the designer, can be treated as versions and saved dependently to each other with the clarification of the 'delta' between each two of them by means of a *Versioning* concept (see 5.2.2.3). The definition of design solutions, which is

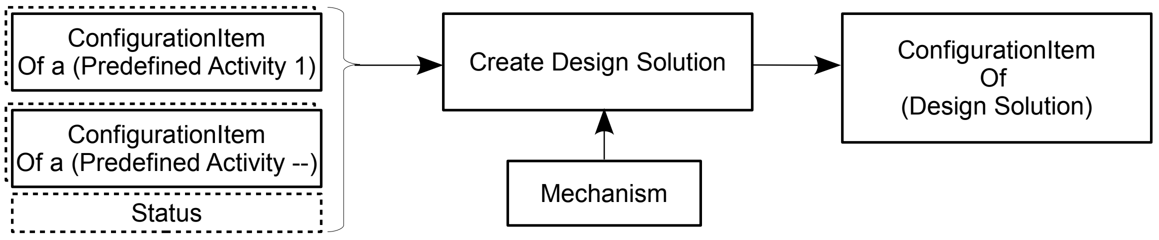

ConfigurationItem Of a (Predefined Activity 1)
ConfigurationItem Of a (Predefined Activity --)
Status
Create Design Solution
Mechanism
ConfigurationItem Of (Design Solution)

Figure 5.23.: Activity: *Create Design Solution*

shown in Figure 5.23 can be described while considering the predefined activity concept (see Table A.16) as follows: a mechanism, which is capable of checking and accordingly of retrieving a number of available *ConfigurationItem*(s), is applied. The next step is to define a `Design Solution`-configuration from the specified *ConfigurationItem*(s) with/without a specific status.

5.2.2.2. Update Status

Depending on the modeling approach which terminates each early design predefined activity by defining a configuration as a *ConfigurationItem*-object, the predefined activity depicted in

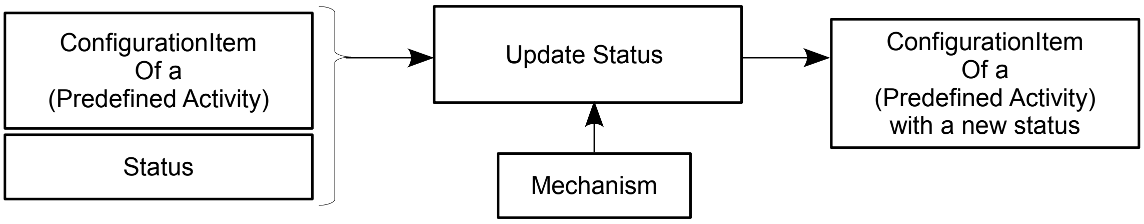

ConfigurationItem Of a (Predefined Activity)
Status
Update Status
Mechanism
ConfigurationItem Of a (Predefined Activity) with a new status

Figure 5.24.: Activity: *Update Status*

Figure 5.24 (see also its partitions in Table A.17) is introduced to enable designer to assign a status to a configuration if regarded necessary. As it is shown in the figure below, the purpose of this activity is confined to update the status (the default status is 'Unspecified') of a selected *ConfigurationItem* and to output/store it with the assigned one.

5.2.2.3. Create Version

Taking into consideration the nature of the early ship design process on one hand and the requests to save the history of specific design tasks in an efficient manner on the other hand, a *Create Version*-activity is modelled. It is capable to reflect this function with help the versions management model. The *Create Version* shown in Figure 5.25 is modelled as predefined activity (see Table A.18). As it is depicted in Figure 5.25, a *ConfigurationItem*, which represents a

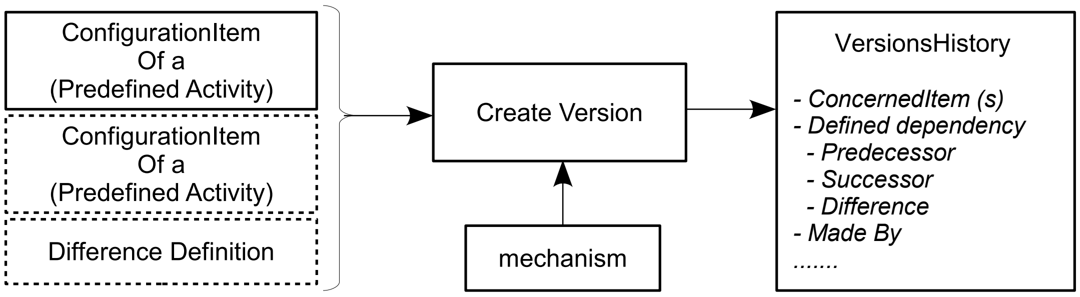


Figure 5.25.: Activity: *Create Version*

combination of a *PredefinedActivity*-details (a combination of the interacted information objects, which play input, control, mechanism and output roles), can be defined as a version. It can be used as an input in the *Create Version* to be added to the *VersionsHistory* of the addressed early design predefined activity. Moreover, if several *ConfigurationItem*(s) for the same predefined activity exist (due to the repetition of the same task), these *ConfigurationItem*(s), according to the designer-objective, can be handled as versions and saved within the *VersionsHistory* depending on each other by defining the difference between each two of them.

5.2.2.4. Control Changes

In order to reflect a change management concept, as it has been described in the information model (see section 4.5.4), on the process model level, three predefined activities are modelled. They are based on the change management model 4.5.4 and provide the following capabilities: *Check Change Influences*, *Request Change* and *Approve/Reject Requested Change*

5.2.2.4.1. Check Change Influences

The purpose of this predefined activity, which is depicted in Figure 5.26 (see also the partitions in Table A.19), is to check the influences associated with a change of a *ChangeableItem*. The applied mechanism (which will be described in the chapter 7), should be able to evaluate the effects of the change of a specific *ChangeableItem* at the information model as well as at the process model levels in accordance with the *ChangeEvaluation*-object (see section 4.5.4)

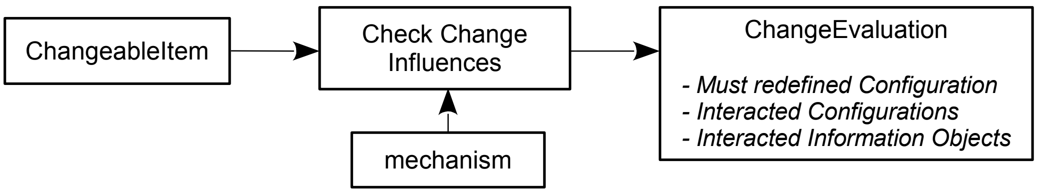

ChangeableItem
Check Change Influences
mechanism
ChangeEvaluation
- Must redefined Configuration
- Interacted Configurations
- Interacted Information Objects

Figure 5.26.: Activity: *Check Change Influences*

5.2.2.4.2. Request Change

This predefined activity, which is depicted in Figure 5.27 (see also the partitions in Table A.20), considers the fact that the checking of the influences of a change can be followed by the needs to request a change. The mechanism, which will be applied to perform this activity, should enable the justification of the requested change in addition to the outputting of this request depending on the *ChangeRequest*-object (see section 4.5.4).

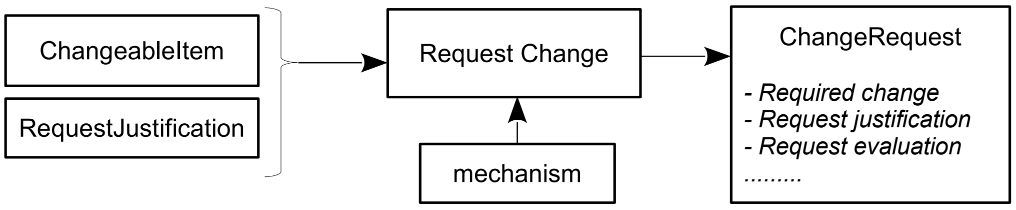


Figure 5.27.: Activity: *Request Change*

5.2.2.4.3. Approve/Reject Requested Change

The purpose of this predefined activity shown in Figure 5.28 (see also the partitions in Table A.21) is to control a requested change.

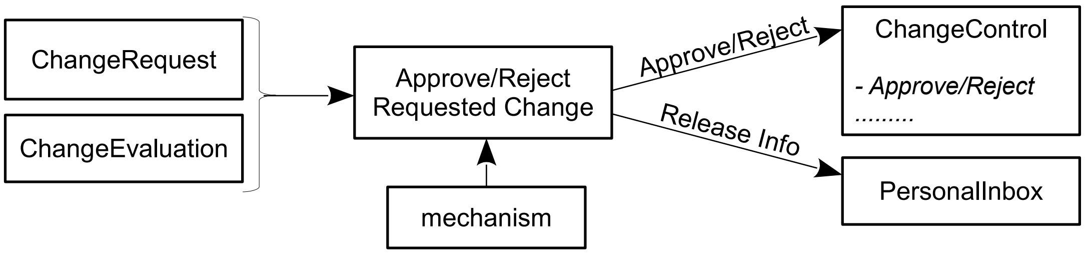


Figure 5.28.: Activity: *Approve/Reject Requested Change*

The applied mechanism, should at first provide the ability to reflect the decision made regarding a specific requested change (approved or rejected) by taking into consideration the *ChangeControl*-object (see section 4.5.4). Secondly, to output the information gained from this decision to the

PersonalInbox(es) of the persons/Organizations (see section 4.4.2), who are concerned with these information.

5.2.2.5. Create Report

In order to satisfy reporting needs within the early design process, the reporting predefined activity is modelled. It is based on the previously described report model (see section 4.4.4). This predefined activity (see its partitions in Table A.22) is depicted in Figure 5.29. As it is shown in the figure below, the input of this activity is any of the *ReportableItem*(s), which can be *DesignProject*, early design task represented as *PredefinedActivity* or *ConfigurationItem* (see 4.4.4). The mechanism, which will be applied to perform the reporting, should be capable of creating considerable reports from this input. The output is represented by the corresponding report with respect to the properties of the *Report*-object (see 4.4.4).

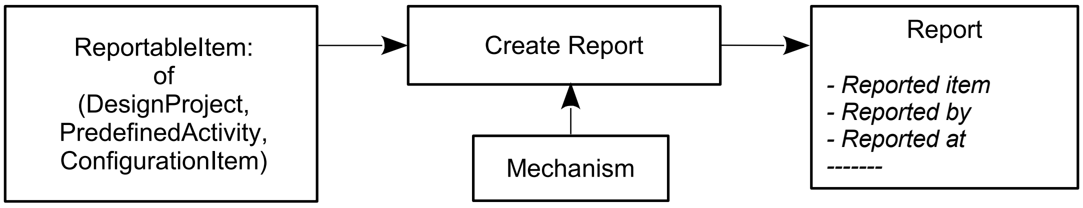


Figure 5.29.: Activity: *Create Report*

6. Ship Early Design Enterprise Model

The developed ship information model (see chapter 4) and ship early design process model (see chapter 5), which are interacting with each other, reflect the principle concept of an enterprise model, which integrates two different aspects: data and process as it is shown in Figure 3.1. The ship enterprise model depicted in Figure 6.1, represents an integration of the two above outlined models. As it will be described in the following sections, this can lead to significant benefits regarding the design process and especially in the early phase. The enhancement of this design phase comes from the overcoming of many of its current drawbacks as mentioned in section 2.2.

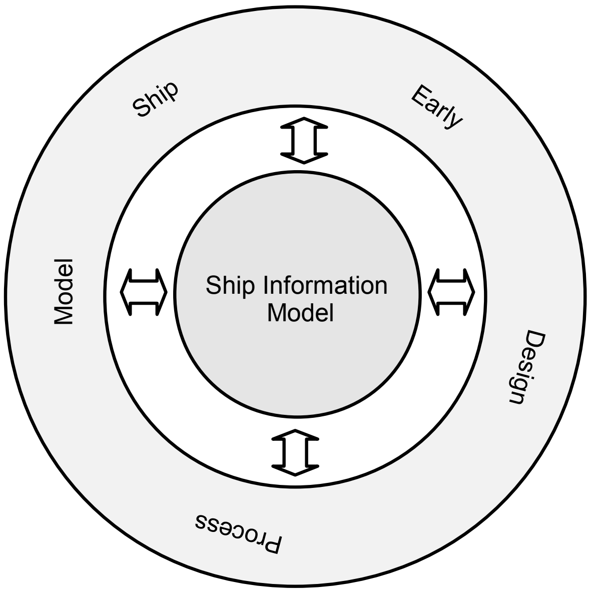


Figure 6.1.: Ship Enterprise Model

6.1. Decision Making

One of the advantages which can be attributed to the developed models is the supporting of the decision-making aspect. This is due to the special importance of this aspect regarding the ship early design phase. Making decisions can be fostered at many stations during the design life cycle regardless of the person who makes the decision. In general, the supporting of the decision-making aspect regarding the enterprise modeling approach can be described as follows:

- Increasing the details associated with the information objects within the ship information model which are used during the performing of the early design tasks, supports the partial decisions made by the designers during the performing of these tasks.
- The employment of the configuration management aspect and accordingly the decision-dimension represented by the 'status'-aspect within the predefined activity concept supports the decision-making.

- Trade-off studies, which can be subjected to the design solutions in order to make decisions regarding the appropriate solution, can be enhanced by providing the precise difference 'delta' between them. The addressing of the design solutions as configurations can be versioned, offers the ability to identify the exact difference between two solutions by using the features supplied by the version management model.
- The decisions, which can be made within the early design stage and which relate to modification issues, represent another decision-making area which can be fostered. The increasing of the details and the detection of the consequences and implications, which can be associated with the potential changes, supports the responsible person(s).
- When considering the rights management model, each of the decisions, which can be made within a design project, is documented and linked to a specific person. This helps to support the history of decision-making concept and not just the decisions themselves.
- The increment of knowledge about a ship by using the enterprise model advantages leads to the ability to make many decisions earlier regarding the ship design process. Given this fact, it can be stated that the decision-making concept within the overall design process is supported (see Figure 2.5).
- As it has been mentioned already (see section 3.2.6), providing the design team with the right data, which they are concerned with, at the right time, supports the collaborative management concept and accordingly the design team by making the decisions.

6.2. Data-Reuse

Considering the important features which can be gained from supporting a data-reuse concept, a special importance is accorded to it by the development of the ship enterprise model. How the data-reuse concept is supported can be seen in two levels. The first is the enhancement of the reuse aspect within the addressed early design phase, and the second is the fostering of the data-reuse capabilities with the downstream design stages. This can be attributed to:

- In general, the foundations provided by the developed models serve the core concept of the data-reuse. The ship information model and subsequently the integrated database offers the ability to deal with data as objects, that can be a subject to many operations. Thus, the capability to retrieve the data in order to be reused is insured.
- Finishing the early design tasks with the definition of *ConfigurationItems*, provides the ability to get an insight into the whole associated information objects as well as the assigned status. This helps the designers to efficiently reuse the data from a design project to other. It can be reused as 'blocks' represented by configurations and not just as isolated objects.
- Providing the capability to deal with the data at the early design stage as it is represented in the following stages supports the data-reuse between design phases. For example, the ability to deal with the 3D models at the early stage as it is in the basic design supports the ability to reuse these models later. It is important to mention that this does not address the creation of the 3D models, which is an implementation aspect, but rather it addresses the treatment of these models as they are represented in order to be reused later.
- Considering the fact that the data are usually reused as they are saved in their final state, without taking into account their history, i.e. the changes which may subjected to these data until they are in their final state. Thus, providing the version history of these data as well as differences between them, as insured within the developed model, will definitely lead to enhance the design team to get an insight into the expected implications of the reusing specific data.

6.3. Data-Exchange

In principle, providing a neutral information model which can be a basis for a neutral integrated database, supports the concept of data-exchange. The database can then be seen as a neutral storage as well as an intermediate layer to exchange data between different applications being applied in the early design phase. Furthermore, it is the basis to transform data to the subsequent design stages. This raises the need to provide data in forms understood or able to be used in the following stages, which is an implementation as well as modeling issue as it has been discussed above, see section 6.2. On the other hand, the developed models together with the developed applications as it will be seen in chapter 7 can help to foster the exchange of data between different participants during the early design stage. It is supported by providing the results of the early design tasks, which have to be exchanged in common formats independent on any CAD-Systems such as IGES, VRML and even PDF-format with all its advantages. The data represented by these formats can be subjected to many functionalities such as save, invoke, etc. Moreover, many formats (representations) can be used to describe the same data, therein the consistency-aspect is insured as it will be seen later.

6.4. Support Design Life-cycle

The ship design life-cycle can be seen as the general supported aspect regarding the developed enterprise model. Its iterative as well as integrated nature are taken into consideration to be supported through:

- Fostering the time dimension of the design life-cycle by means of predefined activity concept. The representation of the tasks in a predefined form helps to automate the design process to the greatest extent possible and accordingly to accelerate the performing of the tasks, which can be demanded to be iterated several times.
- Supporting the control-concept within the design life-cycle. This is achieved by making use of predefined activities for, *Create Design Solution*, *Update Status*, *Create Version* and *Control Changes*
- Supporting the design process by enhancing the clarification of the interactions between design tasks whether they are dependent or independent. This helps to form a clear image about the priorities regarding the order of the tasks-execution and accordingly the efficiency of the design life-cycle is enhanced.
- Increasing the data-details at the early design stage and the collaborative teamwork concept helps to support the design life-cycle by shifting many of the decisions made during the design process to the early phase.
- Supporting the design life-cycle regarding the integration and data reuse aspects. This is counted to the developed ship information model, which offers the basis for an integrated database and provides the foundation for an efficient integration between data and process.

7. Implementation

In order to realize the enterprise modeling approach for the ship in early design phase described in the previous chapters, a system, which is named as 'Ship Early Design System' and based on the developed models, is implemented. The previously introduced predefined activities are implemented in many UseCases. The implemented system and UseCases are integrated with each other in order to achieve the ship early design studies in an efficient manner.

7.1. Database Setup

In order to achieve the implementation of the ship early design system shown in Figure 7.1, a database is set up. The ship information model, which is modelled by the 'Visual Paradigm'-UML tool [21], is mapped within the database in order to transform it from the physical-model level to information processing system level and accordingly to make use of its properties. This translation is achieved by using SQLite [19] as a Relational Database Management System (RDBMS), Python [15] as Object Oriented Programming (OOP) language and SQLAlchemy [18] as Object Relational Mapping (ORM)-technique. The procedures to configure (map) the information

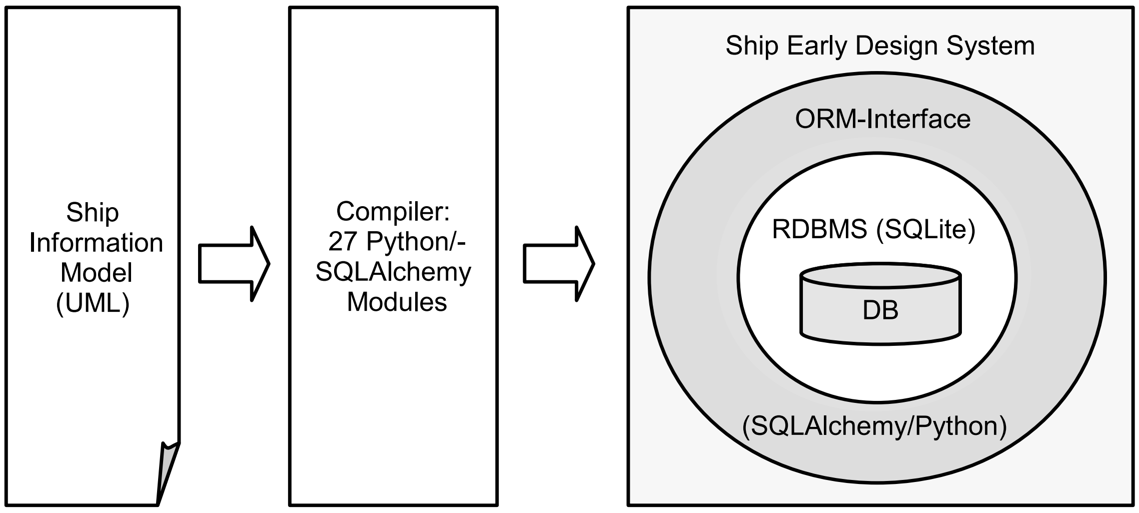


Figure 7.1.: Database Setup

model within the central database (data management component), which is referred to as compiler in Figure 7.1, are realized manually. Regarding the applied modeling approach, each group of information objects (package) is translated to a python-module within which the objects with all their associations are defined regarding the used SQLAlchemy-specifications. Thereby, the fact that the python-language supports a limited form of multiple inheritance concept is taken into consideration. By mapping the information objects to the central database, the attributes of these objects are programmed to be 'Optional'. This offers the ability to exploit these objects with minimum data level and complies with the fact of low level data at the early design phase.

As soon as the information model is configured within the central database, which means that the objects (classes) with their relationships are mapped to the central database, the data can be saved to these mapped objects in instances-form. Many instances can be saved for the same class, these class-instances are identified by IDs created automatically by the system.

7.2. Predefined activities in UseCases

In order to realize the early design tasks represented in predefined activities as they are described in chapter 5, they are implemented in form of UseCases. The implemented UseCases are modelled by means of the 'activity-diagram' from the UML-method (see section 3.1.3). By taking into consideration the applied modeling approach depicted in section 5.1, the UseCases are implemented in a way which insures the integration with the built ship early design system and accordingly the interaction with the configured central database as it is shown in Figure 7.2. The partitions of these activities, which reflect the input, control, mechanism and output aspects regarding the ICOM-concept (see section 5.1), are the procedures (within the UseCases), which have to be followed in order to perform the represented early design tasks. In order to reflect the mechanism-aspect which represents the applied tools within the UseCases, interfaces to several tools are implemented. Within the implementation of the UseCases, the following considerations are taken into account:

- Each UseCase is controlled by a person. This person is in fact the one who is responsible to perform the predefined activity, which is implemented in the UseCase.
- The interaction between the early design tasks, whether they are dependent or not (see section 5.1.3), is taken into account. The user is informed automatically, in accordance to the available data within the central database at the time of accessing the UseCase, whether the execution of the represented early design task is possible or not.
- The concurrency-concept within the developed system is limited due to the capabilities provided by the used SQLite-DBMS. SQLite supports a transaction concept of one writer at any instant of time but it allows unlimited number of simultaneous readers. As a result, when two early design tasks are performed simultaneously, it is not possible to save (commit) the results of the both tasks at the same moment in the central database.
- The results of the executed tasks are available as soon as they are saved within the central database. The results do not exist until they are given an ID within the central database.
- During the execution of the early design tasks, the date/time-data in addition to the data of the user, who performs the task, are derived and associated automatically to the corresponding mapped information objects.
- The implementation realizes the integration strategy described in section 2.3.1, which means, that each task can be performed at each time. The purpose of the system in this case is to react depending on the availability of data. Moreover, the system is capable to inform automatically the responsible users about the available data, which they are concerned with as soon as the data are to be found within the central database. For example, when a new ship form is created and saved within the central database, the system will automatically inform the users, who are responsible to perform design tasks interacted with the 'Generate Hull Form'-task, such as the user who is responsible to perform the 'Create Compartmentation'-task. The system is characterized with the possibility to address the right users at the right time to have them provide the right data. Informing the right users (not all users) increases the efficiency of the design process by avoiding the information overflow. Furthermore, it helps to decrease the complexity by means of the reducing of the potential changes within the design process.

7.3. Ship Early Design System Integration

The developed architecture of the ship early design system including the described UseCases takes into account that the main objective of the system is not to replace the existing ship design tools, but rather to make use of the functionalities offered by these tools to perform the design tasks and manage them in order to achieve the early design phase as efficiently as possible. The integration of the ship early design system with the implemented UseCases is shown in Figure 7.2. These UseCases, in this context, represent the design workbench within a shipyard. Regarding the tools on which they are based to perform the predefined activities, the UseCases can be categorized into two categories as it is shown in Figure 7.2:

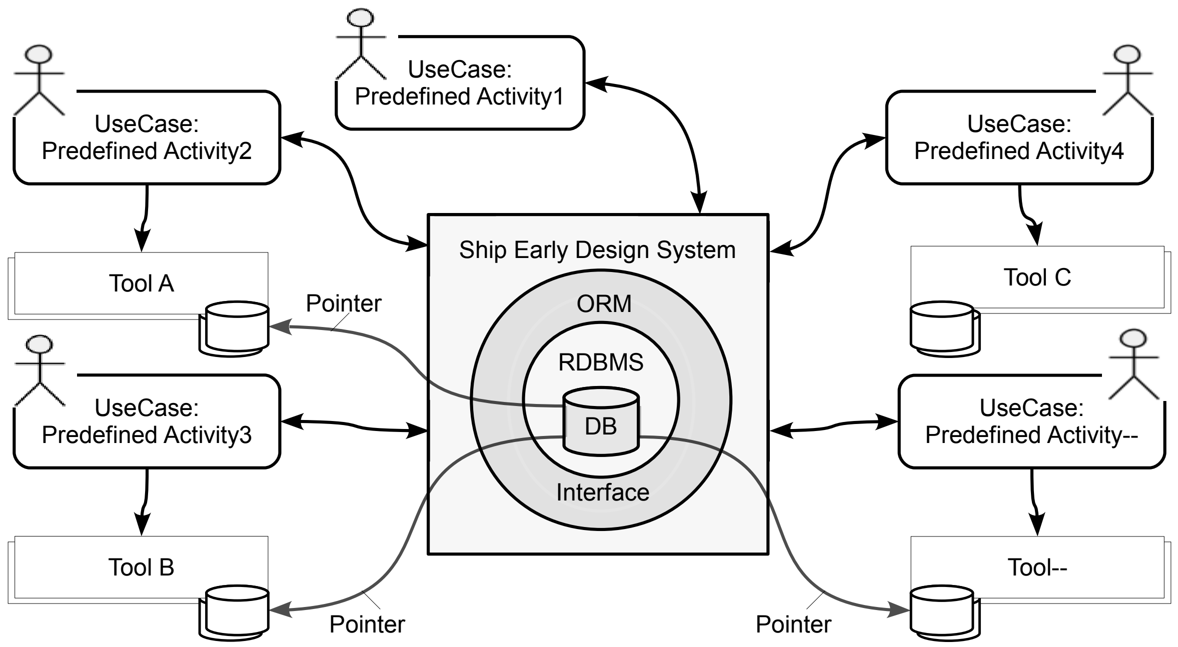


Figure 7.2.: Ship Early Design System Integrated in Shipyard Design Workbench

- UseCases interact with the configured database through the developed system and apply one or more external tools to perform the represented predefined activity such as the UseCases: *Predefined Activity-2,-3,-4* depicted in Figure 7.2. The definition/analysis predefined activities described in section 5.2.1 like *Generate Hull Form*, *Create Compartmentation*, etc. are implemented by means of this category of UseCases. Later on, the term 'Definition/Analysis UseCases' is used to refer to these UseCases.
- UseCases interact with the central database through the developed system and do not apply any external tools. An example of that is the UseCase: *Predefined Activity1* in the upper figure. In fact, this category of UseCases can represent any of the management/reporting predefined activities described in section 5.2.2 such as *Create Version*, *Create Design Solution*, etc. Later on, the term 'Management/Reporting UseCases' is used to refer to these UseCases.

The tools, which are applied by the first category of the implemented UseCases, have their own data management components. Therefore, and in order to increase the efficiency of applying these tools, the developed system offers an optional ability to maintain an indicator (pointer) into the external data management. In fact it is a pointer saved within the central database with the corresponding information object and it refers to the results of the executed task within the applied data management component. As an example is the pointer saved within the central database and refers to the task-results within the data management component of Tool A applied by the UseCase: *Predefined Activity2* (see Figure 7.2). Considering the addressing nature of the ship early design task-results within this dissertation, which enables the designer to save the key-results explicitly (see section 4.3.11), the main purpose of the above mentioned pointer is to enable a designer (when it is needed) to enrich the explicitly saved task-results by offering an indicator to their details within the applied tools.

7.4. Implemented UseCases

The implemented UseCases are shown in Figure 7.3. In the following sections the both categories of the UseCases introduced above are described. After logging-in and selecting the design project, the responsible user should select the corresponding UseCase in order to execute a specific early design predefined activity within the addressed design project. In order to reflect the rights management concept at the implementation level, an examination (checking) of the user responsibility is executed automatically by the system. This is followed by permitting or rejecting the access to the UseCase.

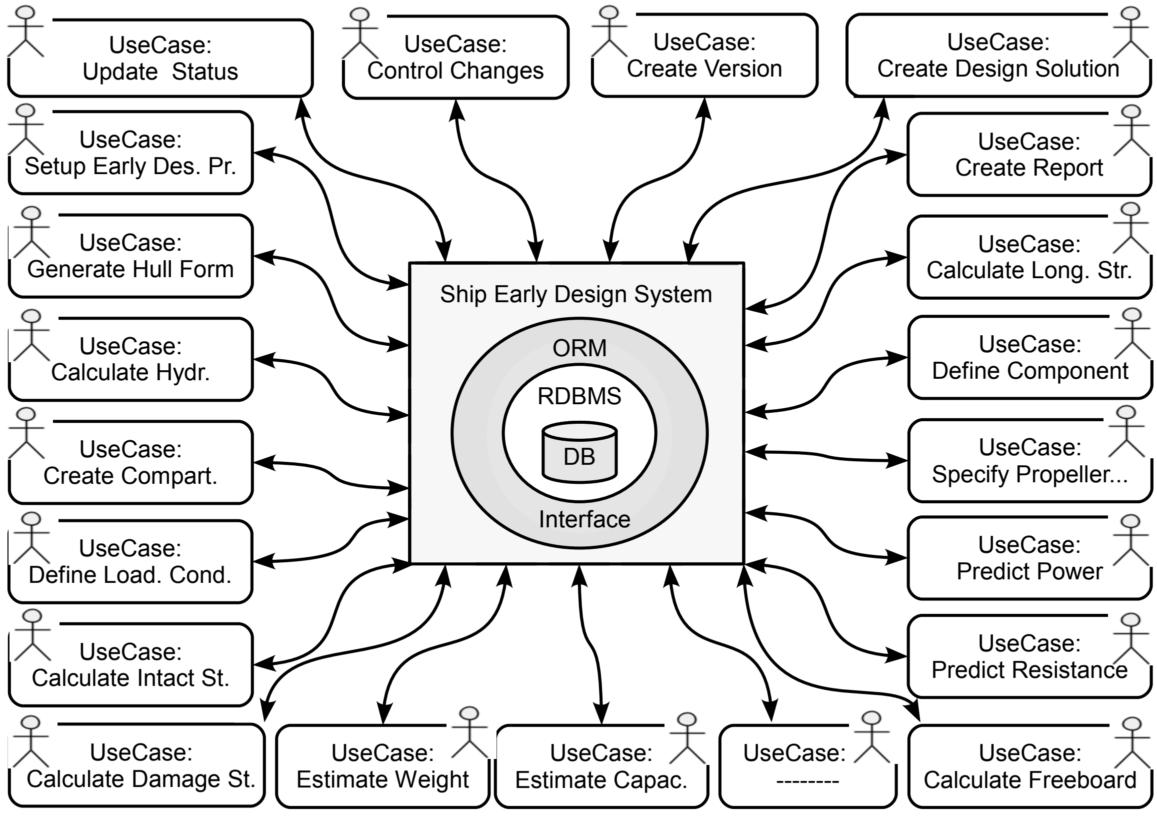

UseCase: Update Status
UseCase: Control Changes
UseCase: Create Version
UseCase: Create Design Solution
UseCase: Setup Early Des. Pr.
UseCase: Create Report
UseCase: Generate Hull Form
UseCase: Calculate Long. Str.
UseCase: Calculate Hydr.
UseCase: Define Component
UseCase: Create Compart.
UseCase: Specify Propeller...
UseCase: Define Load. Cond.
UseCase: Predict Power
UseCase: Calculate Intact St.
UseCase: Predict Resistance
UseCase: Calculate Damage St.
UseCase: Estimate Weight
UseCase: Estimate Capac.
UseCase: --------
UseCase: Calculate Freeboard
Ship Early Design System
ORM
RDBMS
DB
Interface

Figure 7.3.: Implemented Set of UseCases

7.4.1. Definition/Analysis UseCases

Some of these UseCases are described in the following sections, the remaining are depicted in section A.4. The tools, which are applied within the implemented UseCases, are used to reflect the applicable functions. In order to consider the software-specifications of ship companies, these tools, which are just examples, can be replaced by other tools, or even new UseCases can be implemented and integrated with the built system. Taking into account the fact of the multiplicity of the methodologies, which can be applied to perform the early design tasks, some of the following UseCases are programmed to provide the ability to conduct the tasks in different scenarios.

7.4.1.1. Setup Early Design Project

According to the developed approach, the first step, which has to be performed in order to initiate design tasks, is the definition of a new design project. It represents the general framework in which the design goals are to be achieved. This UseCase, which is shown in Figure 7.4, offers the possibility to setup new design projects. It is in fact the implementation of the predefined activity described previously in section 5.2.1.1. As it is shown in Figure 7.4, it includes seven activity partitions represented by means of the 'Activity diagram' from UML-method. These are, *apply setup mechanism*, *specify design project information*, etc. According to the implementation of this UseCase, these partitions must be achieved completely in order to setup a new design project. By the implementation, specifying some data are mandatory to setup a new design project. The mandatory data are: project name, L_{pp}, B, T, D, CB, ship type, speed range, endurance, cargo type and cargo capacity. Additionally, some optional supplementary information can be assigned in order to increase the details of the created design project, such as component specifications (see section 4.3.2), owner designation (see section 4.3.3), etc. During the implementation the right of setting up a new design project is given to the shipyard manager, only the user, who holds the role 'ShipyardManager', is able to set up a new design project. To sum up, the setting up of a new design project can be characterized by the following:

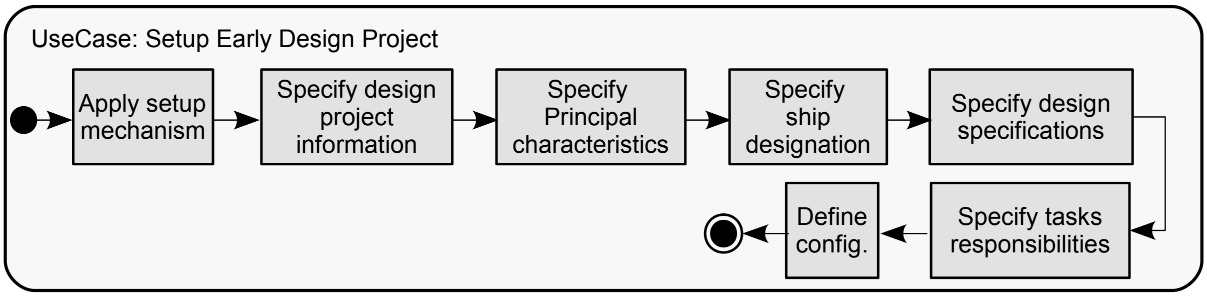

UseCase: Setup Early Design Project
Apply setup mechanism
Specify design project information
Specify Principal characteristics
Specify ship designation
Specify design specifications
Specify tasks responsibilities
Define config.

Figure 7.4.: UseCase: Setup Early Design Project

- The established design project is identified by a unique ID, which is created automatically by the DBMS as soon as the activity partitions (represented in Figure 7.4) are completely performed and the transaction is committed.
- Multiple design projects can be initiated at the same time. The independency as well as the possibility to exchange data between them are thereby insured.
- Based on the developed models, the predefined activities, which represent the ship early design tasks, can be associated with the design project as soon as it is established.
- A 'ProjectManager' as well as any number of 'Designers' (who are saved within the central database as persons and may relate to different organizations) can be assigned to a design project. Each of them has a specific task which is the responsibility of one or more early design predefined activities
- The successful performing of this task is followed by an automatic informing of the persons,

who are authorized to perform the early design predefined activities, by means of messages to their *PersonalInbox*(es) about the related design project and their responsibilities.

7.4.1.2. Generate Hull Form

The predefined activity *Generate Hull Form* is implemented within the UseCase depicted in Figure 7.5. After accessing this UseCase by the responsible user, the principal characteristics as well as ship designation in addition to the hull form specifications (if there is any specification), which are saved in the central database while setting up the design project, are invoked automatically and shown to the user in order to be taken into consideration during the hull form definition. In the next step, the user has to select one of the mechanisms offered to be applied in order to perform the hull-form generation. In this case three mechanisms represent three scenarios to

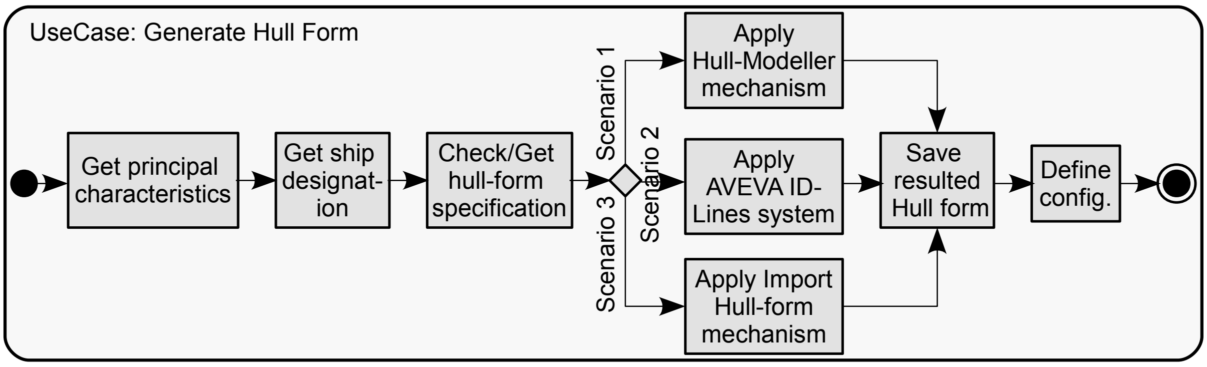

UseCase: Generate Hull Form
Get principal characteristics
Get ship designat-ion
Check/Get hull-form specification
Scenario 1
Scenario 2
Scenario 3
Apply Hull-Modeller mechanism
Apply AVEVA ID-Lines system
Apply Import Hull-form mechanism
Save resulted Hull form
Define config.

Figure 7.5.: UseCase: Generate Hull Form

define the hull-form are provided. This gives an example of the diversity of the techniques which can be used to define a hull form. It is taken into consideration that the generated hull form should, on the one hand meet the principal characteristics (which are specified in the previous predefined activity) and on the other hand should be able to be used in downstream design tasks. Available scenarios are:

Hull-Modeller mechanism: a tool which is able to create the desired hull-form based on ship hull-form library, such as the one described in the following section 7.4.1.2.1.

AVEVA ID-Lines system: In order to offer the possibility to create a hull-form from scratch by means of a geometry-engine, an interface to the lines component of the CAD-system AVEVA Initial Design [108] is implemented. AVEVA ID-Lines is invoked and started automatically as soon as this scenario is selected by the responsible designer.

Import Hull-form mechanism: by selecting this scenario the hull-form is available and has to be added (imported) to the central database.

The ship hull form from the applied mechanism is saved to the central database and accordingly it can be obtained and used in the following design tasks. This is conducted by using the corresponding information objects which offer the ability to save several details about the hull-form (see section 4.3.5) as soon as they are available. Moreover, a definition of an indicator to the store of the created hull-form within the used tool is maintained. The last partition (for configuration definition) is performed automatically. The used instances of the information objects: *PrincipalCharacteristics*, *ShipDesignation*, *Tool*, *ShipForm* with/without the hull-*Specification* are grouped together and saved in the central database as an instance of the information object *ConfigurationItem* with the name of `Ship Form`.

7.4.1.2.1. Hull-Modeller mechanism

A mechanism for the automated creation of a specific ship hull-form based on existing ones is provided. The general approach is shown in Figure 7.6. A library of hull-forms represented in IGES-format [91] and organized according to the ship type are available. One of the available hull-forms is selected in order to be used as a basis to create the project specific one. The selection is made according to the required ship type, which is identified by the user. The IGES-parser is able to read the information of the selected hull-form represented in an IGES-format. The hull form is then modified by means of Hull-form transformation algorithm to meet the specific hull-form parameters. An IGES-creator function is then applied to define the information of the modified hull-form once again according to the IGES-format. The modifications, which are implemented, can be categorized in functions:

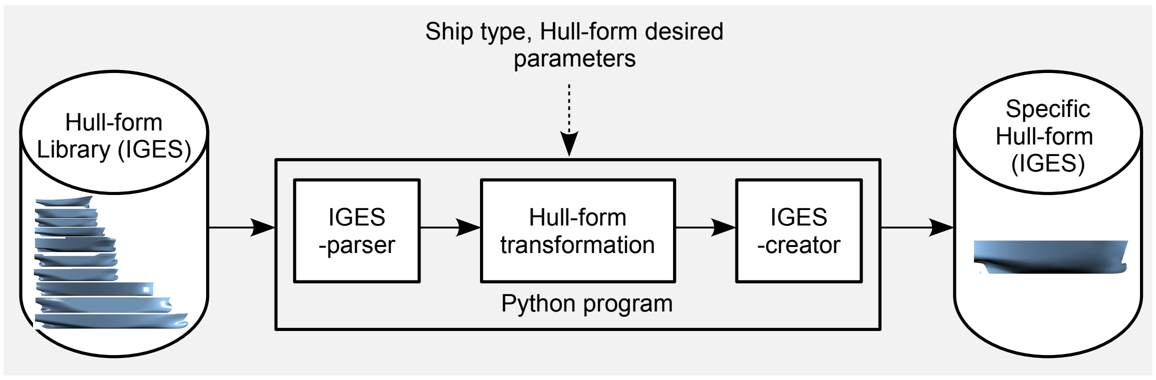


Figure 7.6.: Hull-modeller mechanism

- General modifications: simple functions to make general modifications to the hull-form. They are comprised of modification of the hull-form-length to get the desired L$_{pp}$, modification of the hull-form-breadth to get the desired B and modification of the hull-form-depth to get the desired D.
- General modifications by taking the local regions into consideration: the previous (general) modifications can be performed while preserving local properties, like preserving the bilge radius and the parameters of the bulbous bow.
- Local modifications: functions based on the Lackenby-Transformation [100] are implemented in order to enable the modifications of local regions of the ship hull-form. They include functions for the lengthening of the parallel mid-body, modification of bilge radius in addition to the modification of parameters of the bulbous bow like length and breadth.

In conclusion, this mechanism to provide a ship hull form has advantages such as simplicity as well as time-saving. However, only minor modifications are possible while generating a hull form which can serve as a start of the design process. It is assumed, that this form will later be replaced by a more elaborated one defined with a specific form modeling tool.

7.4.1.3. Create Compartmentation

This UseCase enables the user to perform the *Create Compartmentation* task as it is described above in a predefined form, see section 5.2.1.4. The implemented UseCase is depicted in Figure 7.7. After accessing this UseCase by the responsible user, the availability of the ship hull form(s) is checked automatically. The absence of a ship hull form within the central database related to the addressed design project, signals an error message and the compartmentation process will be terminated. Otherwise, the user is asked to select one of the existing ship forms. After

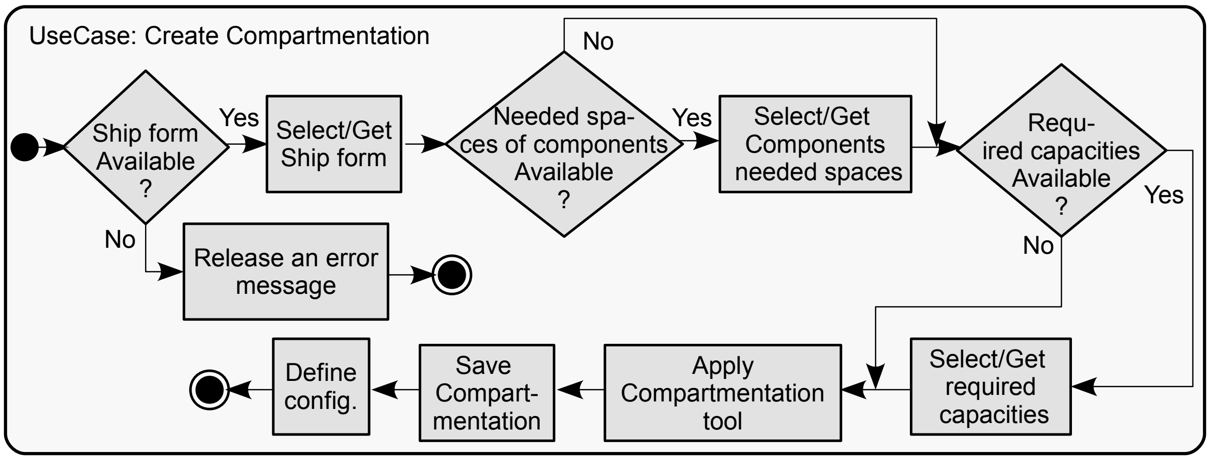

UseCase: Create Compartmentation
Ship form Available ?
Yes
No
Select/Get Ship form
Release an error message
Needed spaces of components Available ?
No
Yes
Select/Get Components needed spaces
Required capacities Available ?
Yes
No
Select/Get required capacities
Apply Compartmentation tool
Save Compartmentation
Define config.

Figure 7.7.: UseCase: Create Compartmentation

the selection, an automated search for the existence of requirements regarding the capacities as well as ship components is executed. This is followed by a selection procedure (in the case of availability) or by an ignoring procedure (in the case of absence). After that, a compartmentation mechanism is applied (such as the one developed and described in section 7.4.1.3.1). The last procedures are to save the resulted compartmentation within the central database by using the corresponding information objects in addition to the definition of the `Compartmentation` configuration. It should be mentioned that these two last steps are implemented to be executed automatically in case of utilizing the developed compartmentation tool described in the following.

7.4.1.3.1. Compartmentation

The developed approach, which has been published in the Proceedings of PRADS-Conference (see A.4), proceeds from the needs to automate and accelerate the process of ship compartmentation at the early design stage. It depends on the general definition strategy of the compartmentation-creation. The spaces of a ship, according to the general definition strategy, are generated from subdividing the ship-form into regions. This subdivision process can be successfully implemented by using horizontal and vertical subdivision elements, such as decks, longitudinal and transverse bulkheads. These subdivision-elements are then used as boundaries to define ship spaces. This strategy is time-consuming when it is manually executed due to the large number of elements that should be defined in order to create compartmentation. Therefore, the developed tool which is shown in Figure 7.8 is implemented to be executed in semi automated procedures as it is described in the following.

Macro Language for Compartmentation-Definition

In a first step, a special syntax is developed which can be considered as a macro language used to efficiently describe the compartmentation of a ship's hull in the early design stage. The scripting of the compartmentation-definition is developed by taking into consideration the descriptions of the information-objects, which are related directly or indirectly to the *SpaceElement*-object. The development is accomplished by having several objectives in mind, such as, it offers a full topological approach, allows modifications efficiently, it is independent of any geometry engine, its re-use capabilities and it improves the space elements details at the early design stage. In the following, some definitions by using the special macro language are shown:

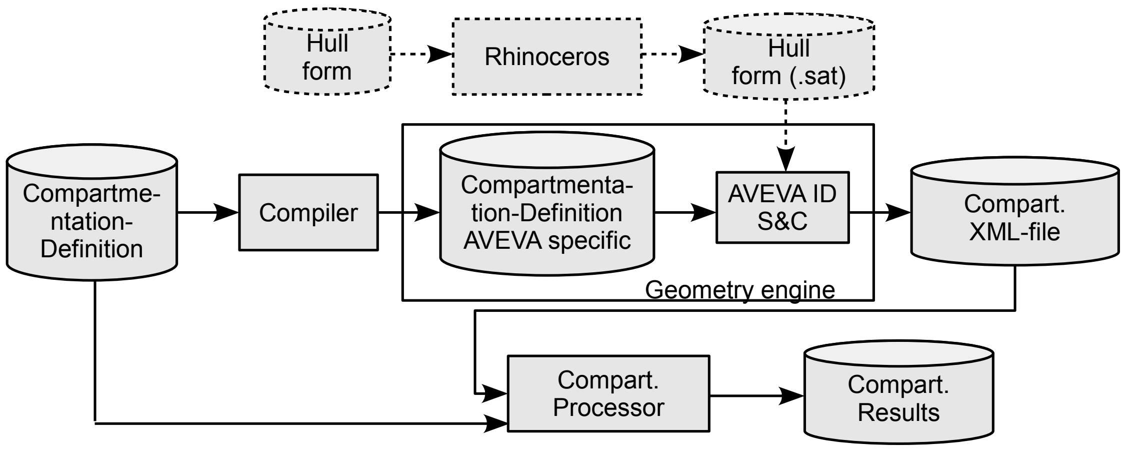


Figure 7.8.: Compartmentation Tool

Subdivision Element (SE): It can be a deck or a transverse-, longitudinal- bulkhead:

```
SE-NAME: location B1 B2 B3 B4 [shape-control];
```

The *location* identifies the placement of a SE, It is specified according to the placement-aspect described above (see section 4.4.1). The *B1 B2 B3 B4* specify the borders of a SE by taking into consideration the border-definition mentioned in section 4.3.6. These borders are dependent on the type of the SE. For example, in the case of a deck definition the borders are: *AFT FORWARD PORT STARBOARD*. The *[shape-control]* refers to the possibility to control and define complex shapes of SEs by using the sheer and camber concepts defined as follows:

```
SHEER(sheer-point-number):  position-number, change-value
CAMBER(camber-point-number):  sheer-point-number, position-number, change-
value
```

The shape of a SE can be controlled by a set of sheer points, using the *position-number* and *change-value*, which can be positive or negative. As an example, in case the SE is a deck, the *position-number* can be a frame number and the *change-value* is the change in Z-direction, which means that the sheer concept in this case will control the shape of the deck in (X-Z) plane (see the right sub-picture of Figure 7.9). A set of camber points can be used at each sheer point to control the shape of the SE in the orthogonal direction, for the deck example the camber concept will be used to control the shape in the (Y-Z) plane (see the left sub-picture of Figure 7.9).

Space Element: Space elements are differentiated with respect to the degree of complexity of their shapes. For simple spaces, which can be defined as a space formed by six boundaries represented by the previously defined SEs or the ship form:

```
SPACE-NAME: AFT FORWARD PORT STARBOARD LOWER UPPER CONTAINMENT-NAME;
```

Whereas the spaces, which have complex shapes can be defined in form of a Constructive Solid Geometry (CSG) approach. The space elements are defined by a binary relationship like 'addition' or 'subtraction'. As a result, the space element in this case will be defined as a volumetric space formed not by six boundaries but by any number of boundaries resulted from these binary relationships. The defined complex-spaces, in turn, can be used to define more complex spaces, which results in a CSG tree of any complexity. The definition of a complex space element is as follows:

```
SPACE-NAME: /ADD L(1..?)(SPACE-NAME) /SUB L(0..?)(SPACE-NAME) CONTAINMENT-
NAME;
```

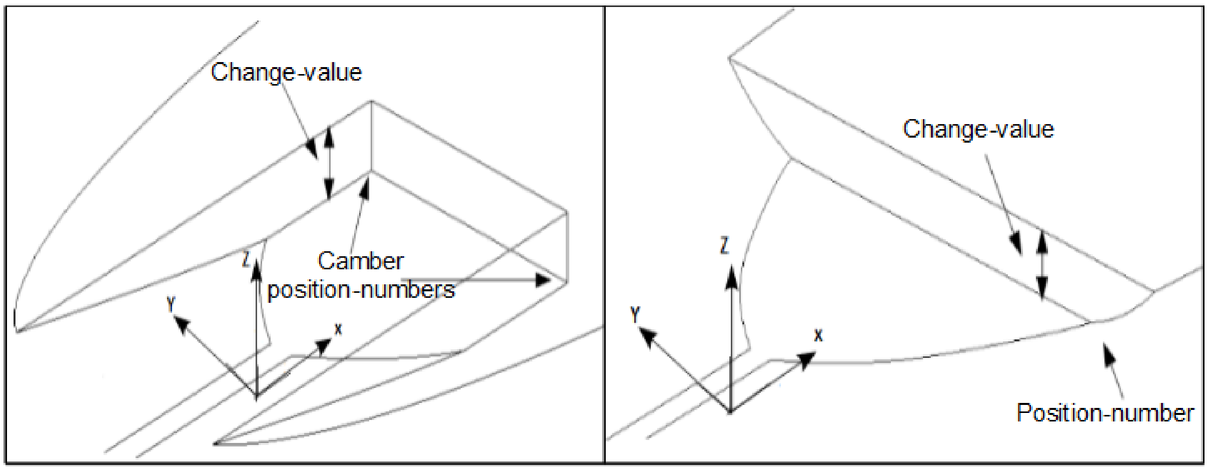


Figure 7.9.: Sheer and camber concepts

It should be noted that *SPACE-NAME* after */ADD* can be a list of one or more, and after */SUB* a list of zero or more space elements. In general, the function of a space element can be associated with the definition through the *SPACE-NAME* for example, 'Engine-Room'. The *CONTAINMENT-NAME* specifies the content included in a space element which is in accordance with the containment types predefined within the compartmentation-definition-script that corresponding to the previous descriptions in section 4.3.7.

Interface to Geometry Engine

The compiler shown in Figure 7.8 is written in python [15]. The function of this program is to interpret the information of the previously described definitions and to translate them into a language interpretable by a geometry engine, which in this case is the 'AVEVA-ID/S&C'-CAD system [108].

Compartment Definition Process

In Figure 7.10 the work-flow of the compartmentation within 'AVEVA-ID/S&C' is shown. The 'Compartmentation-Definition AVEVA specific' resulted from the compiler (see Figure 7.8), is built in a way that enables the automatic execution of the compartmentation process. The definitions, which are written in commands regarding AVEVA ID specific language, are executed consequently and finally export the results of the process as an XML-File. This file includes the compartmentation geometry and can be used as an input for many calculations (such as capacities) related to the ship-compartmentation. It should be mentioned that, when the hull-

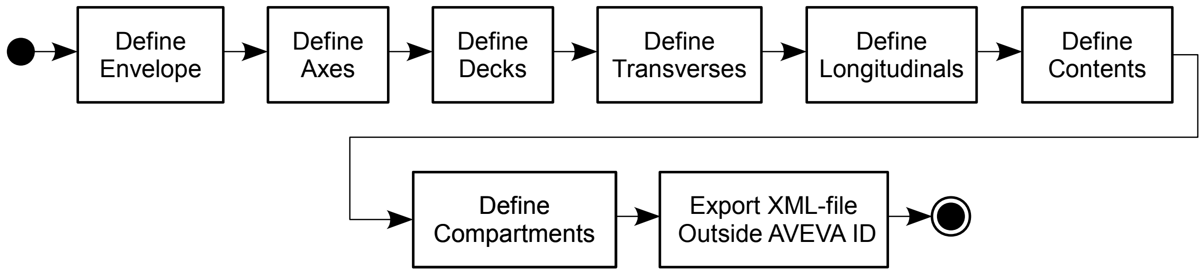


Figure 7.10.: Compartmentation process in AVEVA-ID (UML-Notation)

form that is to be used is not represented in SAT-format, a transformation to this format is required in order to take into consideration the specifications of the AVEVA-ID/S&C-Component

as the applied geometry-engine (see Figure 7.8). Thus, a connection with a tool, which enables this transformation, is implemented. Therefore, if required, the program establishes an automatic connection to the Rhinoceros-tool [16], which can be seen as an example of a tool with this capability.

Compartmentation Geometry Processing

The compartmention-processor shown in Figure 7.8 refers to the functionalities that are developed in order to get the results of the compartmentation depending on the resulted XML-file and available data. The results of the developed approach can be organized with regard to their information in two categories:

- Compartmentation-data: for each space, contains information about : name, content-type, volume, weight, longitudinal center of bouncy (LCB), transversal center of bouncy (TCB), vertical center of bouncy (VCB), surface area of boundaries, and for each space group like the HFO-tanks: content type, density, total volume, total weight, total LCB, total TCB, total VCB, total surface area.
- Compartmentation representation: it refers to the 3D-shape of compartmentation and accordingly for each space element. It is built by reading the necessary information from the XML-file, on which it is based to create a 'Virtual Reality Modeling Language' (VRML)-3D-model [33]. VRML is used due to its features which allow for building the model in form of a hierarchy and also allows adding properties to shape-parts.

7.4.1.4. Calculate Intact Stability

This UseCase is an example of several UseCases that are implemented based on 'AVEVA-ID'-CAD system [108] as a mechanism to perform the corresponding tasks. These UseCases are: *Calculate Damage Stability* (see Figure A.17), *Define Loading Condition* (see Figure A.22), *Estimate Weights* (see Figure A.21), *Calculate Longitudianl Strength* (see Figure A.19), *Calculate Freeboard* (see Figure A.16) and *Calculate Hydrostatics* (see Figure A.15). As depicted in Fig-

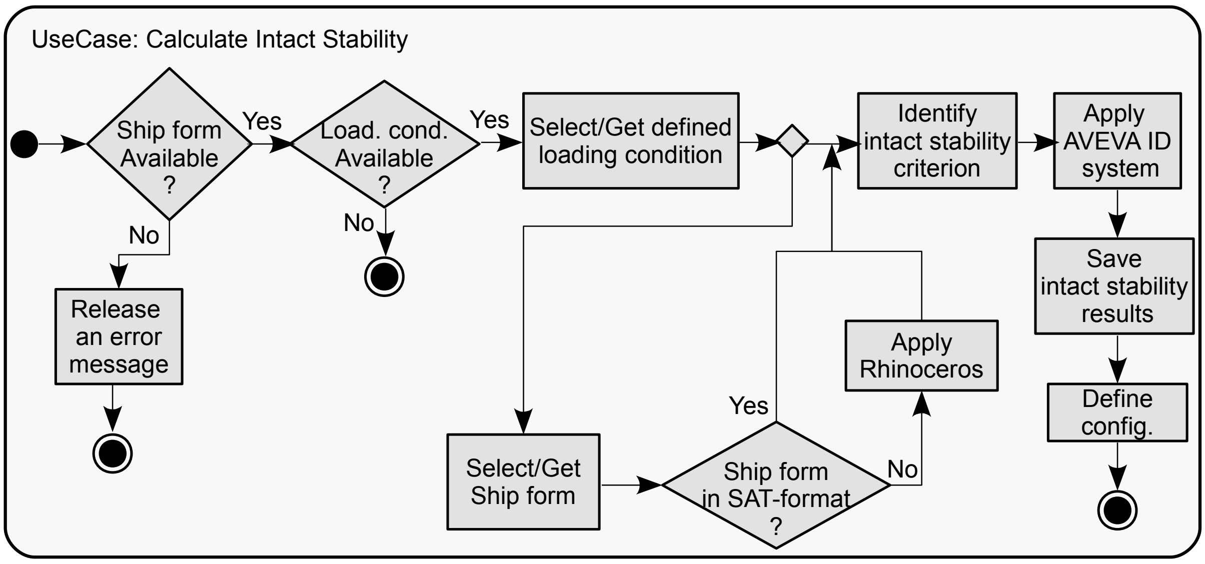

UseCase: Calculate Intact Stability
Ship form Available ?
Yes
No
Load. cond. Available ?
Yes
No
Select/Get defined loading condition
Identify intact stability criterion
Apply AVEVA ID system
Release an error message
Save intact stability results
Apply Rhinoceros
Define config.
Yes
Select/Get Ship form
Ship form in SAT-format ?
No

Figure 7.11.: UseCase: Calculate Intact Stability

ure 7.11, this UseCase is implemented to be executed as follows: After accessed by the responsible user, checking of the existence of any ship form within the central database regarding the addressed design project is automatically executed. The program searches in the central database if any loading condition is defined. If loading condition(s) are available, the user is asked to specify the basis, on which the checking of intact stability should be performed i.e select a specific loading condition and accordingly a specific ship form. After that, the user is asked to identify the intact stability criteria, which is organized in the central database as rules and regulations. The next procedure is to apply 'AVEVA-ID' as the underlying mechanism for this UseCase. This can be preceded by an intermediate step to transform the ship form into SAT-format by means of Rhinoceros-tool [16] in order to be readable by 'AVEVA-ID'. This is dependent on the basis of calculation and on the format of the ship form. The checking of intact stability is finished by saving the results implicitly and explicitly by making use of the information object *IntactStability* and by defining the corresponding configuration.

7.4.1.5. Predict Resistance

The predefined activity: *Predict Resistance* , which is described previously (see section 5.2.1.11.1), is implemented as shown in Figure 7.12. Two scenarios for predicting the resistance are realized within the implementation. These scenarios are differentiated from each other depending on the methodology of the prediction. After accessing this UseCase by the responsible user, it provides

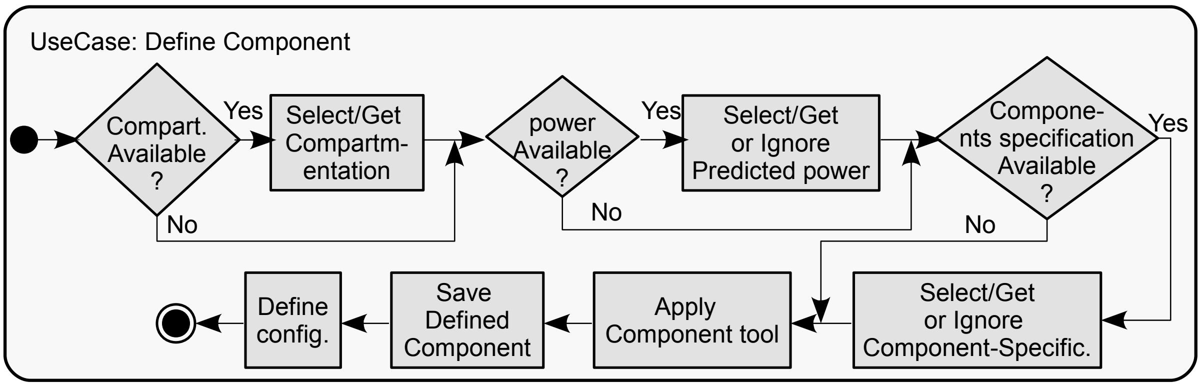

UseCase: Define Component
Compart. Available ?
Yes
No
Select/Get Compartm- entation
power Available ?
Yes
No
Select/Get or Ignore Predicted power
Compone- nts specification Available ?
Yes
No
Select/Get or Ignore Component-Specific.
Apply Component tool
Save Defined Component
Define config.

Figure 7.12.: UseCase: Predict Resistance

automatically the specified speed range. This is followed by asking the user to specify the floating position at which the resistance should be predicted. Then the UseCase searches automatically within the central database for the existence of ship forms regarding the addressed design project. This is followed by one of the two scenarios:

- Scenario1: In case no ship form is available, the prediction will be performed by empirical methods. Therefore, information about the ship's principal characteristics are obtained automatically from the central database and provided to the user. This is followed by applying the Respro XL-tool [146]. It is started automatically in order to make use of its functions, which offer the ability to predict resistance and sequentially power by means of many empirical methods like Holtrop/Mennen [89], Series 60 [143], Harvald [82], etc.
- Scenario2: In case one or more ship form(s) are available, the user is asked to select the required one. The Friendship-system [24] is then applied to create panel-model from the provided ship form. A control file, which controls the following computation, is created. The created panel model and control file are then used to automatically apply the potential-solver (ν-Shallo) developed by HSVA [52]. When the prediction of the resistance, which is computed at the specified floating position and a specific speed (it should be included in the specified speed range) is finished, the results are read and the user is given the ability

to repeat the prediction at a new condition (It is in fact a new speed from the speed range and the same specified floating position).

The results of the prediction are then treated in order to be saved to the central database within the corresponding *Resistance*-object. It is important to mention that the properties of the *Resistance*-object offer the ability to handle the panel model as well as the control file of Scenario2. This is followed by an automatic definition of the `Resistance` configuration from the instances of all used information objects.

7.4.1.6. Predict Power

The implemented UseCase shown in Figure 7.13 is dependent on the description of the *Predict Power* as a predefined activity (see section 5.2.1.11.3). This UseCase is implemented as follows.

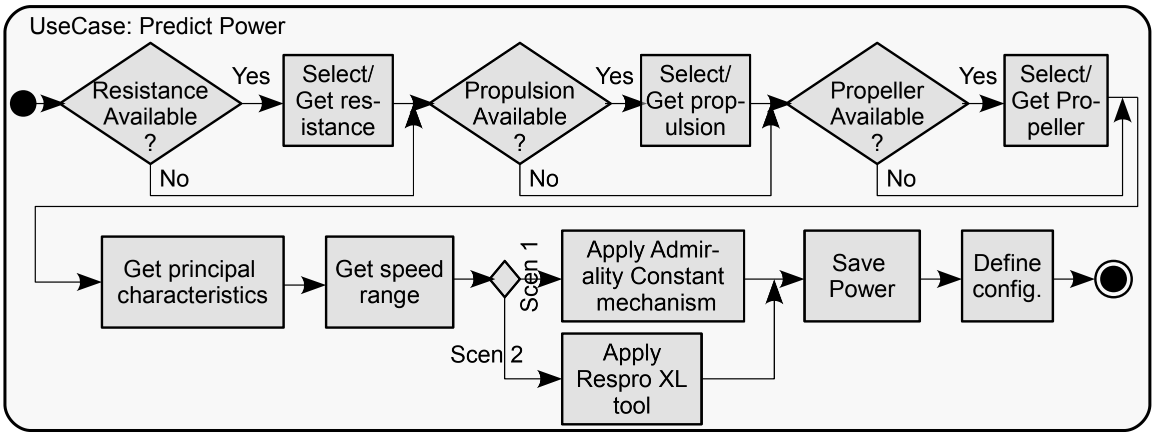


Figure 7.13.: UseCase: Predict Power

After it is accessed by the responsible user, within the central database regarding the addressed design project, it will be search for the available data about the predicted resistance, propulsion properties and propeller characteristics. Each of these searches is followed by a selection procedure in the case of the existence. In the case of the availability of some or all of the data, the user can select the Scenario2 in order to use the functions offered by Respro XL-tool to predict the power. In the other case, the user can select either the Scenario2 or the Scenario1 in order to make a first power estimation by means of simple developed functions based on admiralty-constant [131]. The required data, principal characteristics and speed range, are provided automatically by the system. The last procedures are to save the predicted power within the central database and to define the `Power` configuration.

7.4.1.7. Define Component

The implementation of this UseCase is accomplished in a way that accounts for the activity partitions described in Table A.10, which are included within the *Define Component* task as a predefined activity (see 5.2.1.12). When this UseCase, which is shown in Figure 7.14, is initiated by the responsible user, it checks automatically the availability of the compartmentation, predicted power in addition to any special requirements issued by the owner and saved as component specifications within the central database regarding the addressed design project. The checking of the availability is followed by selection/getting procedure for the compartmentation and selection/getting or ignoring procedure for the other. This considers the fact that the available power

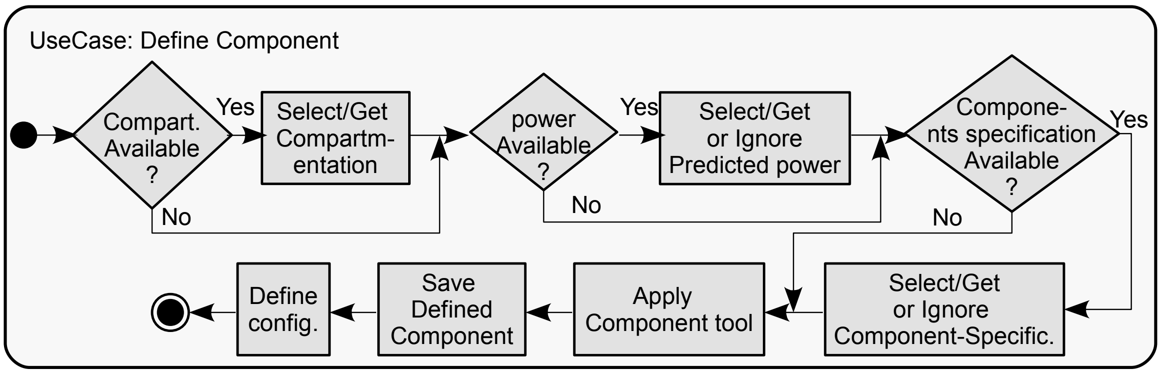

UseCase: Define Component
Compart. Available ?
Yes
No
Select/Get Compartm-entation
power Available ?
Yes
No
Select/Get or Ignore Predicted power
Compone-nts specification Available ?
Yes
No
Select/Get or Ignore Component-Specific.
Apply Component tool
Save Defined Component
Define config.

Figure 7.14.: UseCase: Define Component

or component specifications can be not useful for the definition of some ship components. An example of that is, when the type of the ship main engine is imposed by the owner and is available as a component specification, then this component specification can be ignored by the responsible user when he/she wants to define an outfitting-component. As soon as a component-definition is available regardless the details-level, it can be saved to the *ShipComponent*-object within the central database by means of the developed component tool. This tool offers the ability to locate the defined component by making use of the *Placement*-object. The defined component is then configured with all its details in order to finish the component definition.

7.4.2. Management/Reporting UseCases

Create Design Solution, *Update Status*, *Create Version*, *Control Changes* and *Create Report* are the implemented UseCases for management and reporting purposes, they are described in detail in the following sections.

7.4.2.1. Create Design Solution

In order to enable the definition of design solutions by taking into consideration the previous description (see section 5.2.2.1), the UseCase: *Create Design Solution* is implemented. The

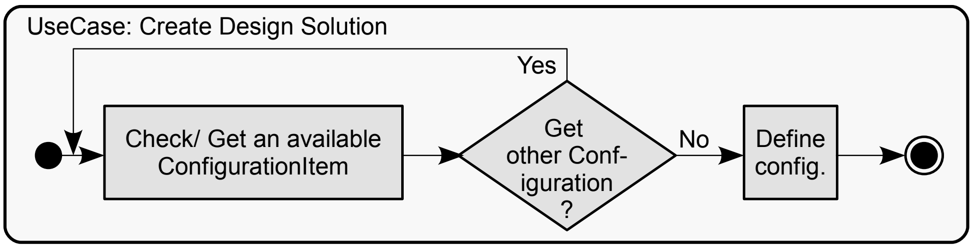

UseCase: Create Design Solution
Check/ Get an available ConfigurationItem
Get other Conf-iguration ?
Yes
No
Define config.

Figure 7.15.: UseCase: Create Design Solution

programming of this UseCase, which is depicted in Figure 7.15, provides the ability to create the design solutions as follows: When this UseCase is accessed by a responsible user, it checks automatically for existing configurations, which are defined by the termination of performing the design tasks regarding the addressed design project. The user is then asked to select a *ConfigurationItem* in order to be used. Finally, the selected configurations are used to define the

`Design Solution`-configuration. It is noteworthy to mention that the implementation of this UseCase does not allow the selection of two configurations representing the same task for the same design solution. As a result a design solution can include at most one configuration of the same early design task.

7.4.2.2. Update Status

This UseCase, which is shown in Figure 7.16, represents in fact the implementation of the pre-defined activity described in section 5.2.2.2. The accessing of this UseCase by the responsible

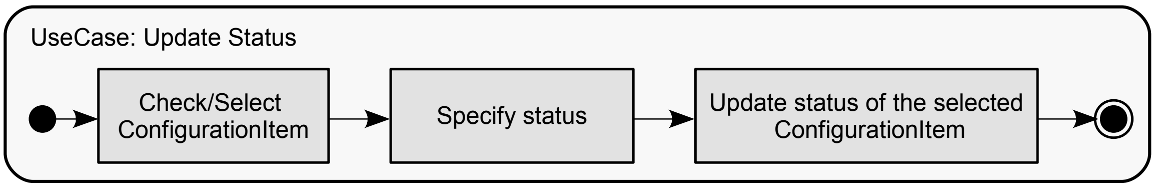

UseCase: Update Status
Check/Select ConfigurationItem
Specify status
Update status of the selected ConfigurationItem

Figure 7.16.: UseCase: Update Status

user is followed by an automatic search in the central database for the existing defined *ConfigurationItem*(s) regarding his/her responsibilities. The user is then asked to select one of the required available one as well as to specify the suitable status. This is followed by a re-definition of the selected configuration with the specified status.

7.4.2.3. Create Version

In order to provide versioning capabilities, a UseCase based on the principal previous discussion (see section 5.2.2.3) is programmed. This Usecase, which is shown in Figure 7.17, enables the

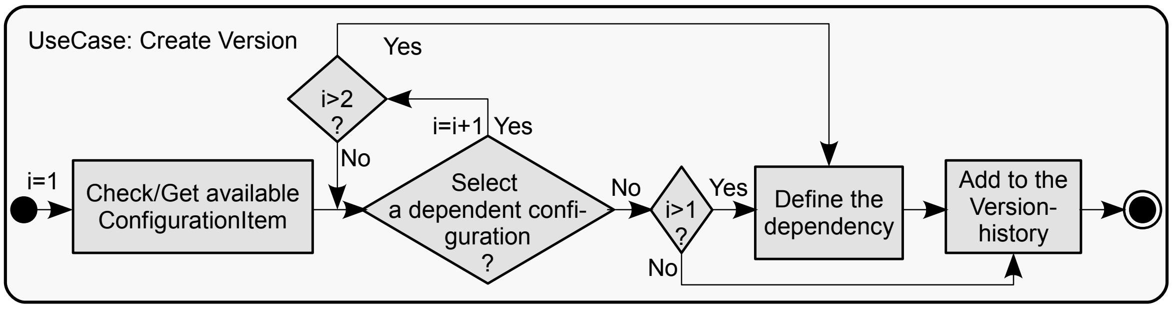

UseCase: Create Version
Yes
i>2 ?
i=i+1
Yes
No
i=1
Check/Get available ConfigurationItem
Select a dependent configuration ?
No
i>1 ?
Yes
No
Define the dependency
Add to the Version-history

Figure 7.17.: UseCase: Create Version

responsible user to apply the versioning of the early design activities represented by their defined *ConfigurationItem*(s) as follows. After accessing this UseCase, the system searches the existing configurations within the central database regarding the addressed design project. This is followed by selecting and obtaining of one of them. The user is then able to add the selected one directly to the version history of the represented early design task, or to select another one. In the latter case, the user is able to define the dependency between the two selected (obtained) *ConfigurationItem*(s). It is important to mention that the programming of this UseCase takes into consideration that in the second selection, just the configurations, which represent the same task as the task represented by the first selection, will be available to be selected. This means that the configurations, which represent different early design tasks such as `Power` and `Weight`, are not able to be versioned depending on each other.

7.4.2.4. Control Changes

The implementation of the change predefined activities described earlier (see section 5.2.2.4) is realized within this UseCase, which is termed as *Control Changes*. As it is shown in Figure 7.18,

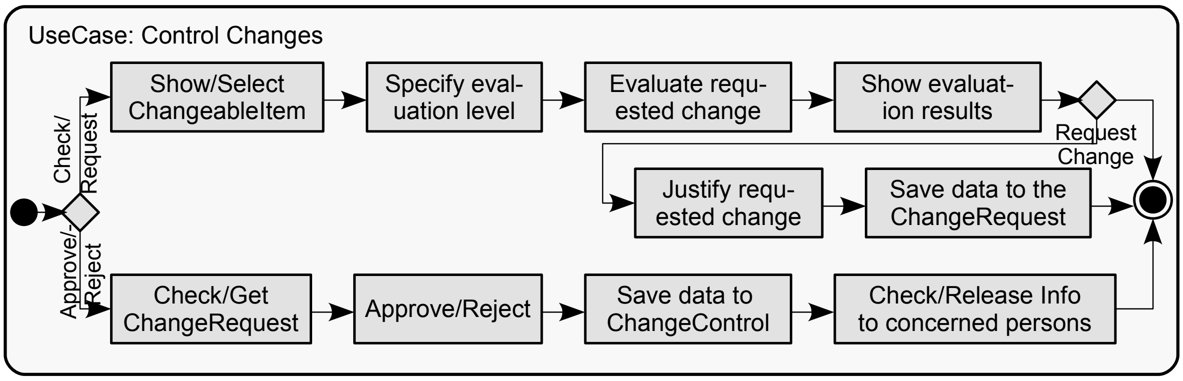

UseCase: Control Changes
Check/ Request
Approve/- Reject
Show/Select ChangeableItem
Specify eval- uation level
Evaluate requ- ested change
Show evaluat- ion results
Request Change
Justify requ- ested change
Save data to the ChangeRequest
Check/Get ChangeRequest
Approve/Reject
Save data to ChangeControl
Check/Release Info to concerned persons

Figure 7.18.: UseCase: Control Changes

this UseCase is implemented as follows: After it is accessed by the responsible user, he/she can choose between two functions:

- 'Check/Request': In this case, the UseCase searches automatically in the central database for the available *ChangeableItem*-instances (respecting the addressed design project) and displays them according to their types such as space elements, subdivision elements, etc. This is followed by a selection procedure to specify the actual one, which is to be evaluated regarding its influences in the case of a change. The system offers the ability to evaluate the change influences at two levels. Therefore, at the next step, the level of the evaluation should be specified by the user. For example, the level 'one' means, that all objects, which are interacting directly with the actual one, are checked and evaluated. Whereas the level 'two' means that the evaluation will check in addition to the previous objects the objects which are interacting directly with these objects and accordingly indirectly with the one which is subject to change. The next procedure is to execute the evaluation depending on the previous discussions (see sections 5.2.2.4 and 4.5.4). This is performed automatically by the UseCase and the resulted evaluation is shown to the user in order to select one of the two possibilities: the first is to exit and the second is to request a change. In the second case, the user should justify his/her request depending on the *RequestJustification*-object. The final procedure is to save all details as an instance of the *ChangeRequest*-object.
- 'Approve/Reject': When the user selects this choice, the UseCase checks automatically the existing *ChangeRequest*(s) within the central database regarding the addressed design project. The user is then asked to select one of the available *ChangeRequest*s in order to be approved or rejected. This is followed by saving all details within the central database as an instance of the information object *ChangeControl*. The last procedure is performed automatically by the system. It checks the users (according to their responsibilities), which can be concerned with the decisions made during the rejecting or approving a requested change. These users are then informed by messages about the decision that is made.

7.4.2.5. Create Report

The *Create Report* predefined activity, which is introduced in section 5.2.2.5, is implemented according to Figure 7.19. It is intended to serve the users (project managers, designers, etc.) to get

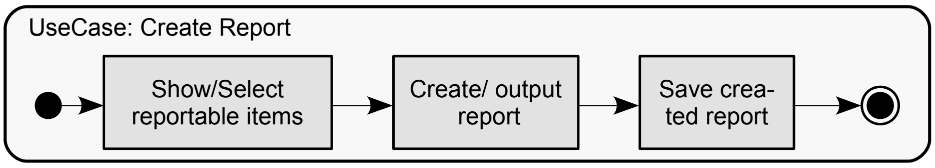


Figure 7.19.: UseCase: Reporting

a clear picture about which tasks are performed to which level. After accessed by the responsible user, it checks automatically for the existence of *ReportableItem*(s) (*DesignProject*, *PredefinedActivity* and *ConfigurationItem* in the central database regarding the addressed project. This is followed by a selection procedure to specify the required item, which is to be reported. The next step is to create the desired report, which contents (included data within the report) are dependent on the properties characterized to the corresponding information object such as *DesignProject* as well as *Report*-object. The last step is to save (reference) the created report to the central database. The following techniques are applied within this UseCase in order to enable the creation of the reports:

- 'XlsxWriter' [22]: a python package for creating Excel-files in python. It is used in cases when tables and diagrams are required to be created in order to organize the results.
- 'PyFPDF' [14]: a library for PDF file generation for Python in addition to '3D-pdf'-format [64] and accordingly the '3D-pdf'-convertor [20]. They are used to profit from all features of the pdf-format, which can be summed up as follows:
  - Easily accessible with standard Adobe Reader (see Figure 8.29), which is available on any PC, tablet. This makes the results easy to exchange between different project partners and independent of any CAD-software.
  - Structured 3D geometry representation, which can be viewed interactively by the receiver (ship owner, design office, etc.).
  - Combination of the different results environments: shape (3D or 2D models), text (meta-data). This allows the viewer to effectively review and examine the results (see Figure 8.29).
  - Small size of the resulted pdf-file for even complex structures, which increases the flexibility of the data-exchange.
  - Security of data throughout its life cycle, as it has been discussed above, is fostered by means of the Enterprise Rights Management (ERM) functions [124]. This supports the user by providing many capabilities like, data security insurance (password protection, digital signature, etc.) and access control which allows the sender to authorize the receiver's functionalities (read, print, etc.)

7.5. Multiple Representations Consistency

The implemented UseCases within the developed ship early design system, which have been described in the previous sections, show a variety of tools, which can be applied in the design process. This raises different issues which have to be handled efficiently. These issues in addition to their addressing methodology within the developed system are described in the following:

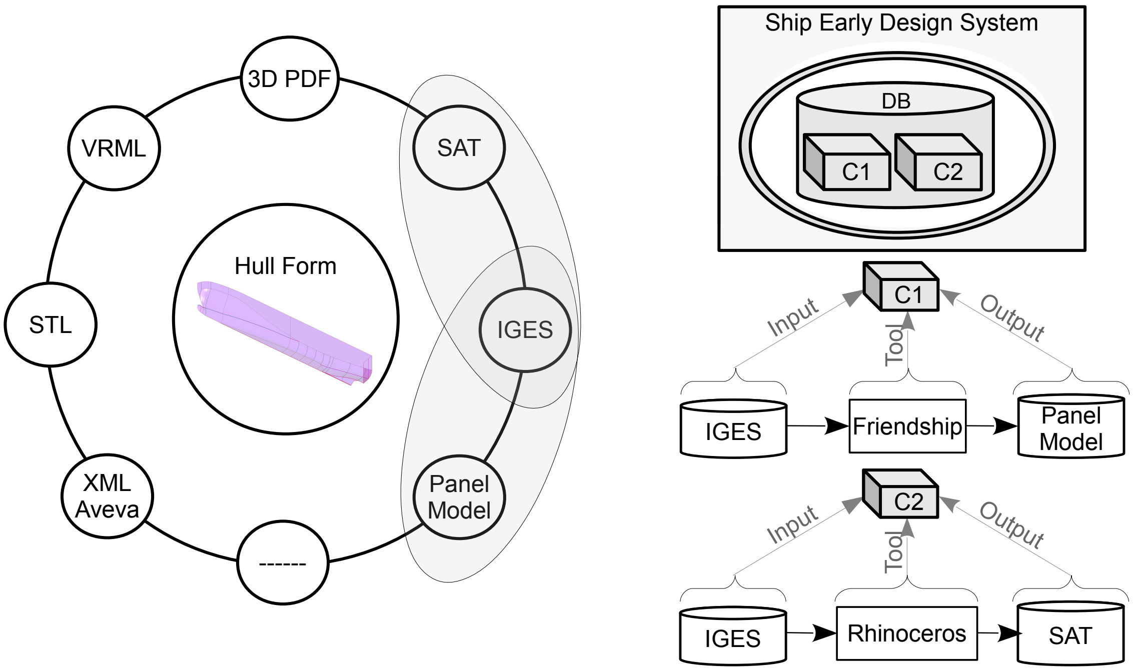


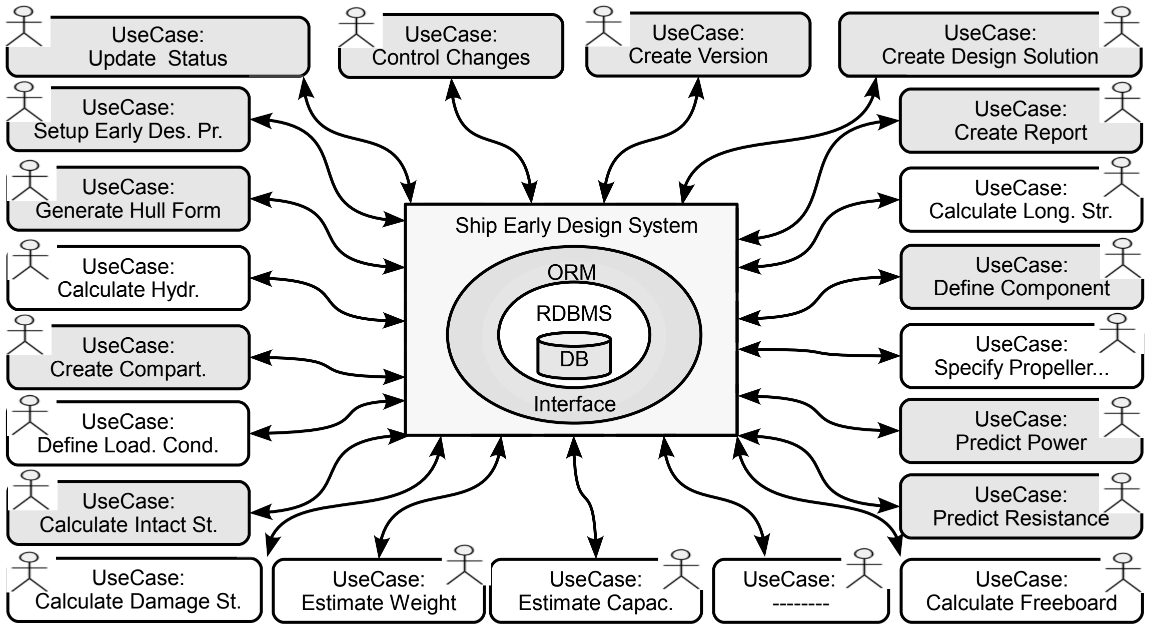
Figure 7.20.: Multiple Representations Consistency of a Hull Form

- Supporting of all kinds of representations adopted by applied tools is insured. This is attributed to the *Representation*-object (see section 4.4.1), which enables the user to deal with (handle) different formats as they are represented.
- Data-exchange between different applied tools is guaranteed by means of the built middleware represented by the central database (see section 7.1).
- The applied configuration management concept insures the consistency of data within the system. This issue can be discussed in two point of views:
  - The configuration concept offers the possibility to track any representation of an object, in the case when more than one representation regarding the same object is produced and saved within the central database. An example is the one shown in Figure 7.20. If many representations such as VRML, STL, IGES, Panel Model, etc. of the hull form exist within the central database, the tracking of each of them in order to detect its dependencies (who created it, at which time and by which tool) is guaranteed. For instance, as depicted in the upper Figure, by means of the configuration concept it is possible to identify that:
    - Both hull forms represented by IGES and Panel Model are related to the configuration 'C1' with different roles: input for the first and output for the second.
    - The Panel Model is generated from the IGES model by means of the Friendship-tool.
    - The same IGES-representation can be related to other configurations such as 'C2' with the same or a different role.
  - In case, that the results of an early design task are saved within the central database by means of a corresponding *Ship Calculation*-object (see section 4.3.11) and thereby an indicator to the stored results within the applied tool is defined, the consistency of resulted data is insured. This is attributed to the properties of the *Calculation*-object, which enables the storing of the defined indicator together with other saved data in the same object, which in turn will be configured by means of the configuration management concept.

To sum up, it can be stated, that within the developed system, the consistency of data is insured. Different representations of the same object are not isolated from each other but rather they are interdependent and can be tracked easily and efficiently.

8. Early design of a RoRo-Ship

The developed ship enterprise approach as it has been described in chapter 7 is applied on a RoRo-ship early design project. The scope of the RoRo-ship example is depicted in Figure 8.1. It shows



UseCase: Update Status
UseCase: Control Changes
UseCase: Create Version
UseCase: Create Design Solution
UseCase: Setup Early Des. Pr.
UseCase: Create Report
UseCase: Generate Hull Form
UseCase: Calculate Long. Str.
UseCase: Calculate Hydr.
UseCase: Define Component
UseCase: Create Compart.
UseCase: Specify Propeller...
UseCase: Define Load. Cond.
UseCase: Predict Power
UseCase: Calculate Intact St.
UseCase: Predict Resistance
UseCase: Calculate Damage St.
UseCase: Estimate Weight
UseCase: Estimate Capac.
UseCase: --------
UseCase: Calculate Freeboard
Ship Early Design System
ORM
RDBMS
DB
Interface

Figure 8.1.: Scope of RoRo-Ship Design Example

the implemented UseCases (in gray color) which are applied to receive the design results represented in the following sections. The used tools in the execution of the RoRo-Ship example are shown in Figure 8.2. Therein, the management/reporting tools refers to the mechanisms realized

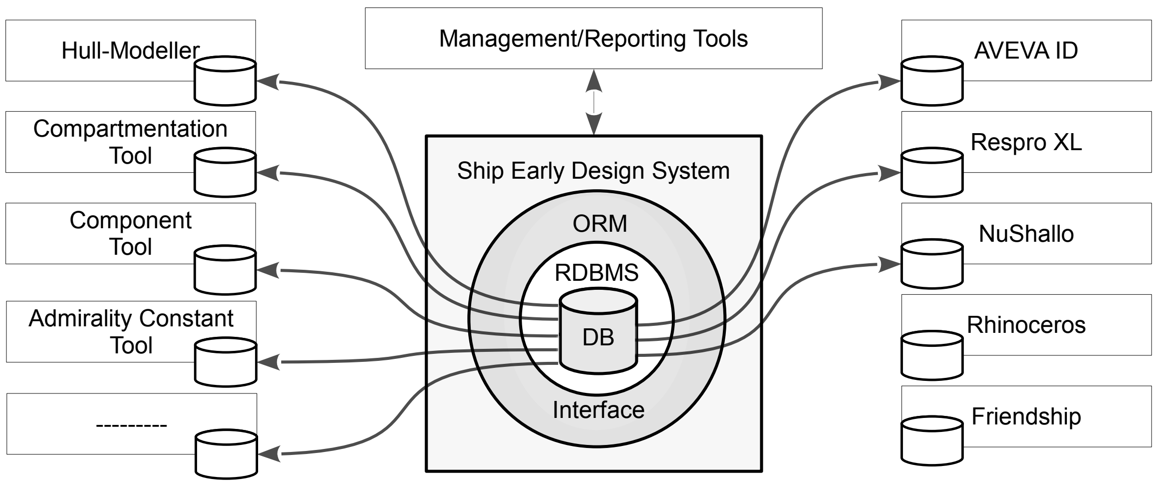


Figure 8.2.: Design Workbench of Example Shipyard

within the ‘Management/Reporting’-UseCases 7.4.2 to perform the management/reporting capabilities such as *Create Version*, *Control Changes*, etc. The arrows between the central database and the tools-stores represent the indicators, which will be defined and stored within the central database. As mentioned in section 7.3, the purpose of these indicators is to enrich the explicitly saved results (within the central database) of the early design tasks by defining pointers to the stored results within the applied tools. The executions by means of the implemented UseCases described in section 7.4, assumes that the data of the persons (users) of the considered shipyard (depicted in Figure 8.3) are already existing in the central database. It can be considered that

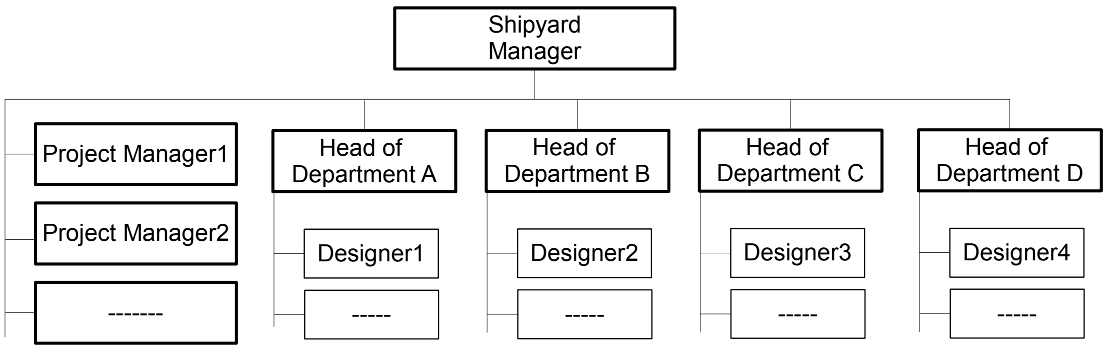


Figure 8.3.: Organizational Structure of Example Shipyard

each of them is registered in the system and can be associated with different design projects to perform specific tasks. Regarding the properties of the *PersonOrganization*-information object (see section 4.4.2), each person has a *PersonalInbox* which is used to communicate with other system-users.

8.1. Definition/Analysis

The definition/analysis early design tasks: *Setup of RoRo-Ship Early Design Project*, *Predict Power*, *Generate Hull Form*, *Create Compartmentation*, *Calculate Intact Stability*, *Predict Resistance* and *Define Ship Component* for the RoRo-ship are described in the following sections.

8.1.1. Setup of RoRo-Ship Early Design Project

Assuming that the preliminary general information and the owner requirements, which are the basis to setup any design project (see section 5.2.1.1), are available to the Shipyard Manager, then setting up an early design project for the ‘RoRo-Ship’ can be achieved by means of the implemented UseCase: *Setup Early Design Project.* The available data are supplied by the Shipyard Manager as shown in Figure 8.4. As it is depicted in the figure, the data are saved in the central database within the corresponding objects as follows, *ProjectName*: RoRo Ship, etc., in the *DesignProject*, L_{pp}: 182.5, *B*: 26, etc., in the *PrincipalCharacteristics*, *SpeedRange*: 18-24, *Endurance*: 8000.0 NM, etc., in the *Specification*-objects, *ShipType*: RoRo in the *ShipDesignation* in addition to the task-responsibilities which are listed in Table 8.1. Table 8.1 shows who is responsible for performing which task. For example, Designer2 is responsible for performing both tasks *Create Compartmentation* and *Check Intact Stability.* The Project Manager1 is authorized to perform *Approve/Reject Requested Change*, etc. As a result of the execution, the configuration with the: *Type*: Early Design Project, *Name*: RoRo Ship, *DefinedBy*: Shipyard Manger, etc., is defined and saved within the central database. The *ConfiguredItems* are the instances of all used objects, which are depicted in gray color in the lower figure, such as the *DesignProject*, the *PrincipalCharacteristics*, etc. Finally, after the successful setting up of the RoRo-project,

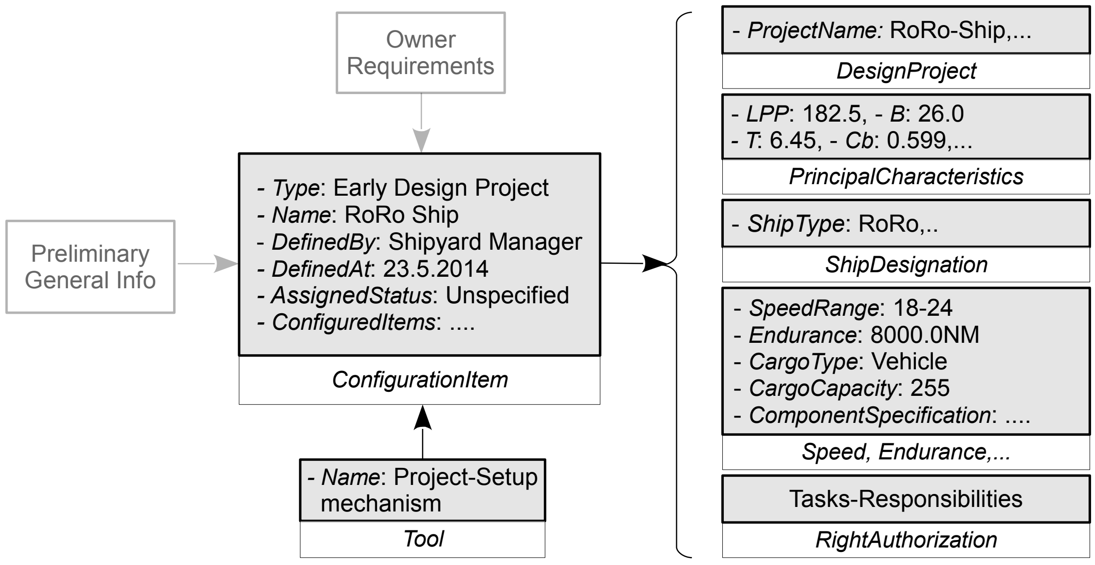


Figure 8.4.: Setup of RoRo-Ship Project

messages (with key data) are issued automatically by the system to the persons, who have responsibilities within the established RoRo-project. These users are shown in Figure 8.5, which describes the collaborative nature of the established RoRo-project.

Right Authorization			
Task	*Responsibility*	*Task*	*Responsibility*
Generate Hull Form	Designer1	Predict Power	Designer3
Create Compartmentation	Designer2	Check Intact Stability	Designer2
Define Component	Designer4	Predict Resistance	Designer3
Create Version	Project Manager1	Update Status	Head of Department(s)
Create Design Solution	Project Manager1	Check Change Influences, Request Change	Head of Department(s) Designer(s)
Approve/Reject Requested Change	Project Manager1	Reporting	All Users

Table 8.1.: Tasks and Responsibilities

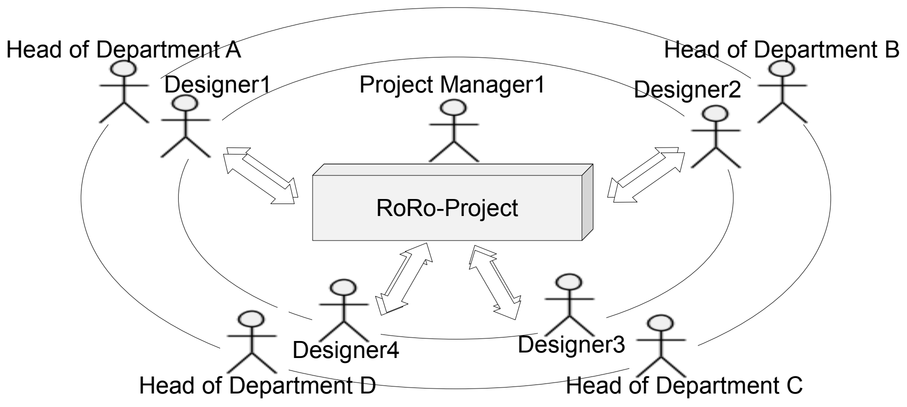
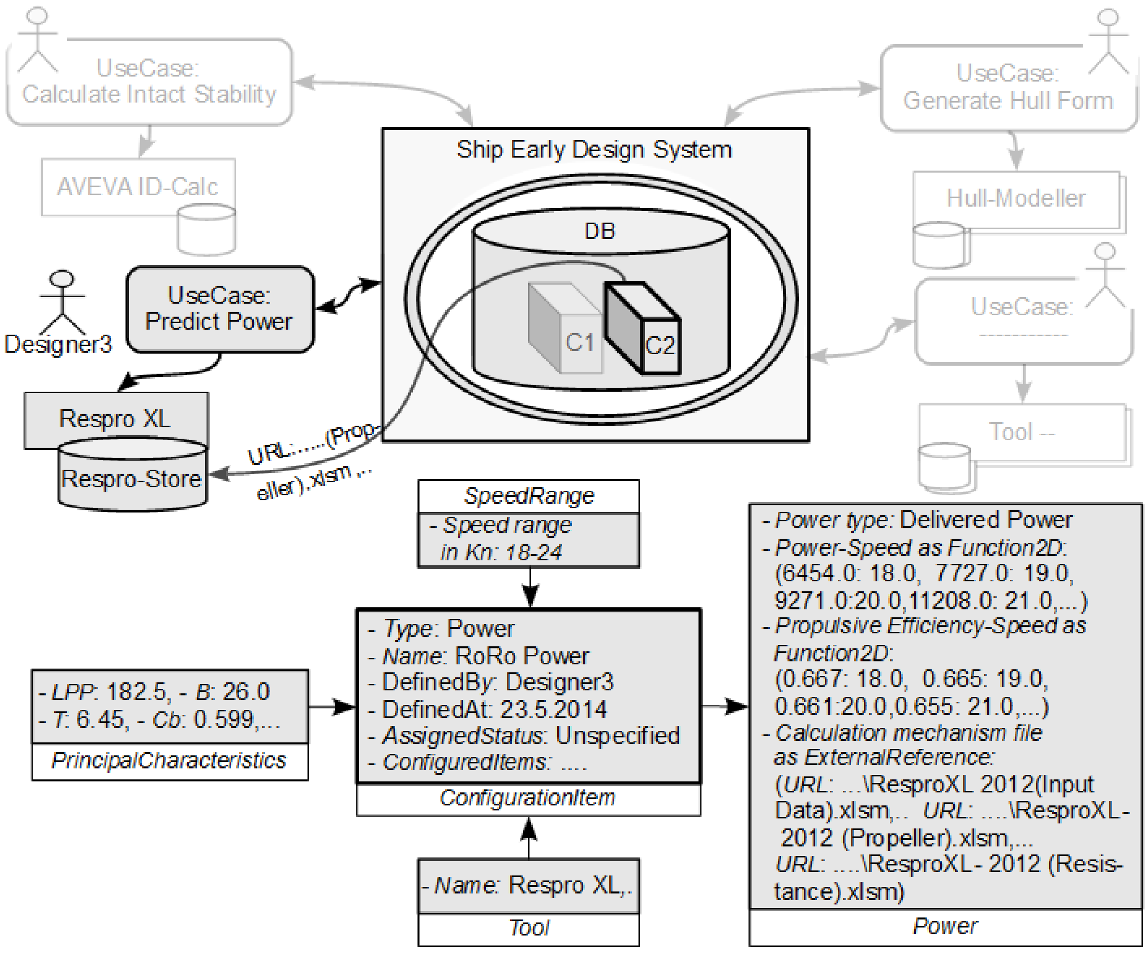


Figure 8.5.: RoRo-project as a collaborative project

8.1.2. Predict Power

Assuming that the power is required to be predicted by the machinery designers immediately after setting up the RoRo project, Designer3, who has the right to perform predict power functions (according to the task authorization by setting up the RoRo design project), is able to predict the power very early in the design process as follows.



UseCase: Calculate Intact Stability
AVEVA ID-Calc
Ship Early Design System
DB
C1
C2
UseCase: Generate Hull Form
Hull-Modeller
UseCase: -----------
Tool --
Designer3
UseCase: Predict Power
Respro XL
Respro-Store
URL:.....(Propeller).xlsm ,..
SpeedRange
- *Speed range in Kn*: 18-24
- *Type*: Power
- *Name*: RoRo Power
- DefinedBy: Designer3
- DefinedAt: 23.5.2014
- *AssignedStatus*: Unspecified
- *ConfiguredItems*: ....
ConfigurationItem
- *LPP*: 182.5, - *B*: 26.0
- *T*: 6.45, - *Cb*: 0.599,...
PrincipalCharacteristics
- *Name*: Respro XL,.
Tool
- *Power type*: Delivered Power
- *Power-Speed as Function2D*: (6454.0: 18.0, 7727.0: 19.0, 9271.0:20.0,11208.0: 21.0,...)
- *Propulsive Efficiency-Speed as Function2D*: (0.667: 18.0, 0.665: 19.0, 0.661:20.0,0.655: 21.0,...)
- *Calculation mechanism file as ExternalReference*: (*URL*: ...\ResproXL 2012(Input Data).xlsm,.. *URL*: ....\ResproXL-2012 (Propeller).xlsm,... *URL*: ....\ResproXL-2012 (Resistance).xlsm)
Power

Figure 8.6.: ConfigurationItem of Predict Power of the RoRo-Ship according to Scenario2

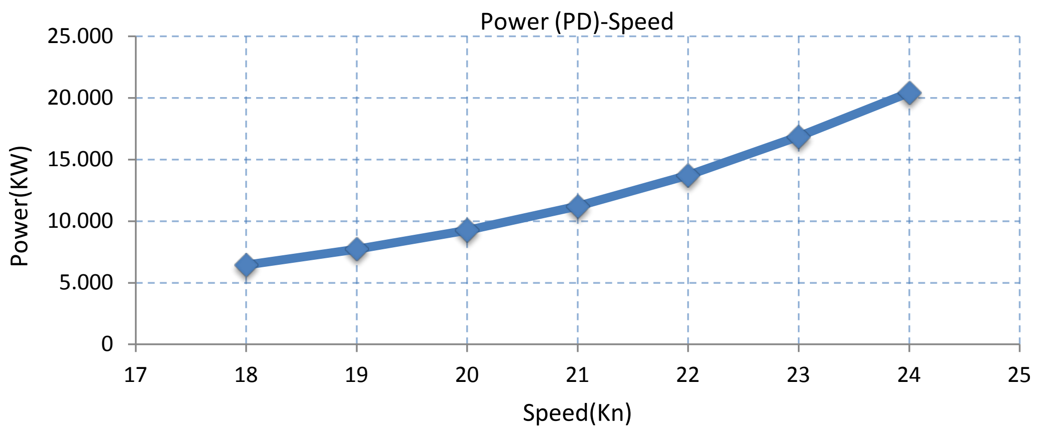


Figure 8.7.: Power-Speed Curve

After logging-in, Designer3 can perform the task he is responsible for, regarding the RoRo design project by means of the UseCase: *Predict Power* (see section 7.4.1.6). Considering the very early moment of time, at which this UseCase is accessed by the Designer3, where no resistance, propulsion or propeller characteristics are available within the central database. Designer3 is informed about the absence of these data on one hand and on the other hand, he is provided with the determined *PrincipalCharacteristics* and specified *SpeedRange*, which are related to the RoRo-project. These data enable the Designer3 to predict the power by means of Scenario2 represented by the Respro XL-tool. In order to complete the power prediction the resulted data should be saved within the central database. As it is shown in Figure 8.6, the following key data are saved to the *Power*-object. Firstly, *power type*: delivered power. Secondly, *power-speed* and *propulsive efficiency-speed*: the resulted power as well as the propulsive efficiency values are saved depending on the speed (by means of the *function2D*, see section 4.4.1), for example, at the speed 18.0 Kn the estimated power is 6454.0 KW and the propulsive efficiency is 0.667. This provides the ability at any time and completely independent of the Respro XL-tool to retrieve the power-speed dependency as it is shown in the diagram 8.7, which is created by the system by means of the 'XlsxWriter'-technique (see section 7.4.2.5). In addition to the indicator, which is defined to the power- as well as resistance-results within the Respro XL-store (see Figure 8.6). The `Power`-configuration, to which is referred as 'C2' in Figure 8.6, is defined automatically by the system. Therein, the *ConfiguredItems* are the instances of all used objects, which are depicted in gray color.

8.1.3. Generate Hull Form

After accessing the UseCase: *Generate Hull Form* by Designer1, who is responsible to create the hull form within the RoRo design project, the related *ShipDesignation* as well as the *PrincipalCharacteristics* are invoked automatically by the UseCase from the central database and provided to Designer1. By taking into consideration the implementation of this UseCase, Sce-

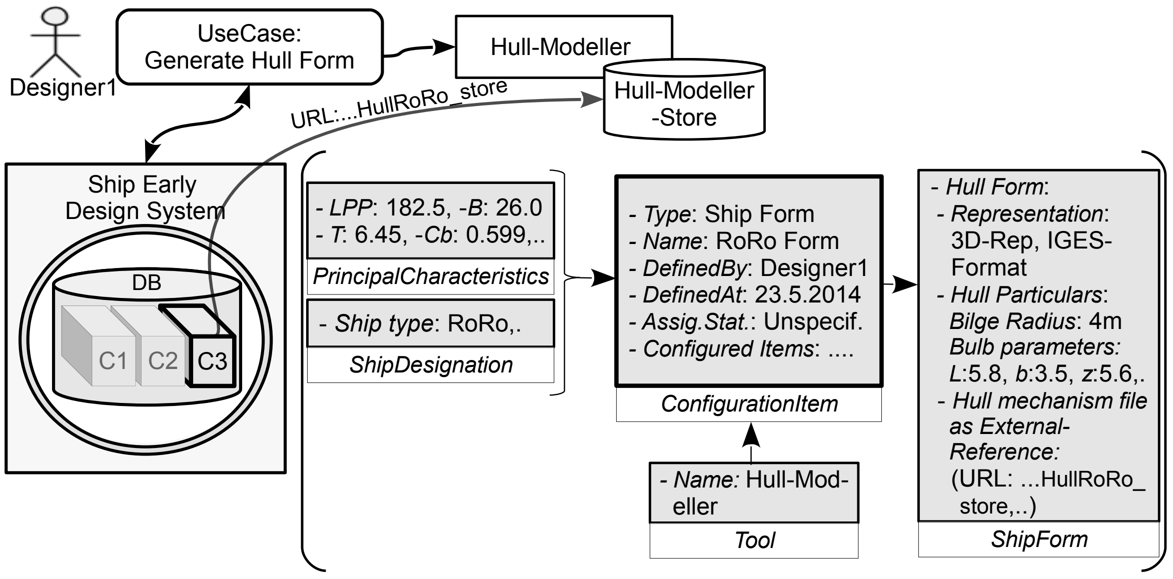


Figure 8.8.: ConfigurationItem of Generate Hull Form of the RoRo-ship

nario1 is selected by Designer1 to perform the definition of the hull form. This is followed by applying the 'Hull-Modeller'-mechanism, which is described previously (see the description in section 7.4.1.2.1), and the generation of the required hull form is executed in almost automatic

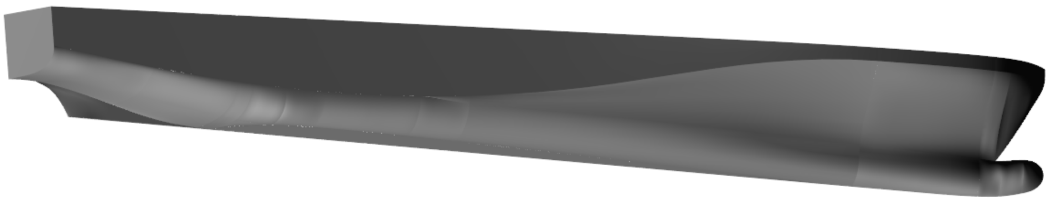
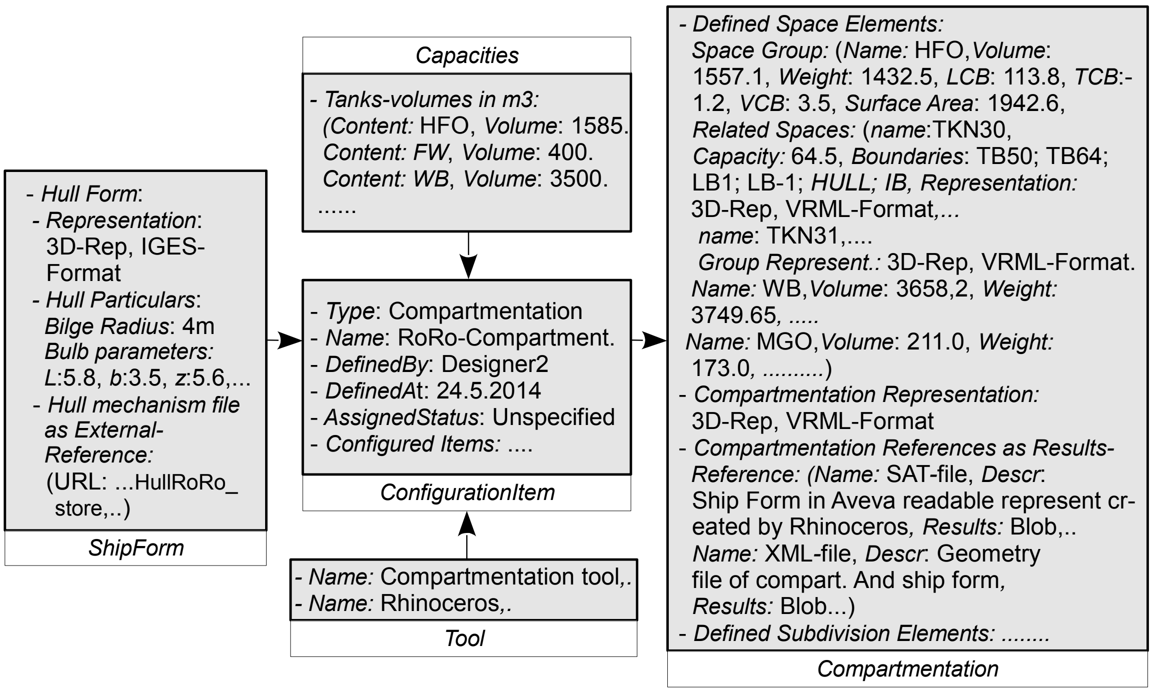


Figure 8.9.: Resulted Hull Form of the RoRo-Ship

process. The resulted RoRo hull-form, which is shown in Figure 8.9, with all its available properties such as the 3D-representation in IGES-format are then saved to the central database as an instance of the object *ShipForm*. Moreover, a pointer to the modeller-store (where the hull-form, which is used as a basis to create the required one) is defined and saved. As it is shown in Figure 8.8, the `Ship Form`-configuration, which combines all previous details of the hull creation, is then defined and saved as an instance of the object *ConfigurationItem*.

8.1.4. Create Compartmentation

The creation of the RoRo-compartmentation is achieved by means of the UseCase: *Create Compartmentation*. It is conducted by the Designer2, who is responsible for performing the compartmentation task (see Table 8.1). By taking into consideration the implementation of this UseCase and by assuming:



Capacities
- Tanks-volumes in m3: (Content: HFO, Volume: 1585. Content: FW, Volume: 400. Content: WB, Volume: 3500. ......
- Hull Form:
- Representation: 3D-Rep, IGES-Format
- Hull Particulars: Bilge Radius: 4m Bulb parameters: L:5.8, b:3.5, z:5.6,...
- Hull mechanism file as External-Reference: (URL: ...HullRoRo_ store,..)
ShipForm
- Type: Compartmentation
- Name: RoRo-Compartment.
- DefinedBy: Designer2
- DefinedAt: 24.5.2014
- AssignedStatus: Unspecified
- Configured Items: ....
ConfigurationItem
- Name: Compartmentation tool,.
- Name: Rhinoceros,.
Tool
- Defined Space Elements: Space Group: (Name: HFO,Volume: 1557.1, Weight: 1432.5, LCB: 113.8, TCB:-1.2, VCB: 3.5, Surface Area: 1942.6, Related Spaces: (name:TKN30, Capacity: 64.5, Boundaries: TB50; TB64; LB1; LB-1; HULL; IB, Representation: 3D-Rep, VRML-Format,... name: TKN31,.... Group Represent.: 3D-Rep, VRML-Format. Name: WB,Volume: 3658,2, Weight: 3749.65, ... Name: MGO,Volume: 211.0, Weight: 173.0, ...)
- Compartmentation Representation: 3D-Rep, VRML-Format
- Compartmentation References as Results-Reference: (Name: SAT-file, Descr: Ship Form in Aveva readable represent created by Rhinoceros, Results: Blob,.. Name: XML-file, Descr: Geometry file of compart. And ship form, Results: Blob...)
- Defined Subdivision Elements: ....
Compartmentation

Figure 8.10.: ConfigurationItem of Compartmentation of the RoRo-Ship

- that the *Estimate Capacities*-task (see section 5.2.1.9) is performed by means of the corresponding UseCase (see Figure A.18) and the *Capacities*-results are saved within the central database
- no space-requirements (at this moment of time) are saved within the database regarding the components-definition within the RoRo project

then and after the accessing the UseCase by the Designer2, the only existing *ShipForm* and the only available *Capacities* regarding the RoRo project are provided by the UseCase to the Designer2 in order to be taken into account throughout the creation (see Figure 8.10). In the next procedure, the developed compartmentation tool described in section 7.4.1.3.1 and depending on the available data is applied as it is described in the following.

Compartmentation-definition: According to the applied compartmentation approach (see section 7.4.1.3.1), the compartmentation is defined by utilizing the developed syntax (Macro Language). Some of the definitions included within the compartmentation-definition-script in accordance to thereby developed macro-language are:

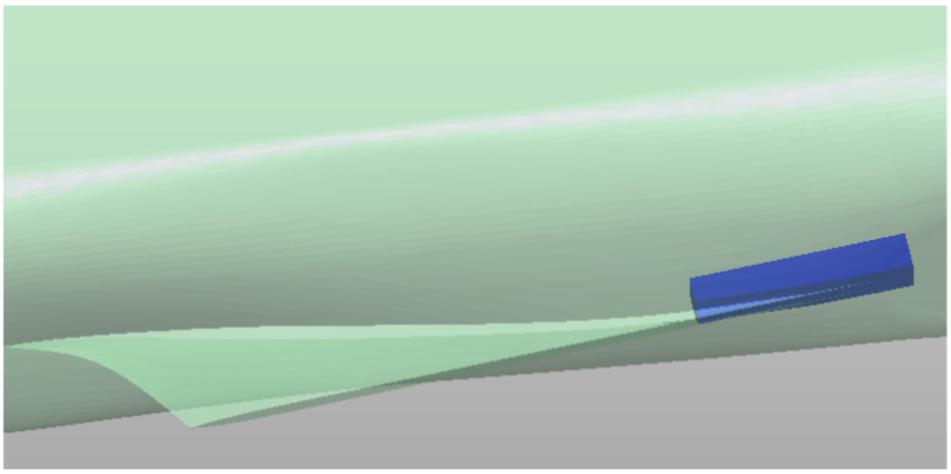


Figure 8.11.: 3D-Representation of the Sludge-Tank

Header Part of the compartmentation-definition script: This part includes the identification of the ship form, the definition of the global co-ordinate system, and the definition of spacing tables:

```
RoRoHull
ORIGIN: 0.0
X-AXIS: positive direction FORWARD
Y-AXIS: positive direction PORT
Z-AXIS: positive direction UPWARDS
Longitudinal-table:  FR-5, 0.60; FR20, 0.75; FR240, 0.60;
Transversal-table:  Y0, 0.0; Y1, 1.2; Y2, 4.5; Y3, 9.3;
Vertical-table:  Z0, 0.0; Z1, 1.74; Z2, 8.6; Z3, 16.7; Z4, 22;
```

Definition of a Sludge-Tank: The definition of the subdivision elements which represent the boundaries of this tank and also the definition of the tank as a simple space element are given. In Figure 8.11, the resulted 3D-representation of the sludge-tank is shown.

```
...
IB: Z1;
TB50:  FR50 HULL IB;
TB64:  FR64 HULL IB;
LB1:  Y1;
LB-1:  Y-1;
ST: TB50 TB64 LB1 LB-1 HULL IB SLUDGE;
...
```

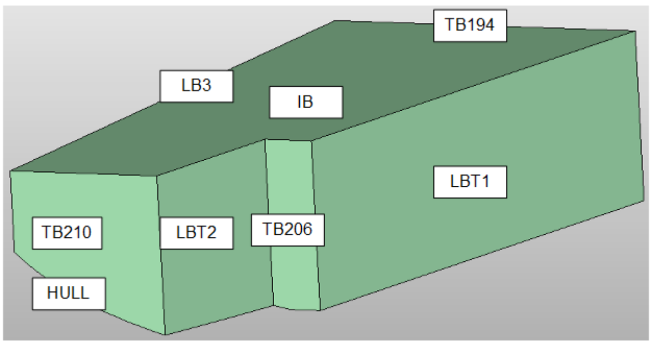


Figure 8.12.: 3D-Representation of a Complex Water Ballast-Tank

Definition of a Complex Water Ballast Tank: In Figure 8.12 the 3D-representation of a water ballast tank is shown. The resulted 3D-representation of all defined water ballast tanks of the RoRo ship is depicted in Figure 8.13. The RoRo ship with all defined space elements is shown in Figure 8.29.

```
...
T4-1:  TB194 TB210 LBT1 LB3 HULL IB WB;
T4-2:  TB206 TB210 LBT1 LBT2 HULL IB WB;
T4:  /ADD (T4-1) /SUB (T4-2) WB;
...
```

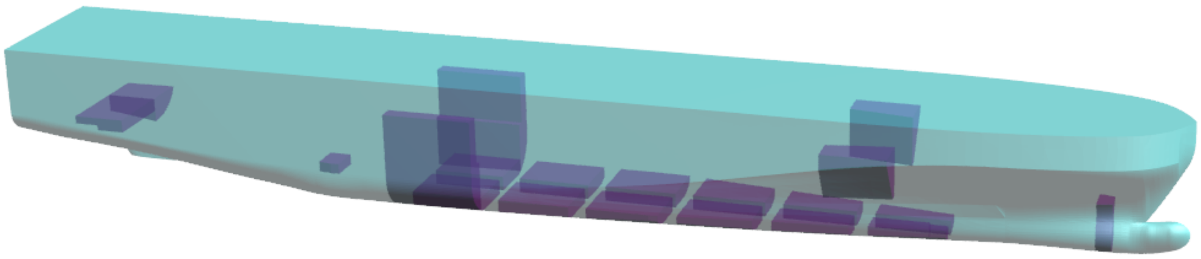


Figure 8.13.: 3D-Representation of the Water Ballast-Tanks

Following the creation of the compartmentation, the resulted compartmentation geometry is processed automatically and the output is saved within the central database as an instance of the *Compartmentation*-object as it is shown in Figure 8.10. For example, the resulting HFO-tanks, which are listed in Table 8.2, are saved as a *SpaceGroup*. As it is shown in Figure 8.10, this *SpaceGroup* is characterized by many properties, such as *Volume*: 1557.1, *LCB*: 113.8, etc., in addition to the *Related Spaces* with all their data, such as the *SpaceElement*, which is a tank with the *name*: TKN30 and *Boundaries*: (TB50, TB64,..), etc. Furthermore the *Defined Subdivision Elements* with their related data are saved.
Moreover, the complete *Compartmentation Representation* is saved in VRML-format and stored as a 3D-*Representation*. It is worth noting that in addition to the previous data, the data, which are resulted from intermediate steps during the compartmentation, are also saved. This is represented by saving the compartmentation geometry XML-file and the SAT-representation of the ship form, which is created by the Rhinoceros-tool in order to be readable by AVEVA-ID-tool. After saving of compartmentation results, they are configured together with all other details as an instance of the *ConfigurationItem*-object. An example of the multiple representations consistency aspect, which is discussed in section 7.5, is the consistency of the multiple representations of the ship hull form within the defined `Compartmentation`-configuration. As it is shown in

Name	*Volume(m3)*	*Weight(t)*	*LCB(m)*	*TCB(m)*	*VCB(m)*	*Sur. Area(m2)*
TKN30	64.5	59.3	168.7	0.0	5.0	130.7
TKN31	224.3	206.4	55.8	-5.0	1.2	293.6
TKN32	224.3	206.4	55.8	5.0	1.2	293.6
TKN33	407.5	374.9	160.6	-2.0	5.0	403.7
TKN34	40.2	37.0	47.2	-7.5	3.0	73.2
TKN35	407.5	374.9	160.6	2.0	5.0	403.7
TKN36	65.7	60.4	45.9	-9.64	2.7	127.6
TKN37	55.2	50.8	47.5	-5.1	0.9	109.0
TKN38	67.9	62.5	38.2	-8.5	3.1	107.7
Sum	1557.1	1432.5	113.8	-1.2	3.5	1942.6

Table 8.2.: Spaces related to '*SpaceGroup*(*Name*: HFO)'

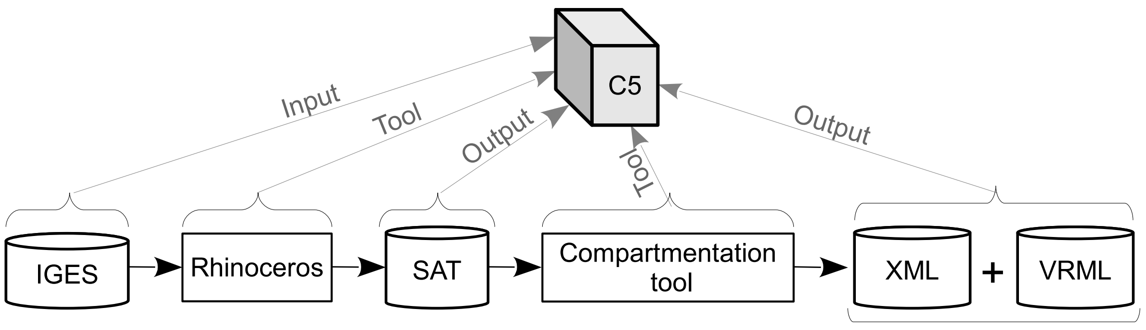


Figure 8.14.: Multiple Representations Consistency within `Compartmentation`-Configuration

Figure 8.14, the defined configuration (to which is referred as 'C5') contains four representations of the ship hull form, which are IGES and SAT representations in addition to the XML and VRML formats. They are organized in a way which enables the user to track their dependencies. For example it is easy to identify that the SAT-representation of the ship form and the XML as well as VRML representations of the ship form and compartmentation are created by the Designer2 at the date 24.5.2014 by means of the Rhinoceros- and Compartmentation-tools depending on the IGES-representation of the ship hull form.

8.1.5. Calculate Intact Stability

The following example of checking the intact stability of the RoRo-ship is implemented by assuming, that the loading condition of the RoRo-ship in ballast at departure is defined and stored within the central database by means of the UseCase: *Define Loading Condition* (see Figure A.22). Therefore, after accessing the UseCase: *Calculate Intact Stability* described in section 7.4.1.4 by Designer2 (who is responsible for performing this task according to the assignment made at project setup, see Table 8.1), the stored *LoadingCondition* as it is shown in Figure 8.15 is selected to check the intact stability of the RoRo-ship. By taking into account the implementation of the UseCase, this is followed by the identification of the intact stability criteria IMO 749A, which govern the calculation. After applying AVEVA ID for executing the check on the intact stability criteria, the results are stored in the central database as an instance of the *IntactStability*-object as it is shown in Figure 8.15. This instance in addition to the other object-instances (shown in Figure 8.15 in gray color) are configured as an instance of the *ConfigurationItem*-object (to which is referred as 'C7' in Figure 8.15). The saving of the results based on the *IntactStability*-object can serve as an example of the methodology adopted to store the calculation results (see 4.3.11).

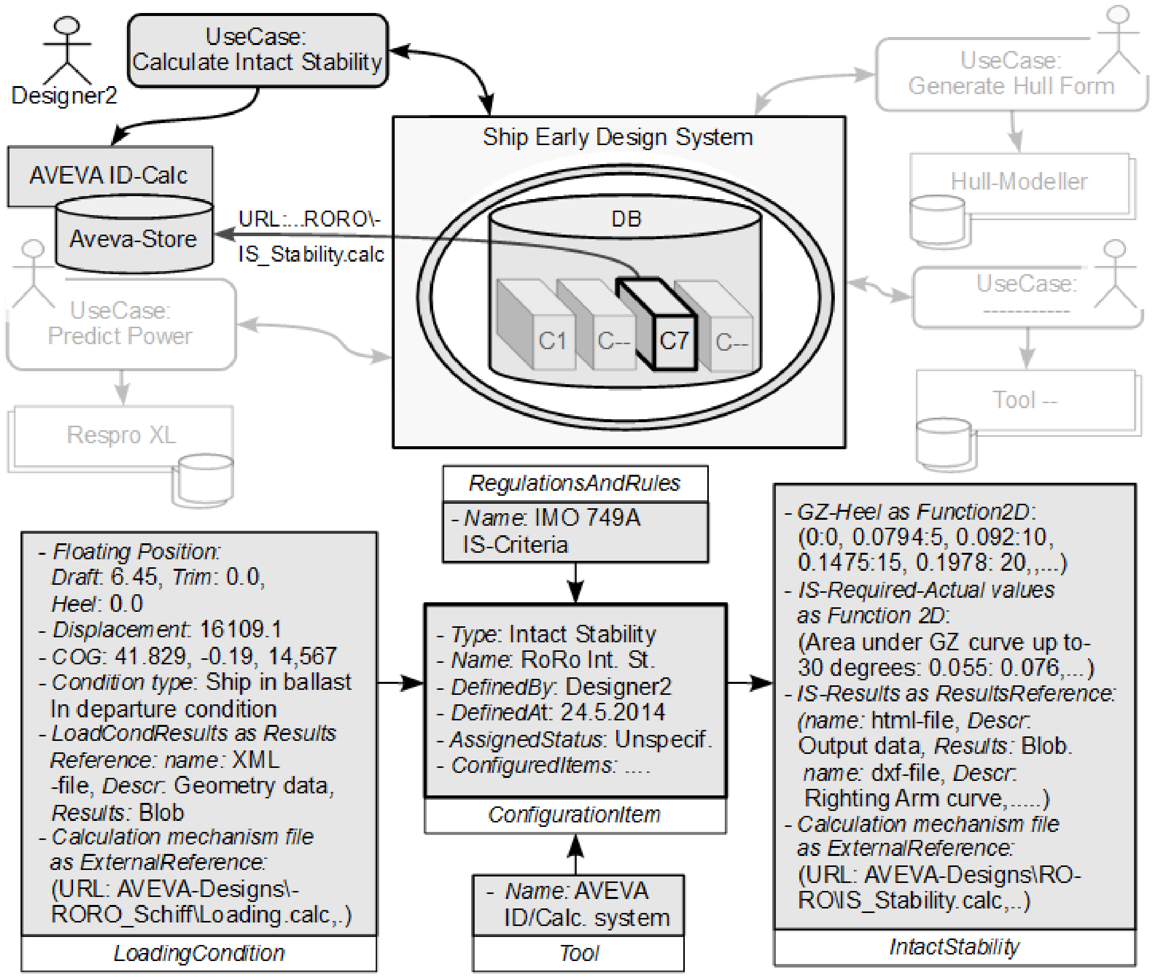

Designer2
UseCase: Calculate Intact Stability
UseCase: Generate Hull Form
AVEVA ID-Calc
Aveva-Store
URL:...RORO\-
IS_Stability.calc
Ship Early Design System
DB
C1
C--
C7
C--
Hull-Modeller
UseCase: ----------
UseCase: Predict Power
Tool --
Respro XL
RegulationsAndRules
- Name: IMO 749A
IS-Criteria
- Floating Position: Draft: 6.45, Trim: 0.0, Heel: 0.0
- Displacement: 16109.1
- COG: 41.829, -0.19, 14,567
- Condition type: Ship in ballast In departure condition
- LoadCondResults as Results Reference: name: XML -file, Descr: Geometry data, Results: Blob
- Calculation mechanism file as ExternalReference: (URL: AVEVA-Designs\-RORO_Schiff\Loading.calc,.)
LoadingCondition
- Type: Intact Stability
- Name: RoRo Int. St.
- DefinedBy: Designer2
- DefinedAt: 24.5.2014
- AssignedStatus: Unspecif.
- ConfiguredItems: ....
ConfigurationItem
- Name: AVEVA ID/Calc. system
Tool
- GZ-Heel as Function2D: (0:0, 0.0794:5, 0.092:10, 0.1475:15, 0.1978: 20,,...)
- IS-Required-Actual values as Function 2D: (Area under GZ curve up to-30 degrees: 0.055: 0.076,...)
- IS-Results as ResultsReference: (name: html-file, Descr: Output data, Results: Blob. name: dxf-file, Descr: Righting Arm curve,.....)
- Calculation mechanism file as ExternalReference: (URL: AVEVA-Designs\RO-RO\IS_Stability.calc,..)
IntactStability

Figure 8.15.: ConfigurationItem of Intact Stability of the RoRo-Ship

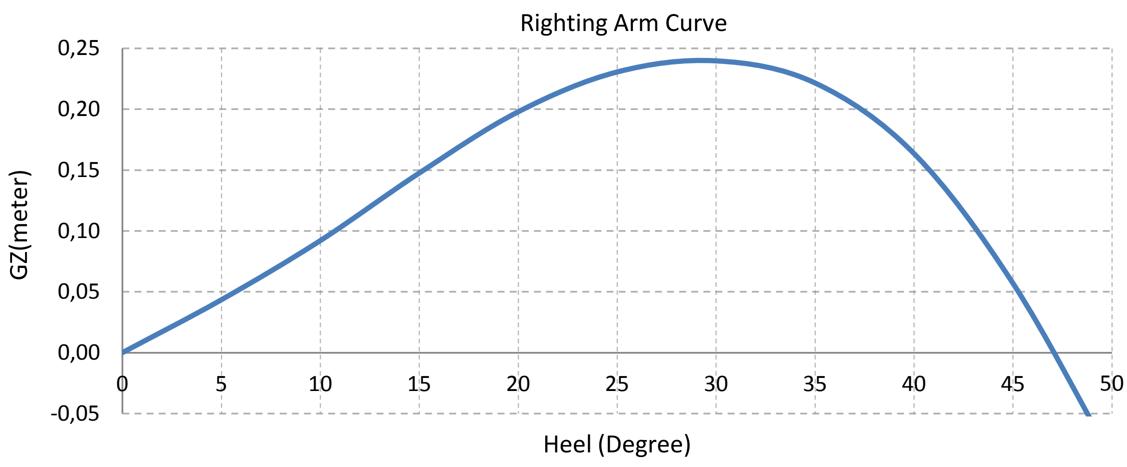


Figure 8.16.: Righting Arm Curve

The key results are saved explicitly in the central database whether the other details are saved implicitly in the central database by defining and saving an indicator to the results stored within the applied tool. The data of the righting arm curve, which represent the values of the *GZ* in accordance to the *Heel*-values are stored as a *Function2D*. This allows at any time to retrieve these data from the central database independent of the applied mechanism.

Criterion	*Required Value*	*Actual Value*
Area under GZ curve up to 30 degrees	0.055	0.073
Area under GZ curve from 30 to 40 deg. or downflood	0.030	0.037
Area under GZ curve up to 40 degree or downflood	0.090	0.110
Initial GM in meter	0.15	0.541
GZ at an angle > 30 degrees	0.2	0.24
Max GZ to be at an angle > 30 degrees	30.0	30.146
IMO Weather Criterion (Maximum Initial Angle Of Heel)	16.0	11.056
IMO Weather Criterion (Areas)	1.0	1.074

Table 8.3.: Intact Stability Criteria

An example is the one shown in Figure 8.16, which is created by means of the 'XlsxWriter'-technique depending on the data obtained from the database from the *IntactStability*-object. Moreover, the actual resulted data of the intact stability criteria such as *Area under GZ curve up to 30 degrees*, *Area under GZ curve from 30 to 40 degrees or downflood*, etc in accordance with the required values specified within the criteria-rules are saved as well in a form of a *Function2D*. These results, which are listed in Table 8.3, show that the designed RoRo-ship satisfies the intact stability criteria by comparing required and resulted values. In addition to the above saved data, the following data are stored. Firstly, the resulted data as they are outputted from the AVEVA ID (in dxf- and html formats) are saved as *ResultsReference*. Secondly, an indicator to the intact stability results within the AVEVA ID-store is defined and stored as an *ExternalReference*.

8.1.6. Predict Resistance

In the following, the prediction of the total resistance of the RoRo-ship by means of the implemented UseCase: *Predict Resistance* described in section 7.4.1.5 is introduced. Regarding the responsibilities listed in Table 8.1, this task is performed by the Designer3. By taking into consideration the implementation of this UseCase, Designer3 is provided by the specified *SpeedRange* (18-24 Kn) directly after accessing the UseCase. The *FloatingPosition* (*Draft*: 6.45, *Trim*: 0.0, and *Heel*: 0.0), at which the resistance will be predicted, is specified by Designer3. The selection of the Scenario2 by Designer3 is followed by obtaining the *ShipForm* (in IGES-representation). This is followed by the creation of the panel model (RoRo.pan) of the RoRo-ship, which is generated from the obtained IGES-model by means of the Friendship-tool. The resistance is predicted seven times at seven conditions (the specified *FloatingPosition* and the seven speeds in accordance to the *SpeedRange*: 18, 19,..., 23, 24) by means of the potential solver: ν-Shallo. For each computation the control file: (RoRo-Resistance.ctl) is created and the total resistance is read from the (.ptl)-file.

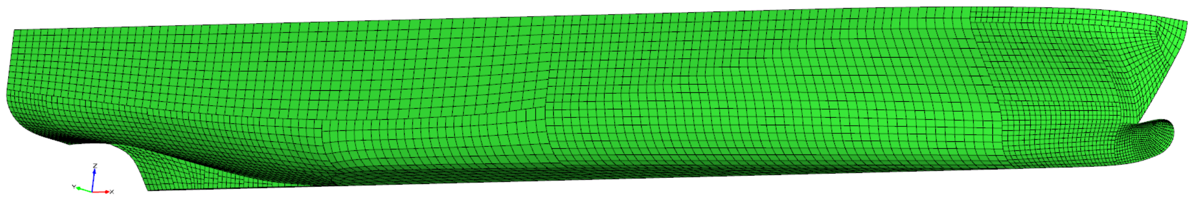


Figure 8.17.: Panel-model of the RoRo-Ship

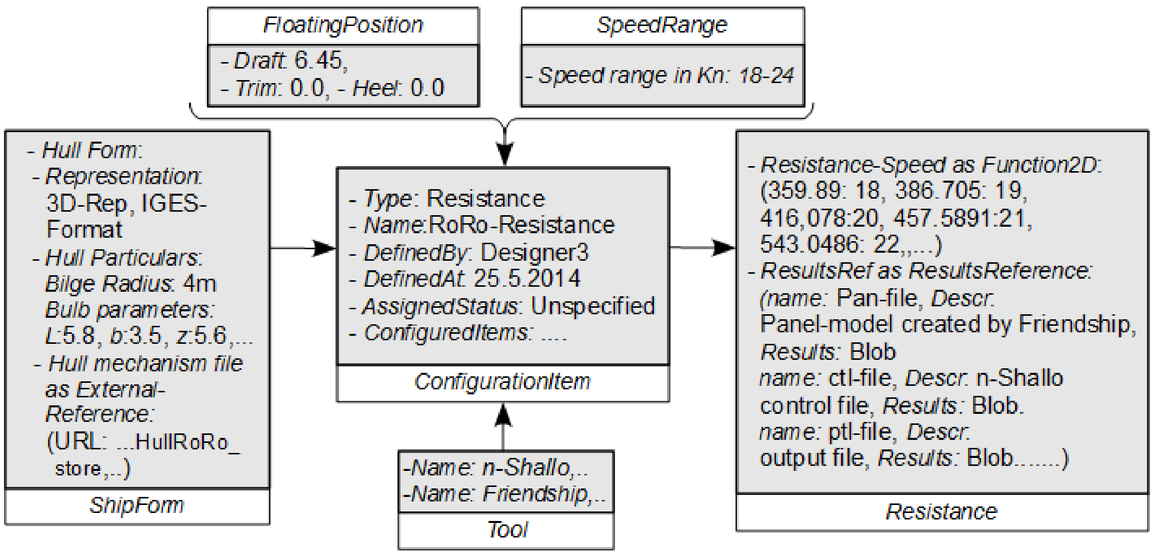

FloatingPosition
- Draft 6.45,
- Trim: 0.0, - Heel: 0.0
SpeedRange
- Speed range in Kn: 18-24
- Hull Form:
- Representation: 3D-Rep, IGES-Format
- Hull Particulars: Bilge Radius: 4m Bulb parameters: L:5.8, b:3.5, z:5.6,...
- Hull mechanism file as External-Reference: (URL: ...HullRoRo_store,..)
ShipForm
- Type: Resistance
- Name:RoRo-Resistance
- DefinedBy: Designer3
- DefinedAt: 25.5.2014
- AssignedStatus: Unspecified
- ConfiguredItems: ....
ConfigurationItem
-Name: n-Shallo,..
-Name: Friendship,..
Tool
- Resistance-Speed as Function2D: (359.89: 18, 386.705: 19, 416,078:20, 457.5891:21, 543.0486: 22,,...)
- ResultsRef as ResultsReference: (name: Pan-file, Descr. Panel-model created by Friendship, Results: Blob
name: ctl-file, Descr. n-Shallo control file, Results: Blob.
name: ptl-file, Descr. output file, Results: Blob.......)
Resistance

Figure 8.18.: ConfigurationItem of Resistance of the RoRo-Ship according to Scenario2

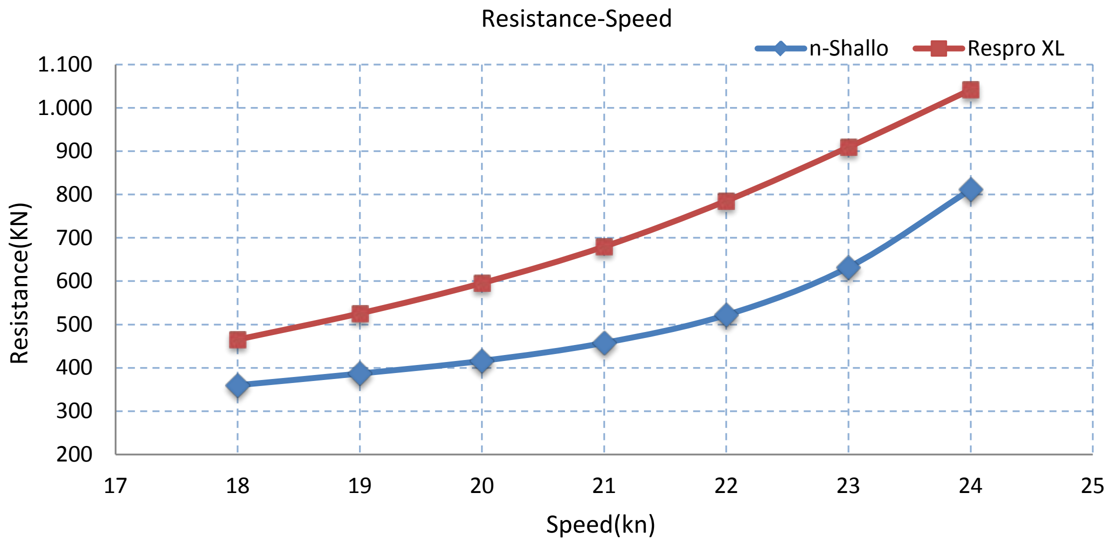


Figure 8.19.: Resistance-Speed Curve

After finishing the computation, the results are stored in the central database as an instance of the *Resistance*-object and the `Resistance`-configuration is defined. As it is shown in Figure 8.18, the stored results in the *Resistance*-object include the following data:

- The resistances depending on the corresponding speeds as a *Function2D*, for example (..., 386.705: 19,...). This allows for obtaining these results at any time from the central database and checking the resistance-speed diagram independently of the applied tools. The diagram depicted in Figure 8.19 shows the data read from the *Resistance*-object (blue line) comparing to those gained from the 'Respro XL'-tool (red line), to which an indicator is defined and stored within the *Power*-object (see section 8.1.2).
- The panel model, which is shown in Figure 8.17 is stored as a *ResultsReference*,
- A pointer to the store of the results of the computed RoRo-ship within the ν-Shallo tool is defined and saved.

8.1.7. Define Ship Component

An example of the definition of a ship component by means of the implemented UseCase: *Define Component* (see section 7.4.1.7) is introduced in the following. Accessing this UseCase by Designer4, who is responsible to define the ship components within the RoRo design project (see Table 8.1), is followed by the checking and informing of the availability of the *Compartmentation*, *Power* and *ComponentSpecification*. By taking into consideration the implementation of

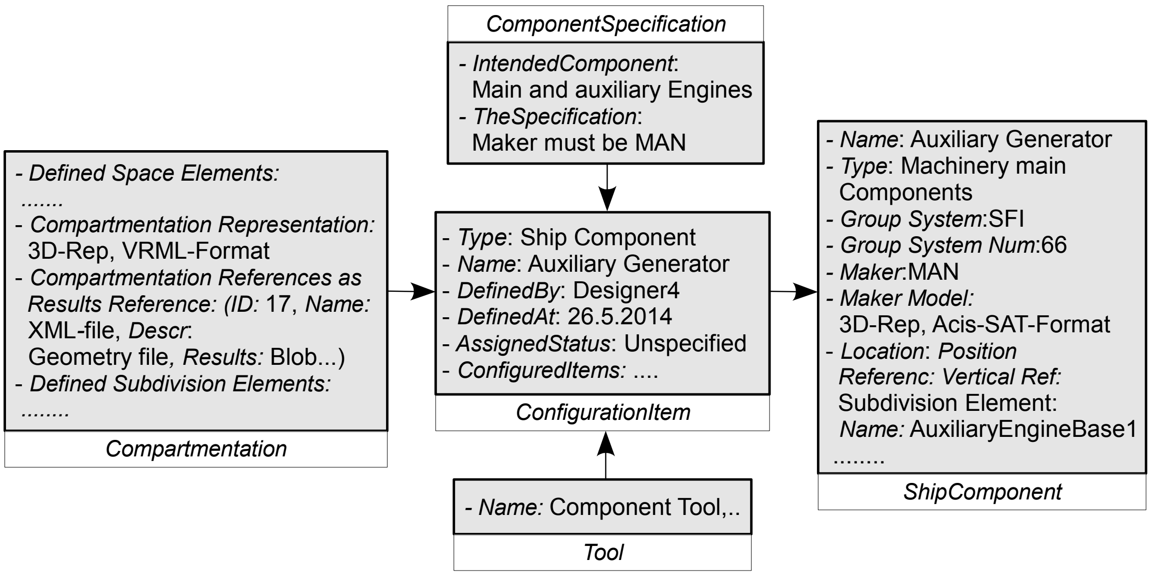

ComponentSpecification
- IntendedComponent: Main and auxiliary Engines
- TheSpecification: Maker must be MAN
- Defined Space Elements: ....
- Compartmentation Representation: 3D-Rep, VRML-Format
- Compartmentation References as Results Reference: (ID: 17, Name: XML-file, Descr: Geometry file, Results: Blob...)
- Defined Subdivision Elements: ....
Compartmentation
- Type: Ship Component
- Name: Auxiliary Generator
- DefinedBy: Designer4
- DefinedAt: 26.5.2014
- AssignedStatus: Unspecified
- ConfiguredItems: ....
ConfigurationItem
- Name: Component Tool,..
Tool
- Name: Auxiliary Generator
- Type: Machinery main Components
- Group System:SFI
- Group System Num:66
- Maker:MAN
- Maker Model: 3D-Rep, Acis-SAT-Format
- Location: Position Referenc: Vertical Ref: Subdivision Element: Name: AuxiliaryEngineBase1 ....
ShipComponent

Figure 8.20.: ConfigurationItem of Define 'Auxiliary Generator'-Component of the RoRo-Ship

this UseCase and in order to define the auxiliary generator component, the available *Compartmentation* and the *ComponentSpecification* are selected, whereas the available *Power* is ignored by Designer4. After the definition of the auxiliary generator by Designer4 by taking into ac-

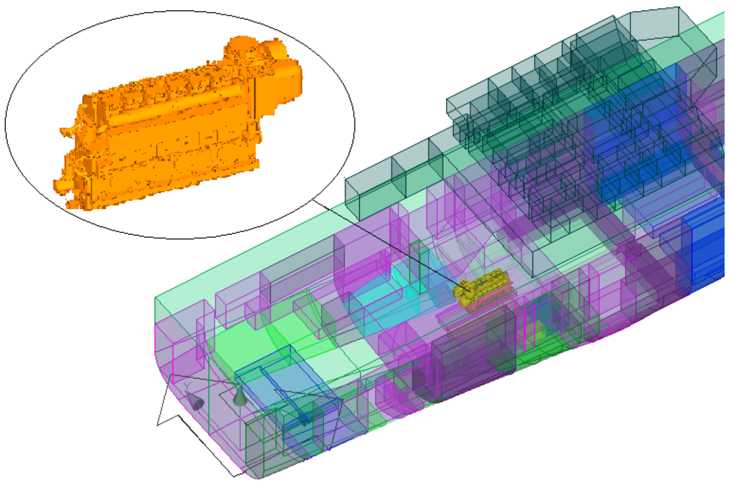


Figure 8.21.: Placement of the 'Auxiliary Generator' in the RoRo-Ship

count all obtained available data such as the specification (identified by setting up the RoRo project) that the component should be made by the MAN-company, the component-tool is used to locate and store the component in the central database as an instance of the *ShipComponent*-object. As it is shown in Figure 8.20 the *ShipComponent*-object includes several details of the defined component, such as *Name*: Auxiliary Generator, *Type*: Machinery main components, *Group System Number*: 66, the *Maker Model*: 3D-representation of the component model as it is obtained from the MAN-company (in SAT-format), etc. Moreover, the auxiliary generator, which is shown in Figure 8.21, is located on the position-reference: *SubdivisionElement* with the *Name*: AuxiliaryEngineBase1 by means of the *Placement*-object. As a result, in the case of shifting the referenced *SubdivisionElement* due to any reason, the auxiliary generator will be shifted automatically to the new position.

8.2. Management/Reporting

In the following sections the *Management/Reporting*-UseCases, introduced in section 7.4.2, are utilized to: *Create Design Solution*, *Update Status*, *Request Change*, *Approve Requested Change*, *Create Version* and to *Create Report* within the RoRo Ship project.

8.2.1. Create Design Solution

An example of a design solution, which can be created by means of the implemented UseCase: *Create Design Solution* is depicted in Figure 8.22. When the UseCase is accessed by Project

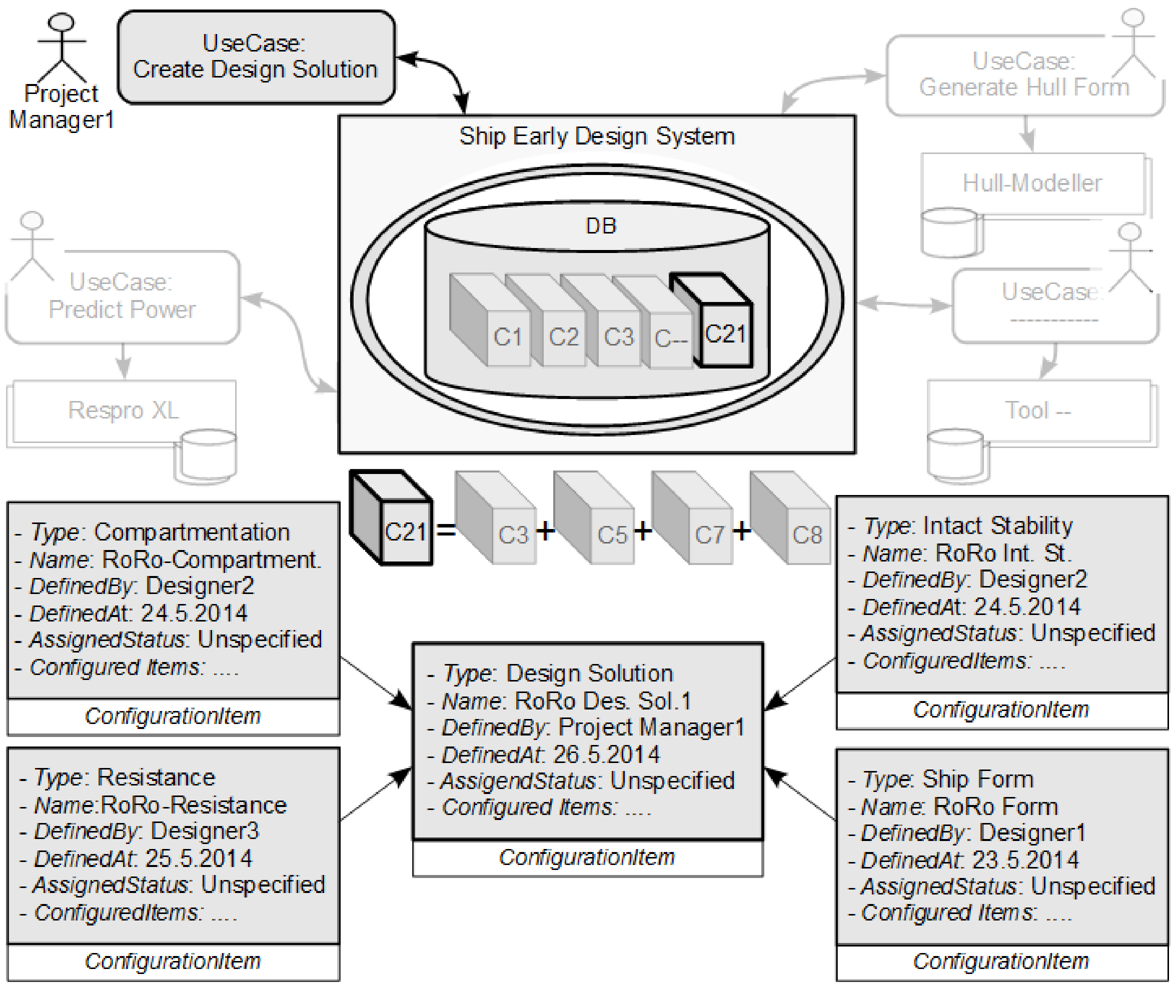


Figure 8.22.: Design Solution of the RoRo-Ship

Manager1, who is responsible for creating the design solutions within the RoRo design project (see Table 8.1), all defined configurations related to the RoRo-project, which are saved in the central database (they are depicted as C1, C2, etc. in Figure 8.22) and resulted from performing the design tasks, are checked in order to enable the Project Manager1 to define a design solution. The following four configurations, which are `Compartmentation`, `Resistance`, `Intact Stability` and `Ship Form` are selected by the Project Manager1 to create a design solution within the RoRo project. This is followed by an automated procedure: the selected configurations are applied to create the `Design Solution`-configuration (to which is referred as C21 in Figure 8.22) as an instance of the *ConfigurationItem*-object.

8.2.2. Update Status of Predicted Resistance

According to the task-responsibilities, the right of assigning (updating) the status of a defined configuration is authorized to the 'Head of Departments'. An example of updating a status is shown in Figure 8.23. Assuming that the status of `Resistance`-configuration (see Figure 8.18) is set to 'Preliminary' at a certain moment in the design process, the updating of this status in order to be taken into consideration by other system-users is performed by Head of Department C by means of the UseCase: *Update Status*. The redefined `Resistance`-configuration with the new *Status*: Preliminary is shown in Figure 8.23.

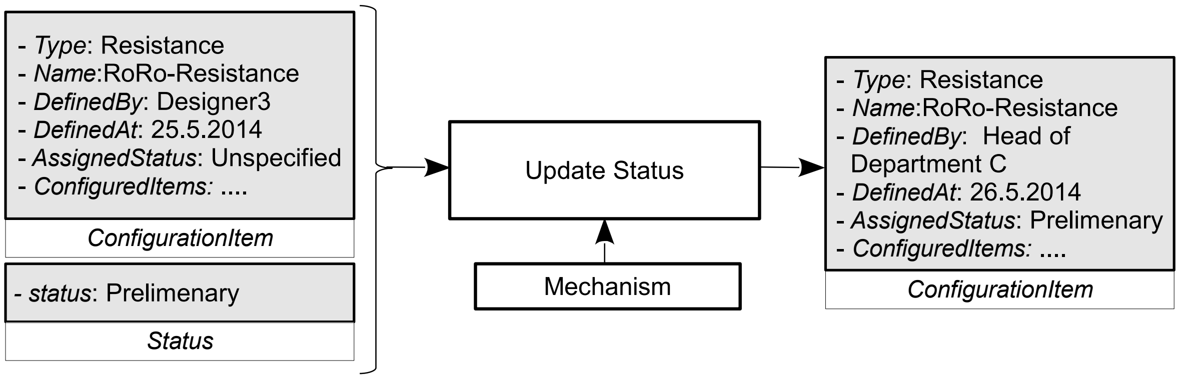


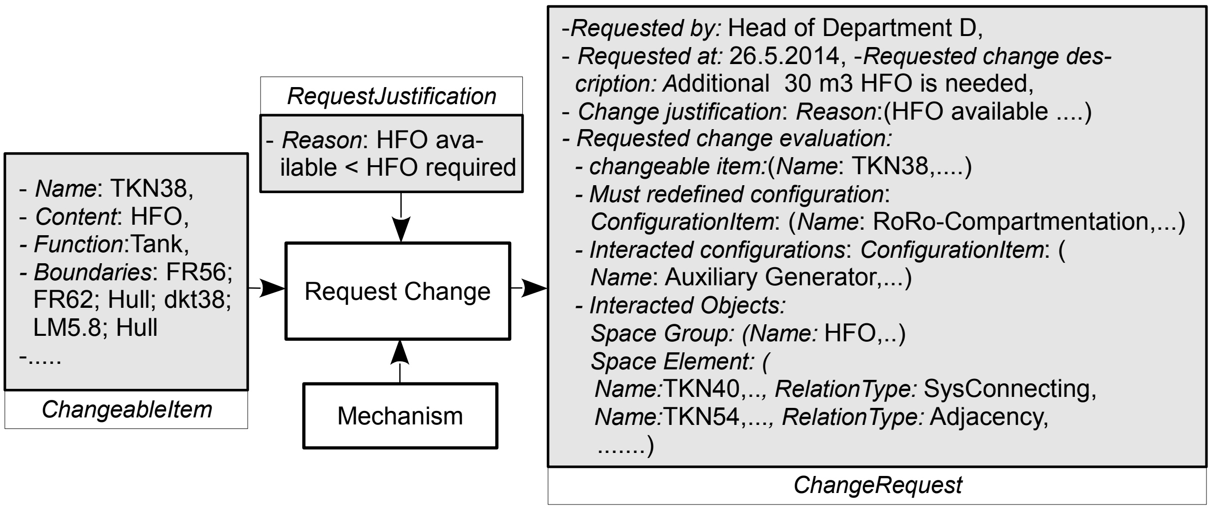
Figure 8.23.: Update Status of `Resistance`-Configuration

8.2.3. Control Changes Examples

8.2.3.1. Request Change

After performing the compartmentation task (see section 8.1.4), Designer4 and accordingly the Head of Department D figure out that the volume of the HFO-tanks is not compatible with the consumption of the main and auxiliary engines. Therefore, a change is requested by the Head of Department D by means of the UseCase: *Control Changes* (see the 'Check/Request'-choice in section 7.4.2.4). The Head of Department D has two alternatives to request the required change. The first is represented by selecting the *SpaceGroup* (*Name*: HFO) as the *ChangeableItem*. In this case, the checking of the change influences, which is executed automatically at both data and process levels, is subjected to the all HFO-tanks which are listed in Table 8.2. The second one, is to identify a specific HFO-tank to be requested to be changed. An example of this case is depicted in Figure 8.24. The *SpaceElement* (*Name*: TKN38) is selected and specified to be changed. As it is shown in the Figure, the saved requested change as an instance of the

ChangeRequest-object gives reasons for the required change and its influences. It is explicitly



- *Name*: TKN38,
- *Content*: HFO,
- *Function*:Tank,
- *Boundaries*: FR56; FR62; Hull; dkt38; LM5.8; Hull
-.....
ChangeableItem
RequestJustification
- *Reason*: HFO available < HFO required
Request Change
Mechanism
-*Requested by:* Head of Department D,
- *Requested at:* 26.5.2014, -*Requested change description:* Additional 30 m3 HFO is needed,
- *Change justification*: *Reason*:(HFO available ....)
- *Requested change evaluation:*
- *changeable item:*(*Name*: TKN38,....)
- *Must redefined configuration*: *ConfigurationItem*: (*Name*: RoRo-Compartmentation,...)
- *Interacted configurations*: *ConfigurationItem*: (*Name*: Auxiliary Generator,...)
- *Interacted Objects:*
Space Group: (Name: HFO,..)
Space Element: (
*Name:*TKN40,.., *RelationType:* SysConnecting,
*Name:*TKN54,..., *RelationType:* Adjacency,
.......)
ChangeRequest

Figure 8.24.: Request a Change: 'Space element must be larger'

shown, that the *ChangeRequest* is requested by the Head of Department D at 26.5.2014 due to the *Reason* that the available HFO-tanks are smaller than the required ones. Therefore, an additional 30 m3 HFO is required. Moreover, the evaluation of the TKN38-tank shows that the change of this tank is associated with the following influences (identified automatically). Firstly, the tank must be re-created by the compartmentation task and accordingly the `Compartmentation`-configuration must be redefined. Secondly, the `Auxiliary Generator`-configuration, which is defined previously, can be influenced. This is due to the interaction between the two tasks: *Create Compartmentation* and *Define Component*. Thirdly, the *SpaceGroup* to which the TKN38-tank is related can be affected. Fourthly, *SpaceElement*(s) which are interacting or related to the TKN38 in different *RelationType*(s) such as SysConnecting, Adjacency, etc. are listed as *Interacted Objects*.

8.2.3.2. Approve Requested Change

According to the task-responsibilities listed in Table 8.1, the right to approve or reject a requested change is assigned to Project Manager1. This does not conflict with the fact that within many shipyards the decision to approve or reject a change is made through a meeting between the project manager and heads of the departments, who participate in a design project. But rather it means that Project Manager1 is the user who is responsible for controlling the decision-making and accordingly in the implemented UseCase: *Control Changes* (see the 'Approve/Reject'-choice in the section 7.4.2.4). After accessing the UseCase by the Project Manager1 and selecting the *ChangeRequest*, all previously details associated with the requested change are provided automatically. Assuming that Project manager1 depending on the offered data, decides to approve the requested change. As it is shown in Figure 8.25, submitting of the approve-decision is followed by: Firstly, saving of the approve-change details (such as *Controlled by*: Project Manager1, *Controlled at*: 27.5.2014, etc.) within the central database as an instance of the *ChangeControl*-object. Secondly, a message is released, which includes the decision made and other key details, to the system-users who are concerned with the decision. In this context, a replay-message is released to the Head of Department D as well as to Designer4 and a message, that a new compartmentation is demanded, is released to Head of Department B as well as to Designer2.

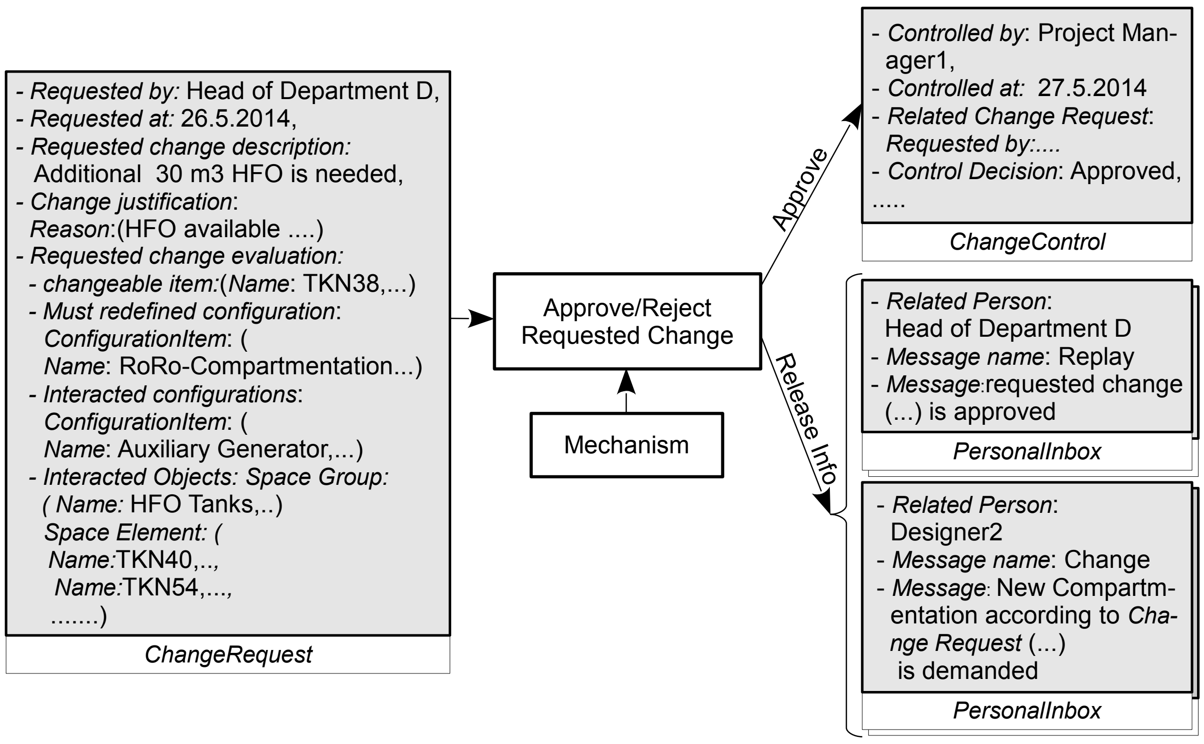

- Requested by: Head of Department D,
- Requested at: 26.5.2014,
- Requested change description: Additional 30 m3 HFO is needed,
- Change justification: Reason:(HFO available ....)
- Requested change evaluation:
- changeable item:(Name: TKN38,...)
- Must redefined configuration: ConfigurationItem: (Name: RoRo-Compartmentation...)
- Interacted configurations: ConfigurationItem: (Name: Auxiliary Generator,...)
- Interacted Objects: Space Group: (Name: HFO Tanks,..) Space Element: (Name:TKN40,.., Name:TKN54,..., ...)
ChangeRequest
Approve/Reject Requested Change
Mechanism
Approve
Release Info
- Controlled by: Project Manager1,
- Controlled at: 27.5.2014
- Related Change Request: Requested by:....
- Control Decision: Approved, ...
ChangeControl
- Related Person: Head of Department D
- Message name: Replay
- Message:requested change (...) is approved
PersonalInbox
- Related Person: Designer2
- Message name: Change
- Message: New Compartmentation according to Change Request (...) is demanded
PersonalInbox

Figure 8.25.: Approve Requested 'Space element must be larger' Change

8.2.4. Create Version

After approving the *ChangeRequest*, the new comaprtmentation is created by Designer2 by taking into consideration the formulated demands that additional 30 m3 HFO should be carried by TKN38-tank. This is conducted by exploiting the benefits offered by the compartmentation tool of the UseCase: *Create Compartmentation*. The location of the aft boundary (transverse bulkhead) of the *SpaceElement*: TKN38 is transformed from position 'FR56' to position 'FR54' by changing the value in the compartmentation definition script. In Figure 8.27, the transformed bulkhead in addition to the resulted influences are depicted. In Table 8.4, the HFO-tanks of the new compartmentation are listed (see the change of the data related to the TKN38 comparing to Table 8.2). These results in addition to other results are saved in the central database and a new `Compartmentation`-configuration is defined and saved in the same procedure as it has been described in the first execution. An example of versioning within the RoRo-project is

Name	*Volume(m3)*	*Weight(t)*	*LCB(m)*	*TCB(m)*	*VCB(m)*	*Sur. Area(m2)*
TKN30	64.5	59.3	168.7	0.0	5.0	130.7
TKN31	224.3	206.4	55.8	-5.0	1.2	293.6
TKN32	224.3	206.4	55.8	5.0	1.2	293.6
TKN33	407.5	374.9	160.6	-2.0	5.0	403.7
TKN34	40.2	37.0	47.2	-7.5	3.0	73.2
TKN35	407.5	374.9	160.6	2.0	5.0	403.7
TKN36	65.7	60.4	45.9	-9.64	2.7	127.6
TKN37	55.2	50.8	47.5	-5.1	0.9	109.0
TKN38	101.4	93.3	39.1	-8.4	3.0	136.7
Sum	1590.6	1463.3	113.9	-1.2	3.5	1971.6

Table 8.4.: Spaces related to '*SpaceGroup*(*Name*: HFO)'

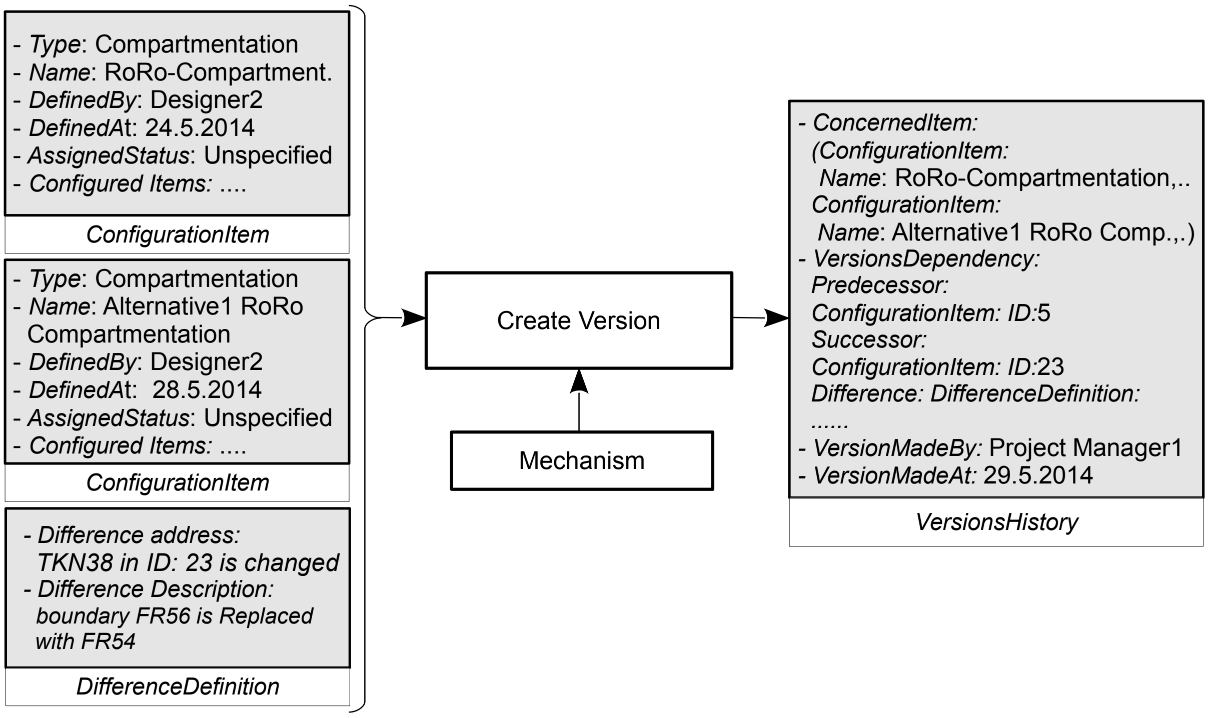

- Type: Compartmentation
- Name: RoRo-Compartment.
- DefinedBy: Designer2
- DefinedAt: 24.5.2014
- AssignedStatus: Unspecified
- Configured Items: ....
ConfigurationItem
- Type: Compartmentation
- Name: Alternative1 RoRo Compartmentation
- DefinedBy: Designer2
- DefinedAt: 28.5.2014
- AssignedStatus: Unspecified
- Configured Items: ....
ConfigurationItem
- Difference address: TKN38 in ID: 23 is changed
- Difference Description: boundary FR56 is Replaced with FR54
DifferenceDefinition
Create Version
Mechanism
- ConcernedItem: (ConfigurationItem: Name: RoRo-Compartmentation,.. ConfigurationItem: Name: Alternative1 RoRo Comp.,.)
- VersionsDependency: Predecessor: ConfigurationItem: ID:5 Successor: ConfigurationItem: ID:23 Difference: DifferenceDefinition: ......
- VersionMadeBy: Project Manager1
- VersionMadeAt: 29.5.2014
VersionsHistory

Figure 8.26.: Versioning of ‘Compartmentation’-Configuration

depicted in Figure 8.26. It is performed by Project Manager1 by means of the UseCase: *Create Version*. The Figure shows, that both defined `Compartmentation`-configurations within the RoRo project are versioned and saved in the central database depending on each other as an instance of the *VersionsHistory*-object. The former configuration is specified as *Predecessor* and the latter is specified as *Successor* in addition to the definition of the difference between them.

8.2.5. Create Report

Two examples of reports, which are created by the UseCase: *Create Report* are discussed in the following. After accessing the UseCase by Designer4, `Compartmentation`-configuration (*Name*: RoRo Compartmentation), which is defined previously is selected as the item to be reported. In Figure 8.29, the resulting PDF-report is depicted. As it is shown the 3D-representation of the compartmentation as well as the additional data are combined in a same PDF-report by exploiting the benefits of the applied techniques.
During the design process of the RoRo-ship, the achievement level of the design project can be reported by means of selecting of *DesignProject* as the item to be reported at any time. This is made by Project Manager1 and the resulted report is depicted in Figure 8.28. As it is shown in the Figure, the achievement level can be easily figured out from the represented data. The performed design tasks within the RoRo-design project, their responsible users, as well as their states at the moment of creation of the report are depicted. Therein, the following colors are used:

- Green: the status of at least one of the defined and saved task-configurations within the central database is ‘Approved’.
- Blue: at least one of the task-configurations is not ‘Unspecified’.
- Yellow: all task-configurations are ‘Unspecified’.
- Red: no task-configuration is available within the central database.

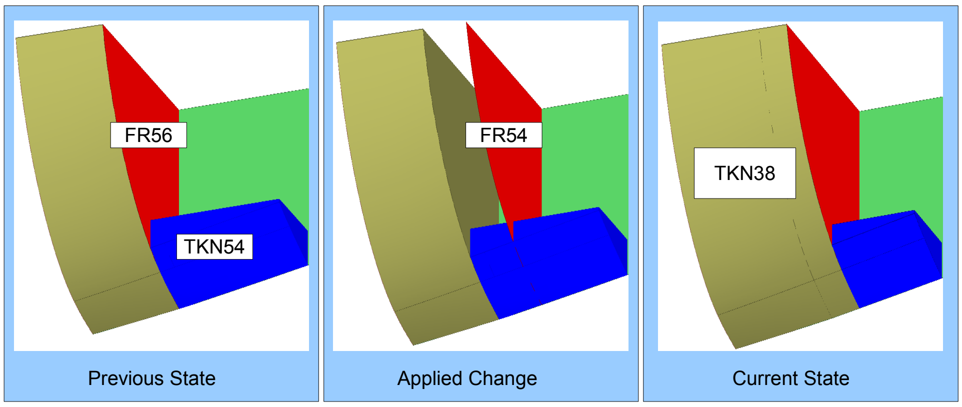
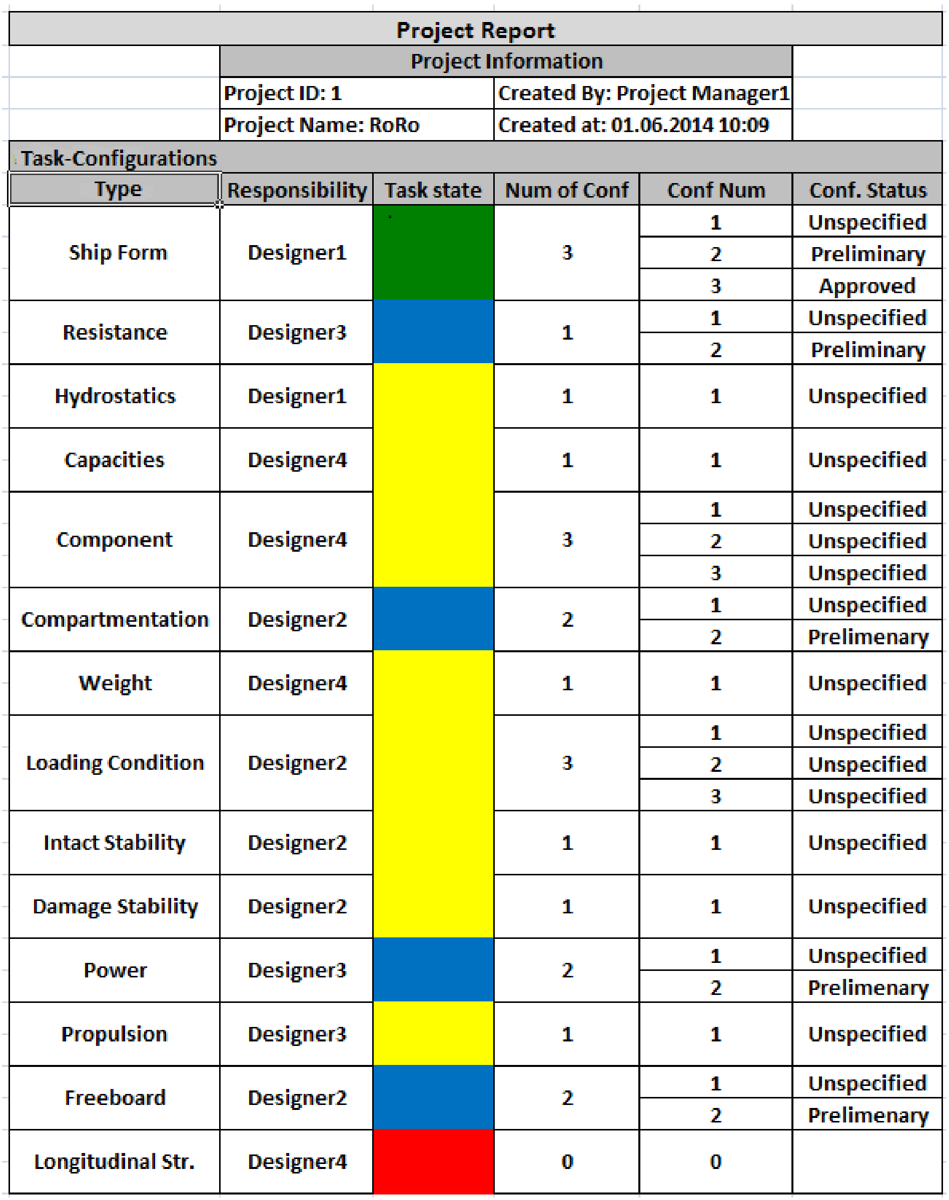


Figure 8.27.: Applied Change to TKN38-tank



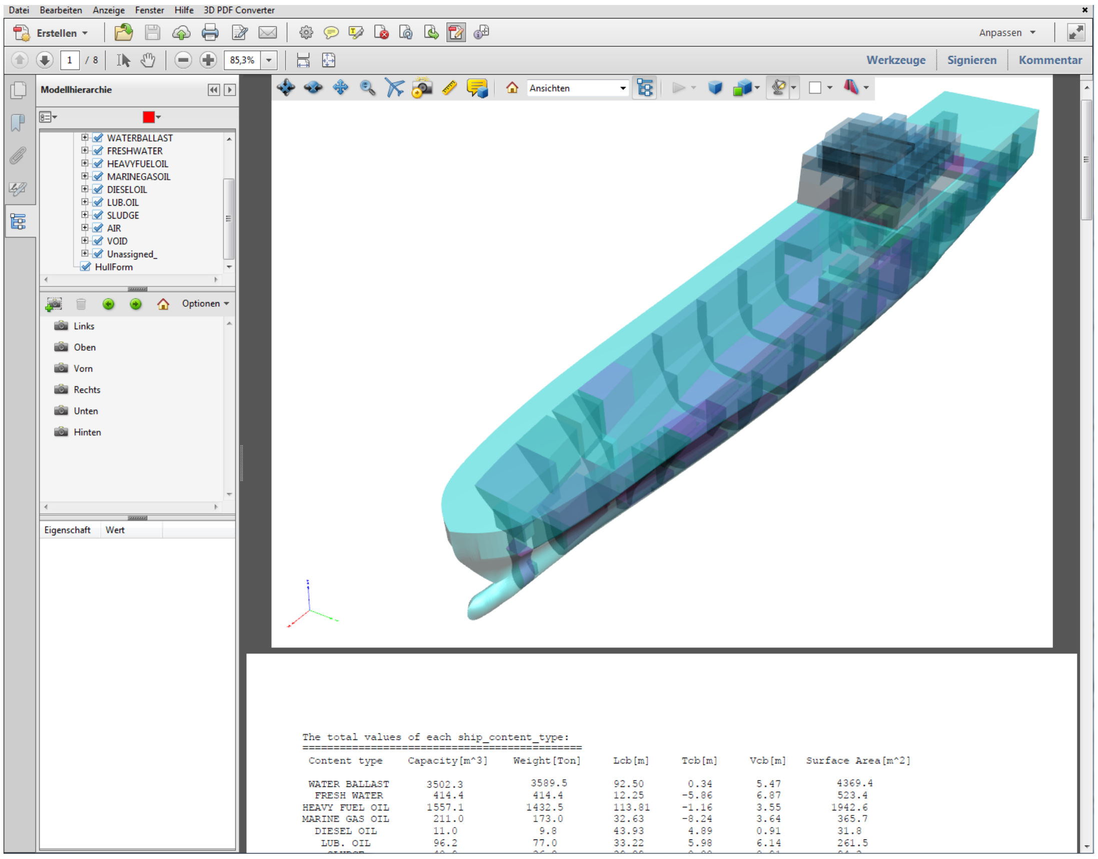
Project Report

	Project Information		
	Project ID: 1	**Created By: Project Manager1**	
	Project Name: RoRo	**Created at: 01.06.2014 10:09**	

Task-Configurations

Type	Responsibility	Task state	Num of Conf	Conf Num	Conf. Status
Ship Form	Designer1	🟩	3	1	Unspecified
				2	Preliminary
				3	Approved
Resistance	Designer3	🟦	1	1	Unspecified
				2	Preliminary
Hydrostatics	Designer1	🟨	1	1	Unspecified
Capacities	Designer4	🟨	1	1	Unspecified
Component	Designer4	🟨	3	1	Unspecified
				2	Unspecified
				3	Unspecified
Compartmentation	Designer2	🟦	2	1	Unspecified
				2	Prelimenary
Weight	Designer4	🟨	1	1	Unspecified
Loading Condition	Designer2	🟨	3	1	Unspecified
				2	Unspecified
				3	Unspecified
Intact Stability	Designer2	🟨	1	1	Unspecified
Damage Stability	Designer2	🟨	1	1	Unspecified
Power	Designer3	🟦	2	1	Unspecified
				2	Prelimenary
Propulsion	Designer3	🟨	1	1	Unspecified
Freeboard	Designer2	🟦	2	1	Unspecified
				2	Prelimenary
Longitudinal Str.	Designer4	🟥	0	0	

Figure 8.28.: RoRo Design Project Report



```
The total values of each ship_content_type:
=============================================
 Content type    Capacity[m^3]   Weight[Ton]     Lcb[m]     Tcb[m]     Vcb[m]   Surface Area[m^2]

 WATER BALLAST      3502.3         3589.5        92.50       0.34       5.47        4369.4
  FRESH WATER        414.4          414.4        12.25      -5.86       6.87         523.4
HEAVY FUEL OIL      1557.1         1432.5       113.81      -1.16       3.55        1942.6
MARINE GAS OIL       211.0          173.0        32.63      -8.24       3.64         365.7
  DIESEL OIL          11.0            9.8        43.93       4.89       0.91          31.8
   LUB. OIL           96.2           77.0        33.22       5.98       6.14         261.5
```

Figure 8.29.: Report of the Configuration: ‘RoRo Compartmentation’, shown in Adobe Acrobat XI

9. Summery and Outlook

The global maritime market is characterized by an increased competition between shipyards. This imposes shipyards to permanently develop their design methodologies in order to efficiently achieve the design studies especially those related to the early design phase. This can be attributed to two reasons: the first is related to the great influence of the early design studies on the entire design process, and the latter is connected to the fact that these studies-results are the basis on which the offer in the tendering process is built.
Therefore, the efforts of this research are subjected to the development of an approach to enhance the early design phase. In order to efficiently achieve this target, an analysis of the early design phase regarding its importance, complexity and its research-needs comparing to the remained design stages, is discussed in detail in chapters 1 and 2. Thereby, the aspects, which can be considered the key development aspects within the early phase such as the decision making, the data-reuse, data exchange and design constraints, are introduced regarding their needs and drawbacks with respect to the current state of the ship early design process. In order to enhance the analyze and complete the picture of the current state of the ship design, some of the international standards, which are concerned with the structuring of the data related to the ship, are reviewed. With the review of many published approaches, the objective target of this research is further refined in so far as to concentrate on the enhancement of the following interactions with respect to the early design phase: *data-data*, *data-process* and *process-process*.
Based on the definition of the enterprise, which is compatible with the domain of system design engineering, several concepts regarding the modeling as well as the management of an enterprise are introduced and discussed in chapter 3. Here, methods of modeling data and process perspectives of an enterprise are introduced. Thereby, a special attention is paid to integration concepts within an enterprise. The core enterprise management concepts, which are *Configuration Management*, *Change Management*, *Version Management*, *Activity Management*, *Rights Management* and *Collaboration Management*, as they are needed in the following chapters, are introduced with regard to their main characteristics.
Depending on these theoretical basics, an *Information Model* for the ship in the early design phase is developed and introduced in chapter 4. It represents the data perspective of the *Ship Early Design Enterprise Model* discussed in chapter 6. Thereby, the information, which are modelled, reflect three areas: *Ship Product*-information, *Support*-information and *management*-information. These three types of information are organized in many packages according to the function or concept which they serve. Each of these packages includes one or more *information object*s. These objects are interrelated to each other. The associations are modelled with two main objectives in mind, to insure the *data-consistency* concept and to guarantee and support the *data-data* interaction concept.
With the development of the activity management package, the notion *Predefined Activitiy* is introduced. It represents the major concept used throughout the following chapters to model the tasks related to the early ship design phase. Each of the predefined activities is further subdivided in *Activity Partition*(s). Activity partitions are interacting with the information objects by means of *Activity Partition Item*(s). In this context, the activity management package represents the foundation, which guarantees the reflection of the *data-process* as well as *process-process* interaction concepts within the ship information model.

The process perspective of the enterprise modeling approach represented by the ship's early design process is discussed in chapter 5. The tasks related to the early design phase are analyzed and formulated according to the *ICOM*-concept regardless of the tools used to perform these tasks. This analysis is used to represent the early design tasks as *Predefined Activities*. Each of these *Predefined Activities* is finalized by the definition of a *ConfigurationItem*, which reflects the *Configuration Management*-concept and represents a combination of all *Activity Partition Item*s interacting with the *Activity Partition*(s) related to a specific early design task represented as a *Predefined Activity*. Thereby, the *Status*-concept is integrated supporting the combined items (represented by the *ConfigurationItem*) with the decision dimension. Moreover, *Predefined Activities*, which reflect the process perspective of the developed management models, are introduced. For example, three *Predefined Activities*, which are *Check Change Influences*, *Request Change* and *Approve/Reject Requested Change*, are showcased to discuss the *Change Management*-capabilities.
In order to illustrate the benefits of the developed models, they are translated in the information processing system level by means of the following techniques: Python as *Object Oriented Programming (OOP)*, SQLite as *Relational Database Management System (RDBMS)* and SQLAlchemy as *Object Relational Mapping (ORM)*, see chapter 7. Thereby, the developed *Ship Information Model* is set up in a database and the *Predefined Activities* are implemented in *UseCase*s. These implementations are integrated in a *Ship Early Design System*. During the implementations the fact of the variety of scenarios, which can be used to perform an early design task, is taken into consideration. With realizing this fact, three types of the interactions with external tools, such as AVEVA ID-system, ν-Shallo tool, etc., are introduced. Moreover, tools for the 3D-Compartmentation and the creation of hull-forms based on available ones are implemented. The developed system endorse the dynamical and the collaborative nature of the ship's early design phase by accounting for the numerous interactions between early design tasks and by ensuring of the key aspect, which can be summarized as follows: *Providing the key persons with the key data at the right time.* Furthermore, the consistency of the multiple representations of a design object is thereby insured.
In order to evaluate the efficiency of the developed *Ship Early Design System*, it is integrated in a shipyard design workbench. The developed system is used to perform studies related to the early design of a RoRo-ship and the results are represented in chapter 8. Thereby, examples of the system-advantages, such as *Create Design Solution*, *Create Version* are introduced.
The prototype implementation of the *Ship Early Design System* offers a means to increase the efficiency of performing the early design studies by making use of the functionalities provided by different tools. Thereby the developed system plays a central role in management of the data-transfer between applied tools by providing a *central database* (data management component). The interactions between the system and the applied tools can be implemented in different ways depending on the tool-characteristics. It should be noted that the applied tools within this thesis are just example tools of an assumed shipyard to reflect the applicable nature of the developed approach, they can be replaced by any other tools in order to take the diverse software-infrastructures of shipyards into consideration.
Moreover, the developed system accounts for the collaborative nature of the early design phase by the efficient control of the interactions between different design tasks and accordingly between the responsible design team members. The performing of any design task represented as a *Predefined Activity* is started by an automatic procedure to detect the available data within the *central database*, which are required to perform the task. This is followed by automatically informing the responsible user about the achievable task-scenarios at that point of time. In this context, a periodical checking of the available data in the *central database* (such as every hour) and accordingly informing the responsible users about the achievable tasks can be developed.

Furthermore, performing a design task is always finished by the definition of a *ConfigurationItem* and by informing the concerned users about the new data available.
Finally, the introduced *Enterprise Approach* offers a means to enhance the integration dimension of the early design phase with the following phases. Here, a development of the *Ship Information Model* to consider more and more details is possible. This can be used, for example, to account for a shipyard perspective regarding the design requirements. Expanding the information model and accordingly the set-up database to play a central role through the entire design life-cycle could be a solution for many drawbacks regarding the design process as of today. Thereby, adding new design tasks with respect to the adopted *Predefined Activity* concept and accordingly implementations of new *UseCase*s to cover the new tasks (predefined activities) can be achieved.

Bibliography

[1] Bray t. et al., Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation 04 February 2004, http://www.w3.org/tr/2004/rec-xml-20040204/.

[2] SFI group system. http://www.spectec.dk/files/billeder/pdf/sfi2001.

[3] *Systems Engineering Fundamentals.* Supplementary Text Prepared by the Defence Acquisition University Press Fort Belvoir, Virginia, 2001.

[4] *ISO 10303-42:2003, Industrial automation systems and integration – Product data representation and exchange – Part 42: Integrated generic resource: Geometric and topological representation.* 2003.

[5] *ISO 10303-218, Industrial automation systems and integratio,Product data representation and exchange,Part 218: Application Protocol: Ship structures.* 2004.

[6] *ISO 10303-41:2005, Industrial automation systems and integration – Product data representation and exchange – Part 41: Integrated generic resource: Fundamentals of product description and support.* 2005.

[7] *ISO 19439:2006: Enterprise integration – Framework for enterprise modelling.* 2006.

[8] *ISO 10007:2003: Quality management systems – Guidelines for configuration management.* 2009.

[9] *ISO 10303-216, Industrial automation systems and integration,Product data representation and exchange,Part 214: Application Protocol: Core data for automotive mechanical design processes.* 2010.

[10] *ISO/IEC TR 18018:2010: Information technology – Systems and software engineering – Guide for configuration management tool capabilities.* 2010.

[11] *ISO 10303-44: 2000: Industrial automation systems and integration – Product data representation and exchange – Part 44: Integrated generic resource: Product structure configuration.* 2012.

[12] *ISO/IEC 19505, Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 1: Infrastructure.* 2012.

[13] *ISO/IEC 19505, Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 2: Superstructure.* 2012.

[14] PyFPDF, A Library for PDF-File generation for Python. Available at: https://pypi.python.org/pypi/fpdf.py, 2013.

[15] Python programming language , Home Page: http://www.python.org/, 2013.

[16] Rhinoceros3D, 3D-Modelling Software, Home Page: http://www.rhino3d.com/, 2013.

[17] SFI codification system, http://www.spectec.net/en/maritime/sfi-codification-system, 2013.

[18] SQLAlchemy, The Python SQL Toolkit and Object Relational Mapper. available at:

http://www.sqlalchemy.org/, 2013.

[19] SQLite, Database Management System. available at: http://www.sqlite.org/, 2013.

[20] Tetra 4D. Home Page: http://www.tetra4d.com/, 2013.

[21] Visual Paradigm for UML - Software Design Tool for Software Developement. available at: http://www.visual-paradigm.com/, 2013.

[22] XlsxWriter: A python module for creating Excel files, available at: https://pypi.python.org/pypi/xlsxwriter, 2013.

[23] Atlantec Enterprise solutions , Home Page: http://www2.atlantec-es.com/, 2014.

[24] Friendship-Systems , Home Page: https://www.caeses.com/, 2014.

[25] IACS-International Association of Classification Societies, Home Page: http://www.iacs.org.uk/, 2014.

[26] IMO-International Maritime Organization, Home Page: http://www.imo.org/, 2014.

[27] NAPA-System , Home Page: http://www.napa.fi/, 2014.

[28] POSEIDON-Tool , Home Page: http://www.gl-group.com/en/gltools/poseidon.php, 2014.

[29] QinetiQ GRC. Paramarine software, Home Page: http://www.grc.qinetiq.com/, 2014.

[30] SHIPFLOW-Tool , Home Page: http://www.flowtech.se/, 2014.

[31] *Merchant Web Services API, Customer Information Manager (CIM), SOAP Guide.* Authorize.Net, September 2014.

[32] Anis Abdmouleh, Michel Spadoni, and François Vernadat. Distributed client/server architecture for cimosa-based enterprise components. *Computers in industry*, 55(3):239–253, 2004.

[33] Andrea L Ames, David R Nadeau, and John L Moreland. VRML Sourcebook 2.0, 1997.

[34] D. Andrews. *The Use Of Simulation In Preliminary Ship Design.* Proceedings of the International Conference on Computer Applications in Shipbuilding, Busan. Korea, 2005.

[35] D Andrews, A Papanikolaou, S. Erichsen, and S Vasudevan. State of the art report on design methodology. In *Proceedings of the International Marine Design Conference, Trondheim*, volume 2, pages 537–576, 2009.

[36] D. Andrews and R. Pawling. *The Application Of Computer Aided Graphics To Prelimenary Ship Design.* Proceedings of the International Marine Design Conference, Ann Arbor, MI, 2006.

[37] D. Andrews and R. Pawling. *Research Into The Use Of Computer Aided Graphics In Preliminary Ship Design.* Proceedings of the International Conference on Computer Applications in Shipbuilding, Portsmouth. UK, 2007.

[38] David Andrews. Creative ship design. *Trans RINA*, 123:447–471, 1981.

[39] L. Baarman. *3D Product Model For Management Of Design And Lifecycle.* Proceedings of the International Marine Design Conference, Glasgow, 2012.

[40] Bryan Barrass. *Ship design and performance for masters and mates.* Butterworth-Heinemann, 2004.

[41] Roland Felkai Arndt Beiderwieden. *Projektmanagement für technische Projekte.* Springer, 2013.

[42] Volker Bertram and H Schneekluth. *Ship design for efficiency and economy.* Butterworth-Heinemann, 1998.

[43] Robert Bronsart. Ship design lectures, University of Rostock, 2013.

[44] William D Brosey, Richard E Neal, and Douglas F Marks. Grand challenges of enterprise integration. In *Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on*, volume 2, pages 221–227. IEEE, 2001.

[45] W Bühr, H Keil, and S Krüger. Rechnereinsatz im Projekt. *Jahrbuch der Schiffbau Technischen Gesellschaft*, pages 353–363, 1988.

[46] Rainer Burkhardt. *UML-Unified modelling language [Medienkombination]·: objektorientierte Modellierung für die Praxis..... Buch.* Addison-Wesley-Longman, 1997.

[47] C Cabos, D Jaramillo, G Stadie-Frohbös, P Renard, M Ventura, and B Dumas. Condition assessment scheme for ship hull maintenance. In *International Conference on Computer Applications and Information Technology in the Marine Industries*, 2008.

[48] A. Cebollero and J. Quiroga. Space management in ship early design. In *Proceedings of the International Conference on Computer Applications in Shipbuilding, Portsmouth, UK*, 2007.

[49] A. Cebollero and F. Zurdo. Integrated management of ship compartments. In *Proceedings of the International Conference on Computer Applications in Shipbuilding, China*, 2009.

[50] Fang Chen, JF Nunamaker Jr, NC Romano, and Robert O Briggs. A collaborative project management architecture. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on.* IEEE, 2003.

[51] Yuh-Min Chen, Wei-Shin Shir, and Chung-Yen Shen. Distributed engineering change management for allied concurrent engineering. *International Journal of Computer Integrated Manufacturing*, 15(2):127–151, 2002.

[52] HSVA Computational Fluid Dynamics. n-SHALLO.

[53] Anthony S Daniels, F Tahmasi, and David J Singer. Intelligent Ship Arrangement (ISA) methodology improvements and capability enhancements. In *Proceedings of International Marine Design Conference, Trondheim, Norway*, 2009.

[54] Christiane Doese. *Datenmodellierung im Schiffsentwurf und Prozeßmodellierung als Grundlage für ein Concurrent Engineering.* Inst. für Schiffbau, 1997.

[55] Hans-Erik Eriksson and Magnus Penker. *Business modeling with UML.* Wiley Chichester, 2000.

[56] S. Erikstad. Efficient exploration of existing corporate knowledge in conceptual ship design. In *Proceedings of the International Conference on Computer and IT Applicatios in the Maritime Industries (COMPIT), Cortona*, 2007.

[57] Ben Kassel et al. An alternate approch to the exchange of ship product model data. *Journal of Ship Production*, Vol.24, 2008.

[58] D. Andrews et al. *Combining The Building Block and Library Based Approaches To Improve Exploration During Initial Design.* Proceedings of the International Conference on Computer and IT Applicatios in the Maritime Industries, Gubbio, 2010.

[59] D. Andrews et al. State of the art report. In *Proceedings of the International Marine Design Conference, Glasgow*, volume 1, 2012.

[60] D. Andrews et al. The true nature of ship concept design and what it means for the future development of casd. In *Proceedings of the International Conference on Computer and IT Applicatios in the Maritime Industries (COMPIT), Cortona*, 2013.

[61] D. Jaramillo et al. Efficient data management for hull condition assessment. In *Proceedings of the International Conference on Computer Applications in Shipbuilding, Busan*, 2005.

[62] D. Jaramillo et al. Specification of hcm (hull condition data model). *Report D-1-3-1*, 2007.

[63] Escola Politecnica et al. Integrating project management, concurrent engineering, and engineering design to improve ship design. In *Third International Conference on Production Research*, 2006.

[64] G. Willmes et al. Communication processes in shipbuilding based on intelligent 3d pdf documents. In *Proceedings of the International Conference on Computer Applications in Shipbuilding, Italy*, 2011.

[65] I. Kuutti et al. Efficient integration of 3d design with engineering at the early design stages. In *Proceedings of the International Conference on Computer Applications in Shipbuilding, Italy*, 2011.

[66] J. Charles et al. Advantages of software integration from initial design through to production design. In *Proceedings of the International Conference on Computer Applications in Shipbuilding, Sweden*, 2002.

[67] M. C. Parker et al. Intelligent ship arrangements: Reseeding scheme development and effectiveness. In *Proceedings of the International Conference on Computer Applications in Shipbuilding, Italy*, 2011.

[68] M. Gaspar et al. Handling complexity aspects in conceptual ship design. In *Proceedings of the International Marine Design Conference, Glasgow*, volume 1, 2012.

[69] R. Pawling et al. Just how valid is the ship design spiral given the existence of cliffs and plateaux? In *Proceedings of the International Marine Design Conference, Glasgow*, volume 1, pages 317–338, 2012.

[70] T. Nakamatsu et al. Total design optimization in initial design stage based on single 3d ship model. In *Proceedings of the International Conference on Computer Applications in Shipbuilding, Italy*, 2011.

[71] U. Eriksson et al. *Space Management As An Enabling Technique For The Development Of Ship's Arrangements Based On A Common Reference Model.* Proceedings of the International Conference on Computer Applications in Shipbuilding, China, 2009.

[72] J Harvey Evans. Basic design concepts. *Journal of the American Society for Naval Engineers*, 71(4):671–678, 1959.

[73] David F Ferraiolo and D Richard Kuhn. Role-based access controls. *arXiv preprint arXiv:0903.2171*, 2009.

[74] Martin Fowler. *Patterns für Enterprise Application-Architekturen.* mitp-Verlag, 2003.

[75] Yacine Gasmi, Ahmad-Reza Sadeghi, Patrick Stewin, Martin Unger, Marcel Winandy, Rani Husseiki, and Christian Stüble. Flexible and secure enterprise rights management based on trusted virtual domains. In *Proceedings of the 3rd ACM workshop on Scalable trusted computing*, pages 71–80. ACM, 2008.

[76] Frank Geisler. *Geisler, Datenbanken. Grundlagen und Design; PR.* Hüthig Jehle Rehm, 2009.

[77] Justin W Gillespie, Anthony S Daniels, and David J Singer. Decomposing ship arrangements using signed networks. In *Proceedings of the Fifteenth International Conference on Computer Applications in Shipbuilding (ICCAS 2011). Trieste, Italy*, 2011.

[78] Justin W Gillespie, Anthony S Daniels, and David J Singer. Approaching ship arrangements from a non-spatial point of view using network theory. In *Proceedings of the 11th International Marine Design Conference*, 2012.

[79] Anurag Goel. Enterprise integration EAI vs. SOA vs. ESB. *Infosys Technologies White Paper*, 87, 2006.

[80] Michael Goul and Karen Corral. Enterprise model management and next generation decision support. *Decision Support Systems*, 43(3):915–932, 2007.

[81] Marcus Grande. *100 Minuten für Konfigurationsmanagement: kompaktes Wissen nicht nur für Projektleiter und Entwickler*. Springer, 2012.

[82] HE Guldhammer and Svend Aage Harvald. *Ship resistance: Effect of form and principal dimensions*. Akademisk Forlag, 1974.

[83] Terry Halpin and Tony Morgan. *Information modeling and relational databases*. Morgan Kaufmann, 2010.

[84] Stefan Harries, Fabian Tillig, Marc Wilken, and George Zaraphonitis. An integrated approach for simulation in the early ship design of a tanker. *10th COMPIT*, 2011.

[85] Beate Hartmann. Enterprise architecture as an instrument of strategic control. In *Enterprise Modelling and Information Systems Architectures (EMISA)*, 2011.

[86] Anne Mette Jonassen Hass. *Configuration management principles and practice*. Addison-Wesley Longman Publishing Co., Inc., 2003.

[87] Sebastian Herden. *Model-Driven-Configuration-Management: ein modellgetriebener Ansatz für das Konfigurationsmanagement von IT-Systemlandschaften*. Springer, 2012.

[88] Steve Hoberman. *Data modeling made simple*. Technics Publications, 2009.

[89] J Holtrop and GG Mennen. An approximate power prediction method. 1982.

[90] ANSI INCITS. Incits 359-2004, american national standard for information technology, role based access control, 2004.

[91] Initial Graphics Exchange Specification (IGES) 5.3. *ANS US PRO/IPO-100*. 1996.

[92] ISSC. *Design Methods, Report of Committee IV.2*. Proceedings of the International Ship and Offshore Structures Congress, Germany, 2012.

[93] Christoph Kecher. *UML 2.0: das umfassende Handbuch*. Galileo Press, 2009.

[94] Intaek Kim. Expert system in software engineering using structured analysis and design technique (sadt). Technical report, DTIC Document, 1990.

[95] Peter Klahold, Gunter Schlageter, and Wolfgang Wilkes. *A general model for version management in databases*. Gesamthochschule, FernUniversität, 1986.

[96] Thomas Koch. *Validation and Quality Control of Design and Production Information - Applying Rule Based Data Mining and Business Intelligence Concepts to Engineering*. Proceedings of the International Conference on Computer and IT Applicatios in the Maritime Industries, Gubbio, 2010.

[97] K Kosanke. Cimosa: Open system architecture for cim. *ESPRIT Consortium AMICE,*

Springer-Verlag, 1993.

[98] S Krüger et al. Offenes Methodenbanksystem für den Propulsorentwurf. *Flensburger Schiffbau-Gesellschaft GmbH&Co. KG., Technical Report, Flensburg*, 1999.

[99] Harald Kühn, Franz Bayer, Stefan Junginger, and Dimitris Karagiannis. Enterprise model integration. In *E-Commerce and Web Technologies*, pages 379–392. Springer, 2003.

[100] H Lackenby. On the systematic geometrical variation of ship forms. *Trans. INA*, 92:289–315, 1950.

[101] Thomas Lamb et al. *Ship design and construction*, volume 1. Society of Naval Architects and Marine Engineers, 2004.

[102] Thomas Lamb et al. *Ship design and construction*, volume 2. Society of Naval Architects and Marine Engineers, 2004.

[103] Marc Lankhorst. *Enterprise architecture at work: Modelling, communication and analysis.* Springer, 2013.

[104] Thomas Lauer. *Change Management, Grundlagen und Erfolgsfaktoren.* Springer, 2010.

[105] K Levander. System Based Ship Design. *Kompendium. TMR 4110 Marine Design and Engineering, Basic Course, NTNU. Trondheim*, 2006.

[106] Lakhoua M and Rahmouni M. Investigation of the methods of the enterprise modelling. *African Journal of Business Management*, 5(16):6845–6852, 2011.

[107] Carlos V. Malheiros. Ship 3d model and design database reuse for warship operation and maintenance purposes. In *Proceedings of the International Conference on Computer and IT Applicatios in the Maritime Industries (COMPITT), Budapest*, 2009.

[108] AVEVA Marine GmbH. AVEVA-Marine 12.1 SP3 Documentation.

[109] Fabio Massacci, John Mylopoulos, and Nicola Zannone. An ontology for secure socio-technical systems. *Handbook of Ontologies for Business Interaction. The IDEA Group*, 2007.

[110] F Mistree, WF Smith, B Bras, JK Allen, and D Muster. Decision-based design: a contemporary paradigm for ship design. *Transactions, Society of Naval Architects and Marine Engineers*, 98:565–597, 1990.

[111] E. Moredo and M. Krikke. Improving the re-use of design data during tender phase. In *Proceedings of the International Marine Design Conference, Glasgow*, 2012.

[112] ISO TC 184/SC4/WG 3 N1133. *ISO 10303-216, Industrial automation systems and integratio,Product data representation and exchange,Part 216: Application Protocol: Ship moulded forms.* 2003.

[113] ISO TC 184/SC4/WG3 N1226. *ISO 10303-215, Industrial automation systems and integratio,Product data representation and exchange,Part 215: Application Protocol: Ship arrangements.* 2004.

[114] Nicolas Rox. *Einfürhrung in die Entwurfsmethoden Datenbank E4.* Technische Universitärt Hamburg-Harburg, January 2014.

[115] H Nowacki. Developments in marine design methodology: roots, results and future trends. In *Tenth international marine design conference*, volume 1, 2009.

[116] Horst Nowacki. Five decades of computer-aided ship design. *Computer-Aided Design*, 42(11):956–969, 2010.

[117] B. Oers. *A Packing Approach For The Early Stage Design Of Service Vessels.* PhD thesis, Delft University of Technology, Netherland, 2011.

[118] B. Oers and H. Hopman. Simpler and faster: A 2.5d packing-based approach for early stage ship design. In *Proceedings of the International Marine Design Conference, Glasgow*, 2012.

[119] Stefan Otte. Version control systems. *Computer Systems and Telematics Institute of Computer Science Freie Universität Berlin, Germany*, 2009.

[120] A Papanikolaou, S Harries, M Wilken, and G Zaraphonitis. Integrated design and multiobjective optimization approach to ship design. In *Proceedings of the International Conference on Computer Applications in Shipbuilding, Italy*, 2011.

[121] Apostolos Papanikolaou. Holistic ship design optimization. *Computer-Aided Design*, 42(11):1028–1044, 2010.

[122] Charles J Petrie. *Enterprise integration modeling: proceedings of the first international conference.* The MIT Press, 1992.

[123] Aiko Pras and Juergen Schönwälder. On the difference between information models and data models. Technical report, RFC 3444, January, 2003.

[124] ProSTEP iViP Association. Enterprise Rights Management. Available at: http://www.prostep.org/. Technical report, 2009.

[125] Raghu Ramakrishnan and Johannes Gehrke. *Database management systems.* Osborne/McGraw-Hill, 2000.

[126] Tupper E C Rawson K J. *Basic Ship Theory.* Longmans, London, 1968.

[127] Douglas T Ross and Kenneth E Schoman Jr. Structured analysis for requirements definition. *Software Engineering, IEEE Transactions on*, (1):6–15, 1977.

[128] Jonathan Schaeffer, Duane Szafron, Greg Lobe, and Ian Parsons. The enterprise model for developing distributed applications. *Parallel & Distributed Technology: Systems & Applications, IEEE*, 1(3):85–96, 1993.

[129] Martin Schedlbauer. *The Art of Business Process Modeling: The Business Analyst's Guide to Process Modeling with UML & BPMN.* The Cathris Group, 2010.

[130] August-Wilhelm Scheer. *ARIS-Modellierungsmethoden, Metamodelle, Anwendungen.* Springer, 2001.

[131] Prof. Dr.-Ing Herbert Schneekluth. *Hydromechanik zum Schiffsentwurf: Vorlesungen.* Koehler, Herford, 1988.

[132] Alexander Dotor Schumann. *Entwurf und Modellierung einer Produktlinie von Software-Konfigurations-Management-Systemen.* PhD thesis, University of Bayreuth, 2011.

[133] SCRA. *STEP APPLICATION HANDBOOK ISO 10303 VERSION 3.* 2006.

[134] Avraham Shtub and Reuven Karni. Enterprise process modeling. In *ERP*, pages 31–57. Springer, 2010.

[135] Margarida Silva. Collaborative Project Management: Issues, methods and tools. Technical report, 2011.

[136] Paul R Smith and Richard Sarfaty. Creating a strategic plan for configuration management using computer aided software engineering (case) tools. Technical report, EG and G Idaho, Inc., Idaho Falls, ID (United States), 1993.

[137] Tim Sperber. *Aufbau einer Projektmanagementstruktur für Investitionsprojekte: Ein Leitfaden.* Diplomica Verlag, 2008.

[138] Josef L Staud. *Datenmodellierung und Datenbankentwurf: Ein Vergleich aktueller Methoden.* Springer, 2005.

[139] I Stroud and PC Xirouchakis. Stl and extensions. *Advances in Engineering Software*, 31(2):83–95, 2000.

[140] Robert S Swarz and Joseph K DeRosa. A framework for enterprise systems engineering processes. In *Proc. of International Conference on Software and Systems Engineering*, 2006.

[141] Orsolya Szegheo and Bjørn Andersen. Modeling the extended enterprise: a comparison of different modeling approaches. In *Proceedings of International Conference on Enterprise Modelling (IEMC'99), Verdal, Norge, June 14-16.* Productivity Press, 1999.

[142] Bhavani Thuraisingham. Mandatory access control. In *Encyclopedia of Database Systems*, pages 1684–1685. Springer, 2009.

[143] Frederick Henry Todd. Series 60 methodical experiments with models of single-screw merchant ships. Technical report, DTIC Document, 1963.

[144] M. Toyoda and Y. Nakajima. Competitive conceptual structural design system using 3d plm technology. In *Proceedings of the International Conference on Computer Applications in Shipbuilding , South Korea*, 2005.

[145] Alexander Tsolkas and Klaus Schmidt. Systemnahes Berechtigungskonzept. In *Rollen und Berechtigungskonzepte*, pages 215–227. Springer, 2010.

[146] Uwe Hollenbach. Respro XL-Tool.

[147] Bart van Oers, Douwe Stapersma, and Hans Hopman. A 3d packing approach for the early stage configuration design of ships. In *Proceedings of the 9th International Conference on Computer and IT Applications in the Maritime Industries (COMPITT2010)*, pages 367–381, 2010.

[148] François Vernadat. Ueml: towards a unified enterprise modelling language. *International Journal of Production Research*, 40(17):4309–4321, 2002.

[149] François Vernadat. The cimosa languages. In *Handbook on Architectures of Information Systems*, pages 251–272. Springer, 2006.

[150] Matthew West. *Developing high quality data models.* Access Online via Elsevier, 2011.

[151] Yang Yu and Tzi-cker Chiueh. Enterprise digital rights management: Solutions against information theft by insiders. *Research Proficiency Examination (RPE) report TR-169, Department of Computer Science, Stony Brook University*, 2004.

[152] Xiaohui Zhao and Chengfei Liu. Version management in the business process change context. In *Business Process Management*, pages 198–213. Springer, 2007.

[153] Michael Zimmermann. *Knowledge-Based Design Patterns for Detailed Ship Structural Design.* PhD thesis, University of Rostock, 2010.

A. Annex

A.1. Relationships of UML Class- and Package-Diagram

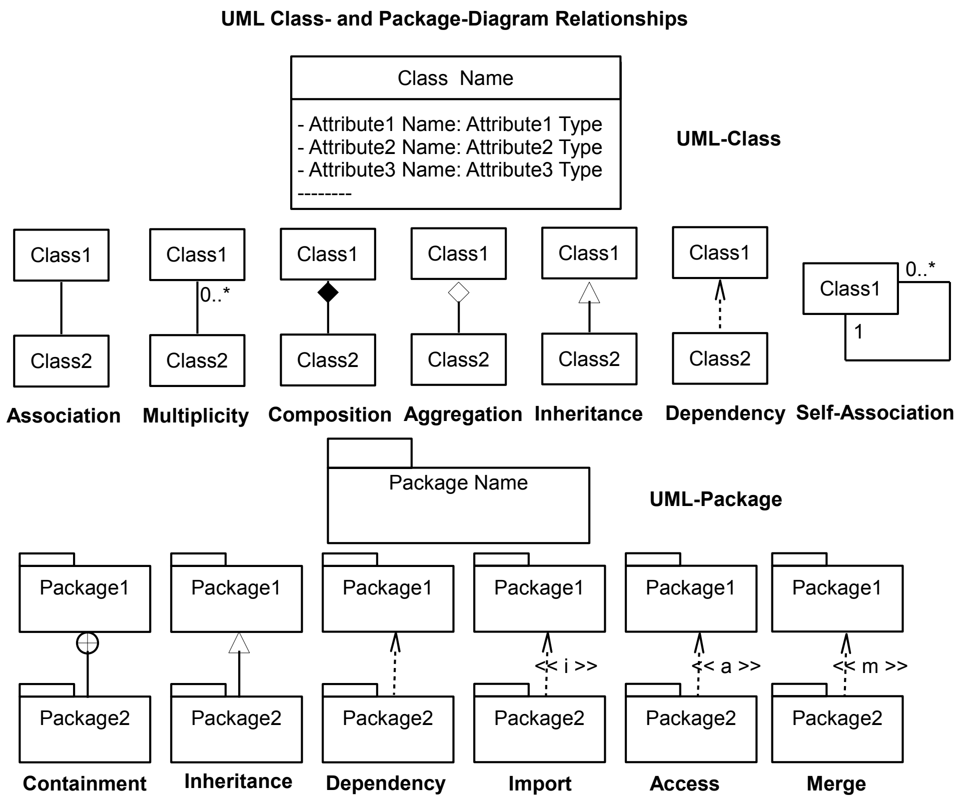


Figure A.1.: Relationships of UML Class- and Package-Diagram

A.2. Ship Information Model Packages

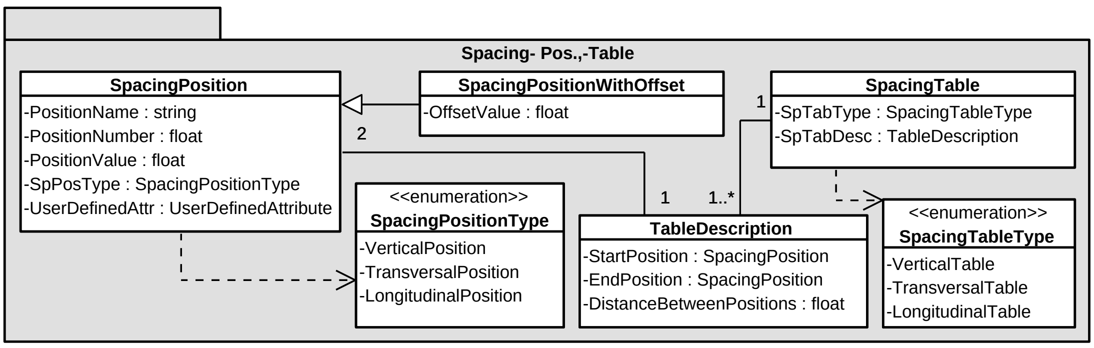


Figure A.2.: Spacing-Position,-Table Package

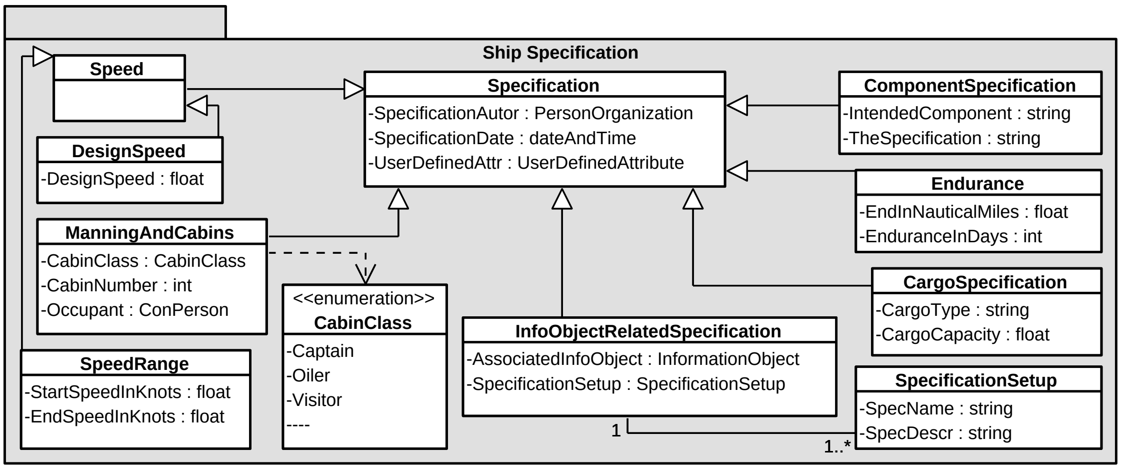


Figure A.3.: Ship Specification Package

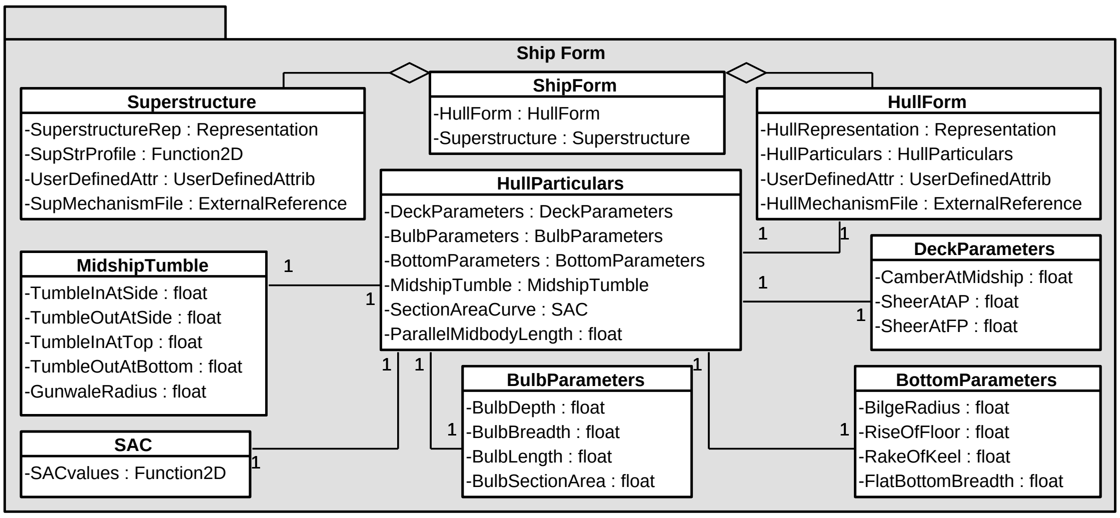

Ship Form
ShipForm
-HullForm : HullForm
-Superstructure : Superstructure
Superstructure
-SuperstructureRep : Representation
-SupStrProfile : Function2D
-UserDefinedAttr : UserDefinedAttrib
-SupMechanismFile : ExternalReference
HullForm
-HullRepresentation : Representation
-HullParticulars : HullParticulars
-UserDefinedAttr : UserDefinedAttrib
-HullMechanismFile : ExternalReference
HullParticulars
-DeckParameters : DeckParameters
-BulbParameters : BulbParameters
-BottomParameters : BottomParameters
-MidshipTumble : MidshipTumble
-SectionAreaCurve : SAC
-ParallelMidbodyLength : float
DeckParameters
-CamberAtMidship : float
-SheerAtAP : float
-SheerAtFP : float
MidshipTumble
-TumbleInAtSide : float
-TumbleOutAtSide : float
-TumbleInAtTop : float
-TumbleOutAtBottom : float
-GunwaleRadius : float
BulbParameters
-BulbDepth : float
-BulbBreadth : float
-BulbLength : float
-BulbSectionArea : float
BottomParameters
-BilgeRadius : float
-RiseOfFloor : float
-RakeOfKeel : float
-FlatBottomBreadth : float
SAC
-SACvalues : Function2D
1
1
1
1
1
1
1
1
1
1
1
1
1
1

Figure A.4.: Ship Form Package

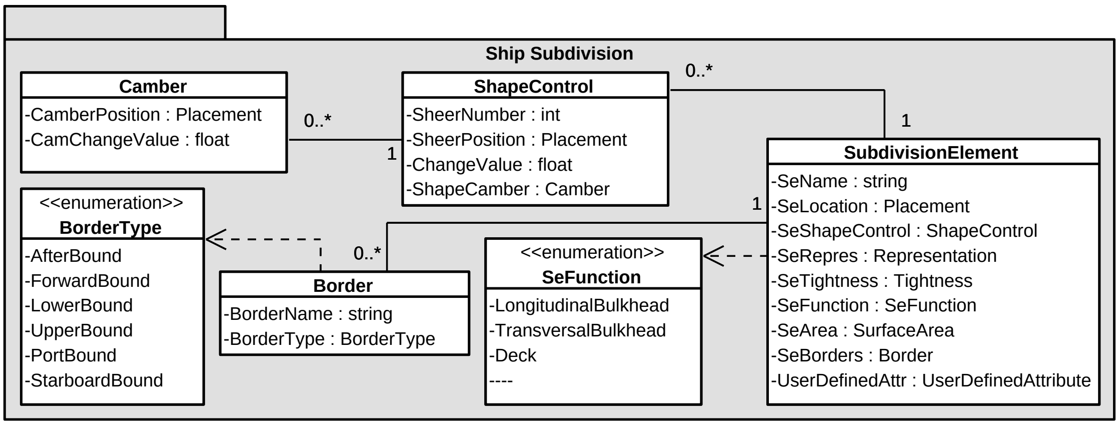


Figure A.5.: Ship Subdivision Package

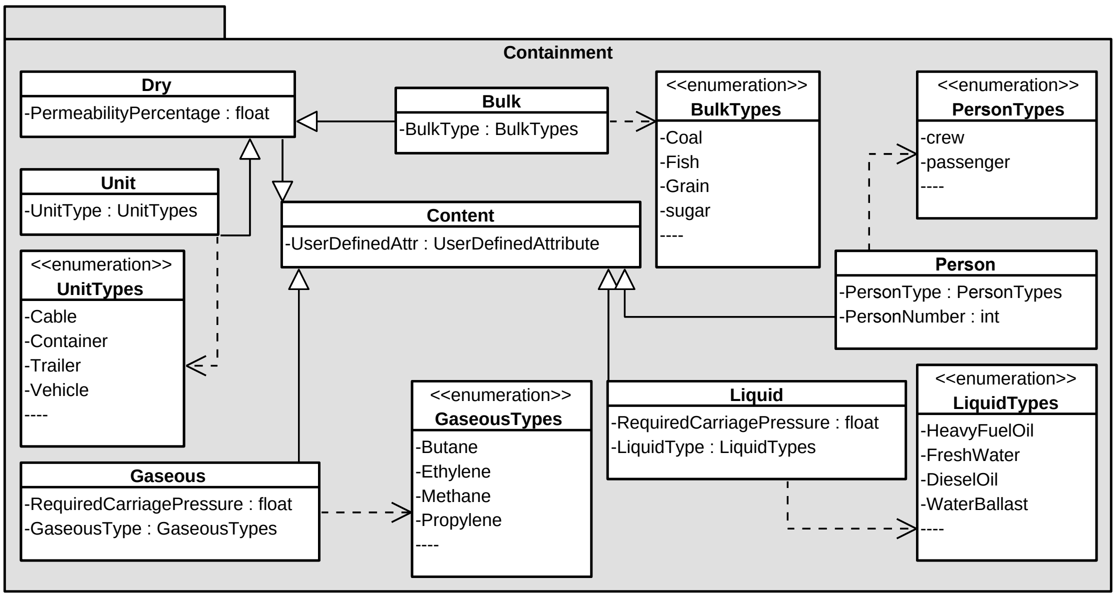


Figure A.6.: Containment Package

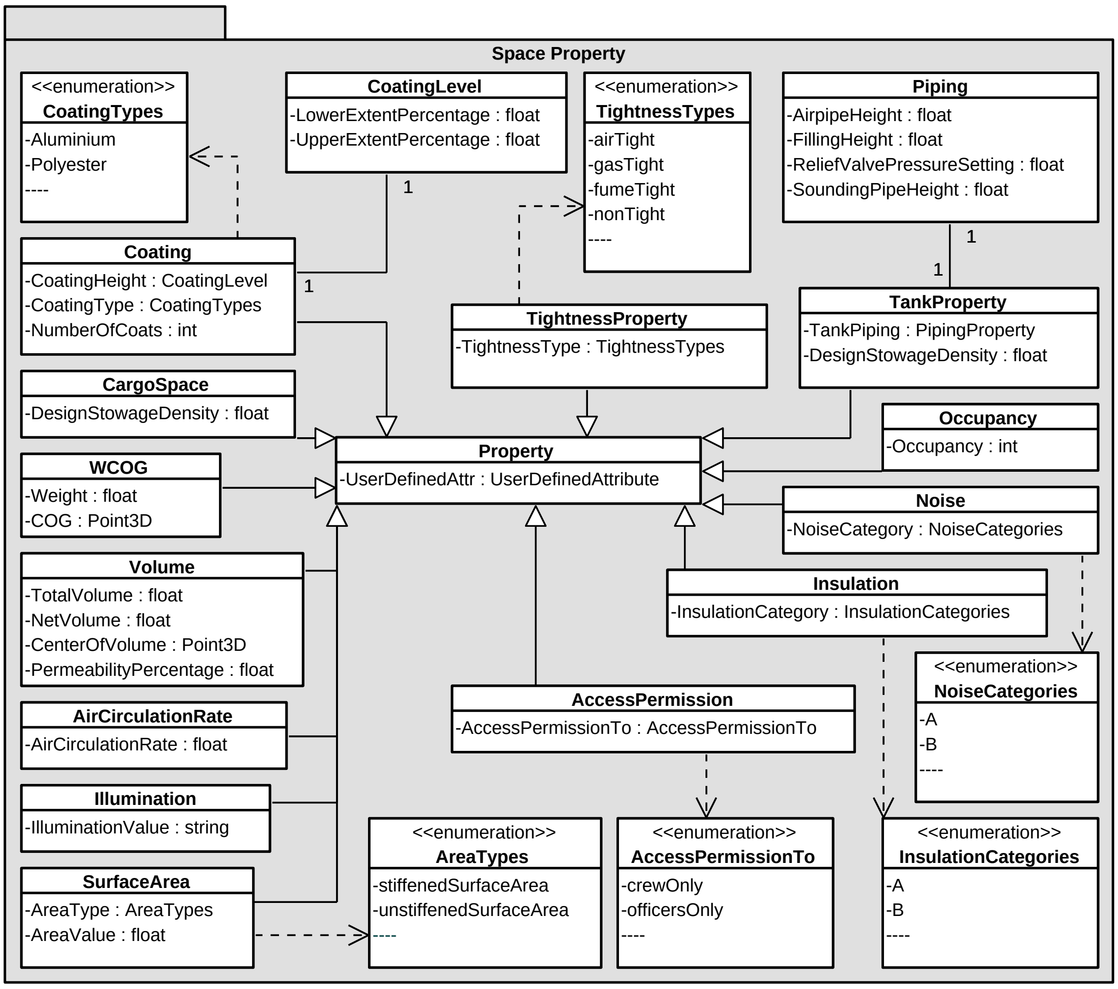


Figure A.7.: Space Property Package

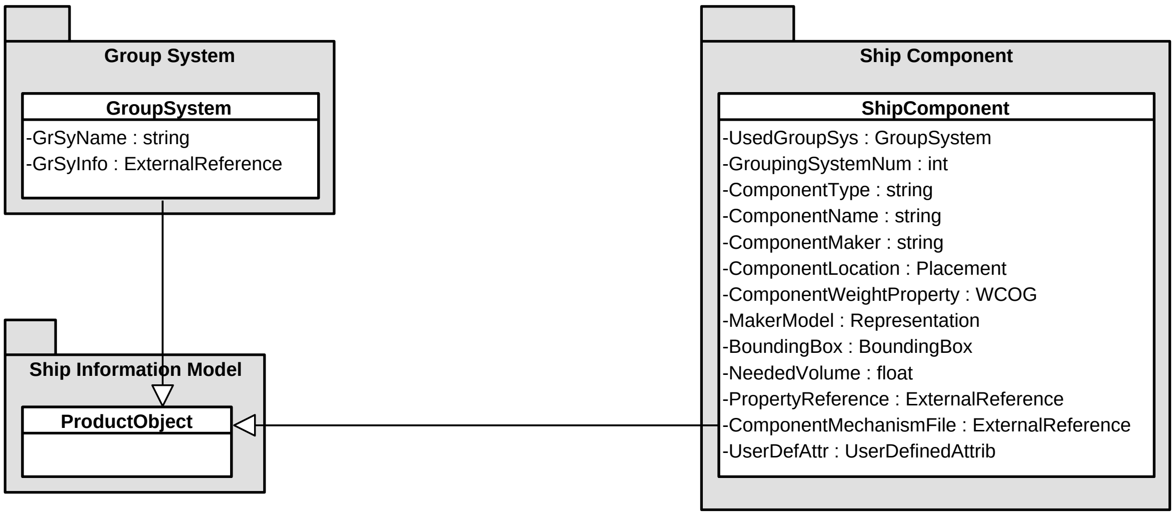

Group System
GroupSystem
-GrSyName : string
-GrSyInfo : ExternalReference
Ship Information Model
ProductObject
Ship Component
ShipComponent
-UsedGroupSys : GroupSystem
-GroupingSystemNum : int
-ComponentType : string
-ComponentName : string
-ComponentMaker : string
-ComponentLocation : Placement
-ComponentWeightProperty : WCOG
-MakerModel : Representation
-BoundingBox : BoundingBox
-NeededVolume : float
-PropertyReference : ExternalReference
-ComponentMechanismFile : ExternalReference
-UserDefAttr : UserDefinedAttrib

Figure A.8.: Ship Component and Group System Packages

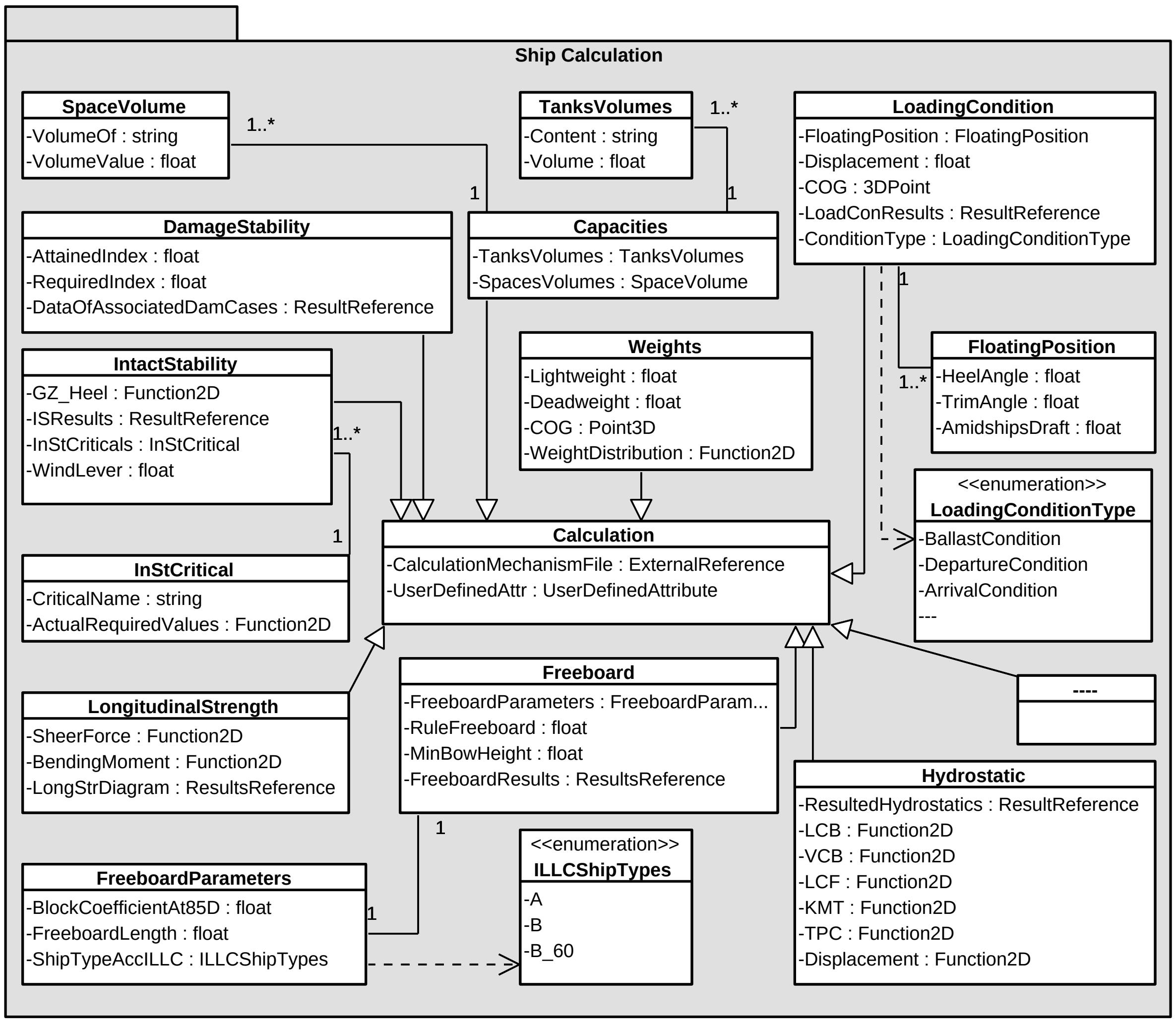


Figure A.9.: Part of Ship Calculation Package

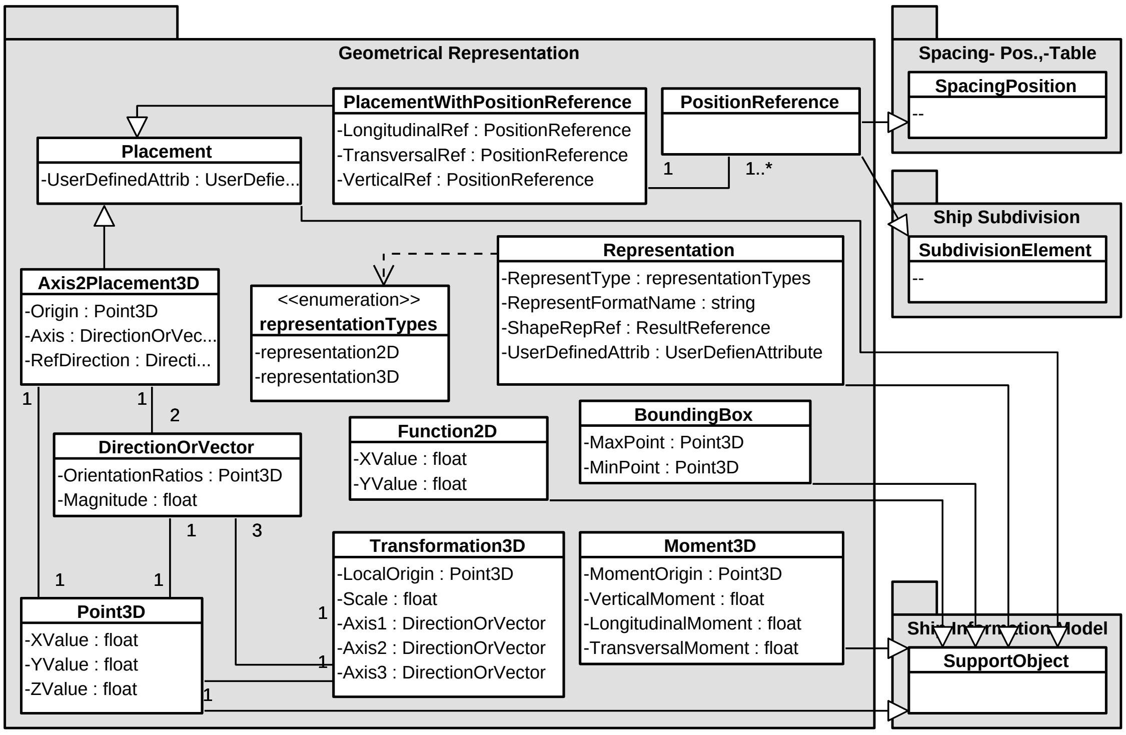


Figure A.10.: Geometrical Representation Package

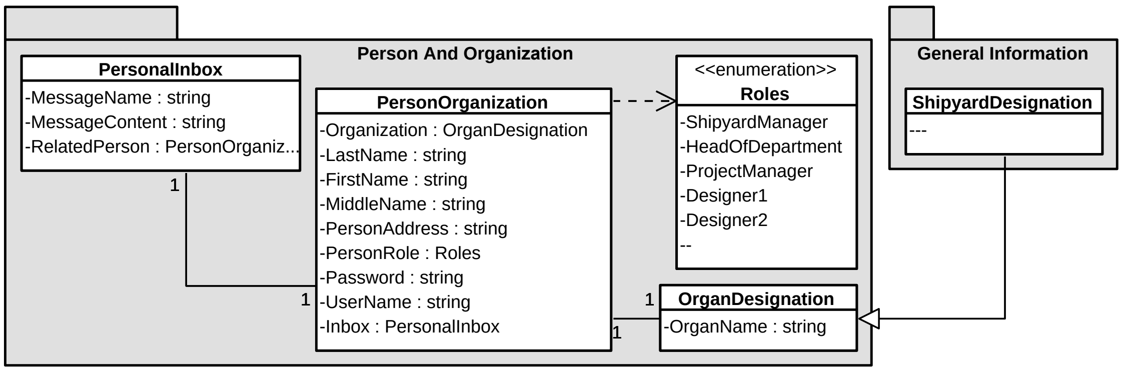


Figure A.11.: Person And Organization Package

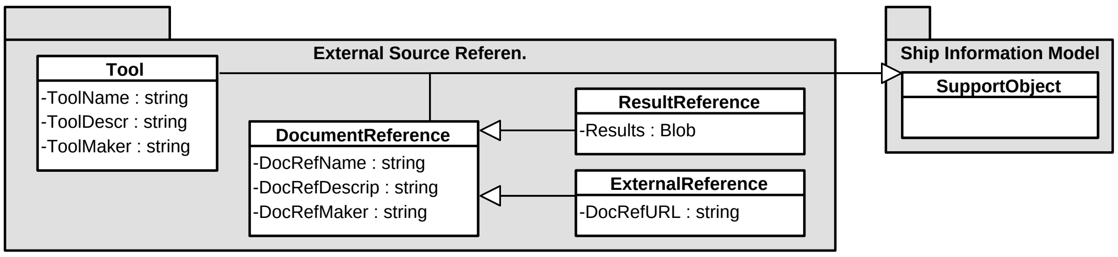


Figure A.12.: External Source References Package

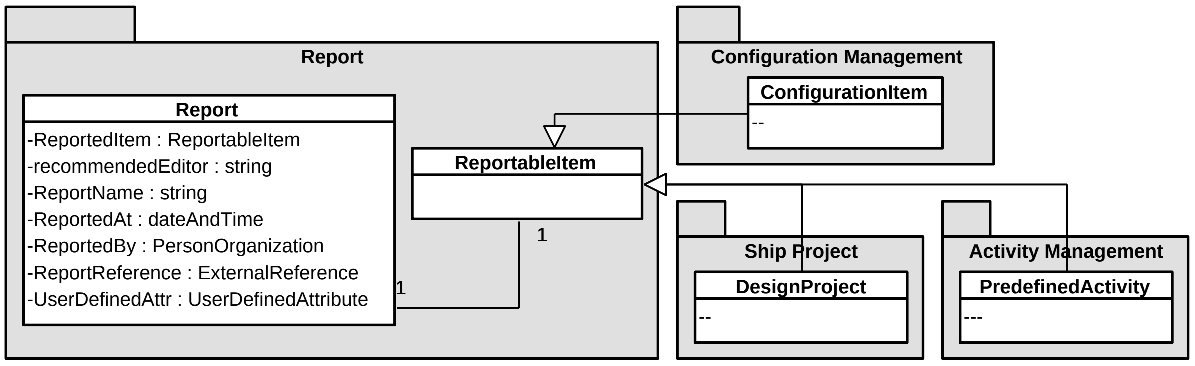

Report
Report
-ReportedItem : ReportableItem
-recommendedEditor : string
-ReportName : string
-ReportedAt : dateAndTime
-ReportedBy : PersonOrganization
-ReportReference : ExternalReference
-UserDefinedAttr : UserDefinedAttribute
ReportableItem
1
1
Configuration Management
ConfigurationItem
--
Ship Project
DesignProject
--
Activity Management
PredefinedActivity

Figure A.13.: Report Package

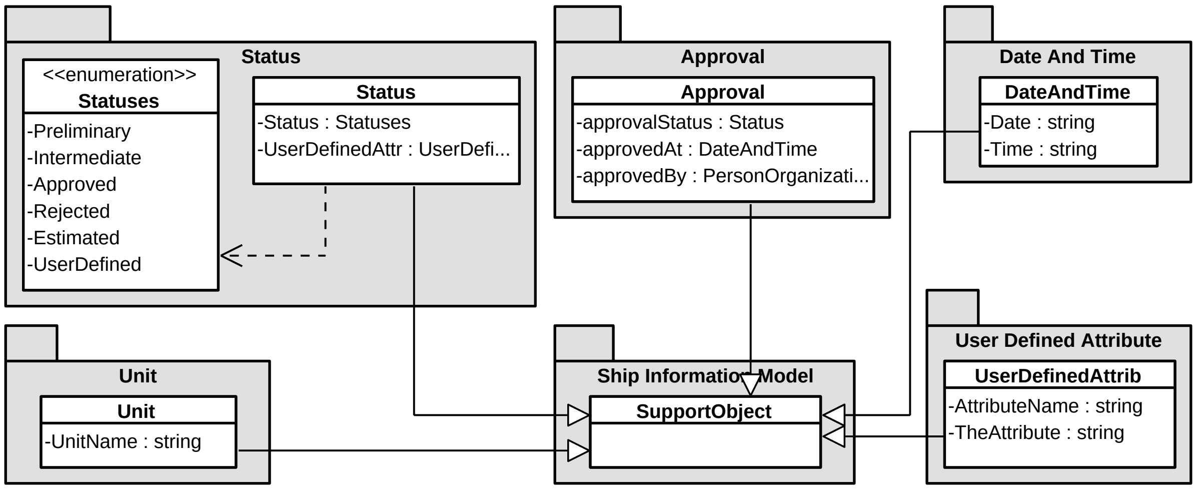


Figure A.14.: Status, Approval, Date And Time, User Defined Attribute and Unit Packages

A.3. Predefined Activities Partitions

Activity Partition		Interacted Information Object	
Num	*Name*	*Name*	*Role*
1	Apply setup mechanism	Tool	Mechanism
2	Specify design project information	DesignProject	Output
3	Specify principal characteristics	PrincipalCharacteristics	Output
4	Specify ship designation	ShipDesignation	Output
5	Specify design specifications	Specification	Output
6	Specify tasks responsibilities	RightAuthorization	Output
7	Define `Design Project` configuration	ConfigurationItem	Output

Table A.1.: Setup Early Design Project as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Get principal characteristics	PrincipalCharacteristics	Input
2	Get ship designation	ShipDesignation	Input
3	Check/Get hull form specification	InfoObjectRelated- Specification	Control
4	Apply hull creator mechanism	Tool	Mechanism
5	Save resulted hull-form	ShipForm	Output
6	Define `Ship Form` configuration	ConfigurationItem	Output

Table A.2.: Generate Hull Form as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Check/Get ship form	ShipForm	Input
2	Identify floating position(s)	FloatingPosition	Control
3	Apply hydrostatics solver	Tool	Mechanism
4	Save calculated hydrostatics	Hydrostatic	Output
5	Define `Hydrostatics` configuration	ConfigurationItem	Output

Table A.3.: Calculate Hydrostatics as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Check/Get ship form	ShipForm	Input
2	Check/Get components needed spaces	ShipComponent	Control
3	Check/Get required capacities	Capacities	Control
4	Apply compartmentation mechanism	Tool	Mechanism
5	Save compartmentation	Compartmentation	Output
6	Define `Compartmentation` configuration	ConfigurationItem	Output

Table A.4.: Create Compartmentation as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Select/Get ship form	ShipForm	Input
2	Select/Get compartmentation	Compartmentation	Input
3	Select/Get weights	Weights	Input
4	Identify definition rules	RegulationsAndRules/ Specification	Control
5	Apply loading condition mechanism	Tool	Mechanism
6	Save defined loading condition	LoadingCondition	Output
7	Define `Loading Condition` configuration	ConfigurationItem	Output

Table A.5.: Define Loading Condition as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Select/Get defined loading condition	LoadingCondition	Input
2	Identify intact stability criteria	RegulationsAndRules	Control
3	Apply intact stability solver	Tool	Mechanism
4	Save intact stability results	IntactStability	Output
5	Define `Intact Stability` configuration	ConfigurationItem	Output

Table A.6.: Calculate Intact Stability as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Select/Get defined loading condition	LoadingCondition	Input
2	Identify damage stability criteria	RegulationsAndRules	Control
3	Apply damage stability solver	Tool	Mechanism
4	Save damage stability results	DamageStability	Output
5	Define `Damage Stability` configuration	ConfigurationItem	Output

Table A.7.: Calculate Damage Stability as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Select/Get ship form	ShipForm	Input
(1)	Get principal characteristics	PrincipalCharacteristics	Input
2	Select/Get components weights	ShipComponent	Input
3	Select/Get compartmentation	Compartmentation	Input
4	Get cargo specifications	CargoSpecification	Control
5	Apply estimate weight mechanism	Tool	Mechanism
6	Save estimated weight	Weights	Output
7	Define `Weight` configuration	ConfigurationItem	Output

Table A.8.: Estimate Weight as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Get principal characteristics	PrincipalCharacteristics	Input
2	Get cargo specification	CargoSpecification	Control
3	Get speed range	SpeedRange	Control
4	Get endurance	Endurance	Control
5	Get manning specification	ManningAndCabins	Control
6	Select/Get predicted power	Power	Control
7	Estimate capacities	Tool	Mechanism
8	Save estimated capacities	Capacities	Output
9	Define `Capacities` configuration	ConfigurationItem	Output

Table A.9.: Estimate Capacities as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Select/Get compartmentation	Compartmentation	Input
2	Select/Get predicted power	Power	Input
3	Select/Get component specifications	ComponentSpecification	Control
4	Apply component mechanism	Tool	Mechanism
5	Save component	ShipComponent	Output
6	Define `Ship Component` configuration	ConfigurationItem	Output

Table A.10.: Define Component as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Select/Get ship form	ShipForm	Input
(1)	Get principal characteristics	PrincipalCharacteristics	Input
2	Get speed range	SpeedRange	Control
3	Specify prediction floating position	FloatingPosition	Control
4	Apply predict resistance mechanism	Tool	Mechanism
5	Save predicted resistance	Resistance	Output
6	Define `Resistance` configuration	ConfigurationItem	Output

Table A.11.: Predict Resistance as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Get principal characteristics	PrincipalCharacteristics	Input
2	Get speed range	SpeedRange	Control
3	Select/Get propeller characteristics	PropellerCharacteristics	Control
4	Apply predict propulsion mechanism	Tool	Mechanism
5	Save predicted Propulsion	Propulsion	Output
6	Define `Propulsion` configuration	ConfigurationItem	Output

Table A.12.: Predict Propulsion as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Check/Get Resistance	Resistance	Input
2	Check/Get Propulsion	Propulsion	Input
3	Check/Get Propeller Characteristics	PropellerCharacteristics	Control
(4)	Get principal characteristics	PrincipalCharacteristics	Input
4	Get speed range	SpeedRange	Control
5	Specify prediction floating position	FloatingPosition	Control
6	Check/Get shaft efficiency	ShipComponent	Control
7	Apply predict power mechanism	Tool	Mechanism
8	Save predicted power	Power	Output
9	Define `Power` configuration	ConfigurationItem	Output

Table A.13.: Predict Power as Predefined Activity

Activity Partition		Interacted Information Object	
Num	*Name*	*Name*	*Role*
1	Check/Get ship form	ShipForm	Input
2	Identify the rules	RegulationsAndRules	Control
3	Apply freeboard solver	Tool	Mechanism
4	Save calculated freeboard	Freeboard	Output
5	Define `Freeboard` configuration	ConfigurationItem	Output

Table A.14.: Calculate Freeboard as Predefined Activity

Activity Partition		Interacted Information Object	
Num	*Name*	*Name*	*Role*
1	Check/ Get defined loading condition	LoadingCondition	Input
2	Apply longitudinal strength mechanism	Tool	Mechanism
3	Save longitudinal strength results	LongitudinalStrength	Output
4	Define `Longitudinal Strength` configuration	ConfigurationItem	Output

Table A.15.: Calculate Longitudinal Strength as Predefined Activity

Activity Partition		Interacted Information Object	
Num	*Name*	*Name*	*Role*
1	Check/Get ConfigurationItem(s)	ConfigurationItem	Input
2	Define `Design solution` configuration	ConfigurationItem	Output

Table A.16.: Create Design Solution as Predefined Activity

Activity Partition		Interacted Information Object	
Num	*Name*	*Name*	*Role*
1	Check/Select ConfigurationItem	ConfigurationItem	Input
2	Specify the status	Status	Input
3	Update the configuration status	ConfigurationItem	Output

Table A.17.: Update Status as Predefined Activity

Activity Partition		Interacted Information Object	
Num	*Name*	*Name*	*Role*
1	Show/Select ConfigurationItem(s)	ConfigurationItem	Input
2	Define dependency	DifferenceDefinition	Input
3	Add to version-history	VersionsHistory	Output

Table A.18.: Create Version as Predefined Activity

Activity Partition		Interacted Information Object	
Num	*Name*	*Name*	*Role*
1	Identify the wanted ChangeableItem	ChangeableItem	Input
2	Evaluate requested change	ChangeEvaluation	Output

Table A.19.: Check Change Influences as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Identify the wanted ChangeableItem	ChangeableItem	Input
2	Justify the requested change	RequestJustification	Control
3	Evaluate requested change	ChangeEvaluation	Output
4	Save data to the ChangeRequest	ChangeRequest	Output

Table A.20.: Request Change as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Check/Get ChangeRequest	ChangeRequest	Input
2	Approve/Reject	Approval	Output
3	Save data to the ChangeControl	ChangeControl	Output
4	Release Info to concerned persons	PersonalInbox	Output

Table A.21.: Approve/Reject Requested Change as Predefined Activity

Activity Partition		**Interacted Information Object**	
Num	*Name*	*Name*	*Role*
1	Show/Select available reportable items	ReportableItem	Input
2	Create/Output report	Tool	Mechanism
3	Save the created report	Reporting	Output

Table A.22.: Create Report as Predefined Activity

A.4. Implemented UseCase(s)

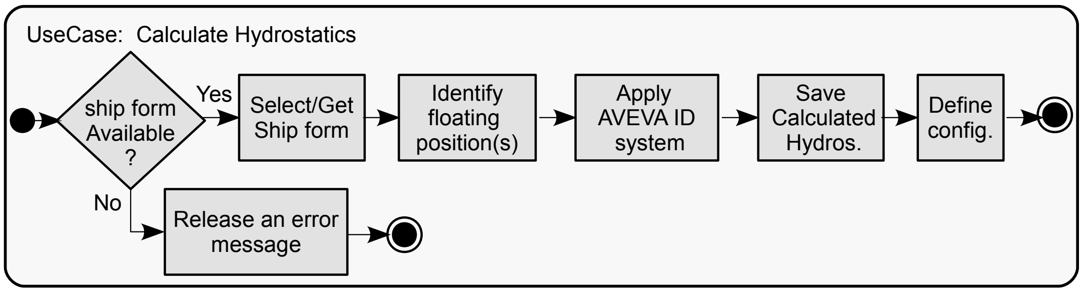


Figure A.15.: UseCase: Calculate Hydrostatics

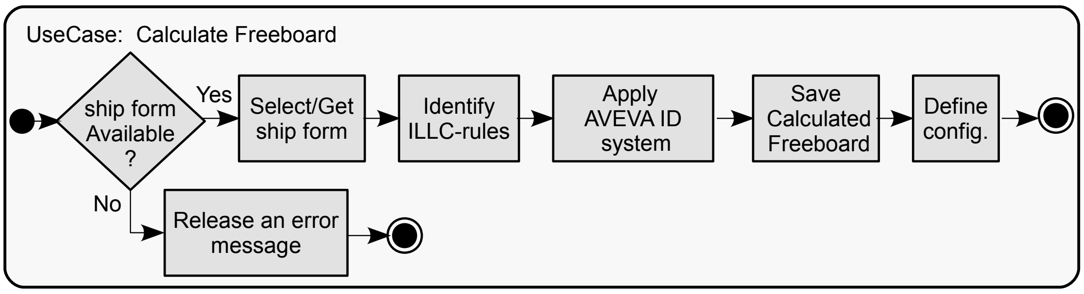


Figure A.16.: UseCase: Calculate Freeboard

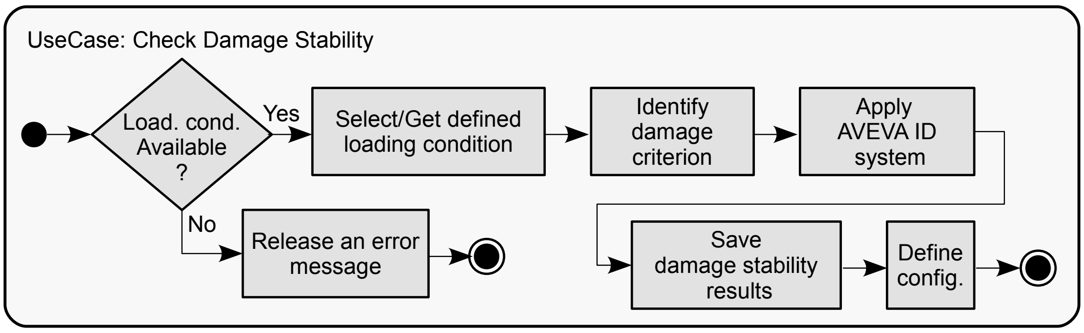

UseCase: Check Damage Stability
Load. cond. Available ?
Yes
No
Select/Get defined loading condition
Identify damage criterion
Apply AVEVA ID system
Release an error message
Save damage stability results
Define config.

Figure A.17.: UseCase: Calculate Damage Stability

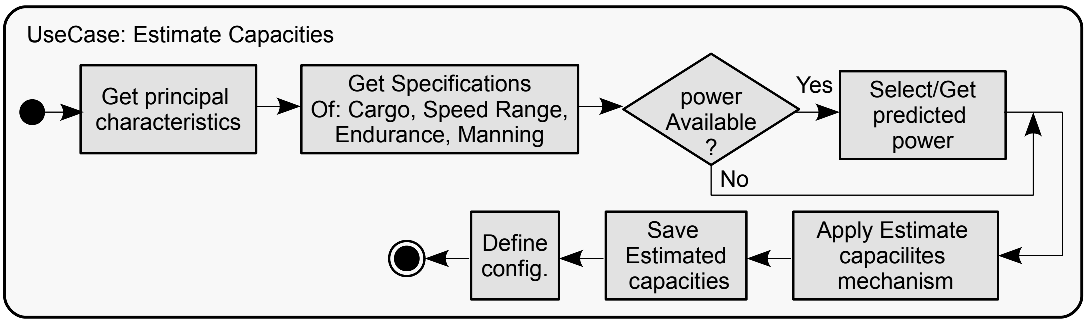

UseCase: Estimate Capacities
Get principal characteristics
Get Specifications Of: Cargo, Speed Range, Endurance, Manning
power Available ?
Yes
No
Select/Get predicted power
Apply Estimate capacilites mechanism
Save Estimated capacities
Define config.

Figure A.18.: UseCase: Estimate Capacities

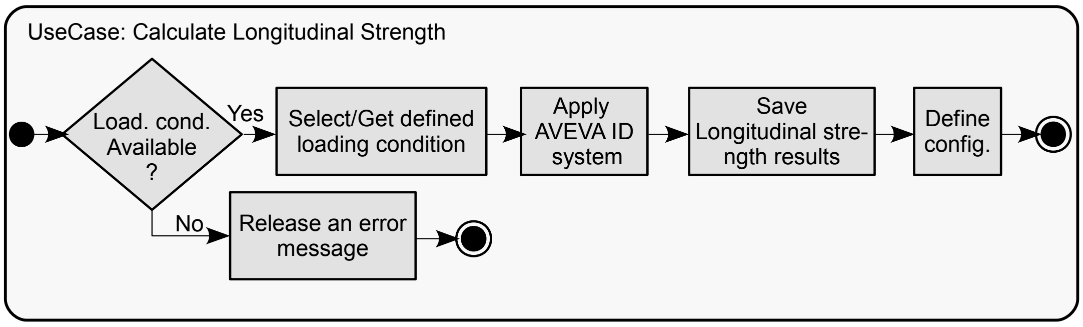

UseCase: Calculate Longitudinal Strength
Load. cond. Available ?
Yes
No
Select/Get defined loading condition
Apply AVEVA ID system
Save Longitudinal stre-ngth results
Define config.
Release an error message

Figure A.19.: UseCase: Calculate Longitudinal Strength

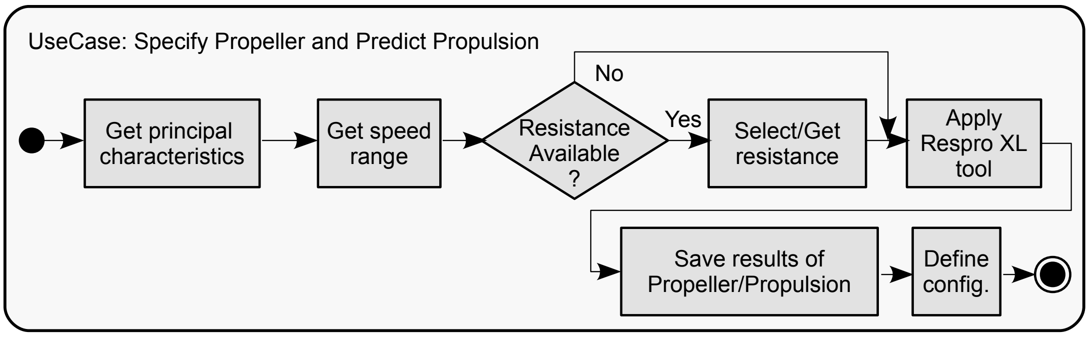

UseCase: Specify Propeller and Predict Propulsion
Get principal characteristics
Get speed range
Resistance Available ?
No
Yes
Select/Get resistance
Apply Respro XL tool
Save results of Propeller/Propulsion
Define config.

Figure A.20.: UseCase: Specify Propeller And Predict Propulsion

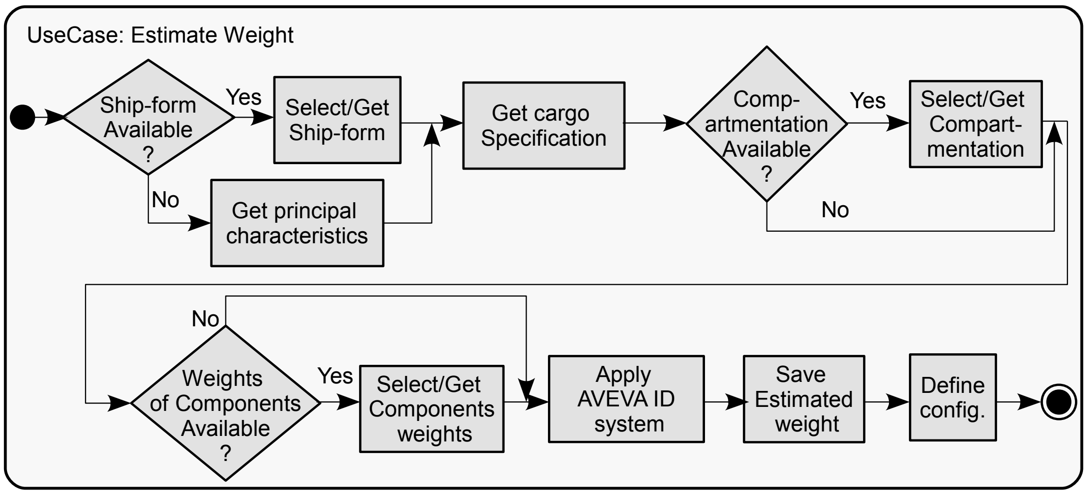


Figure A.21.: UseCase: Estimate Weight

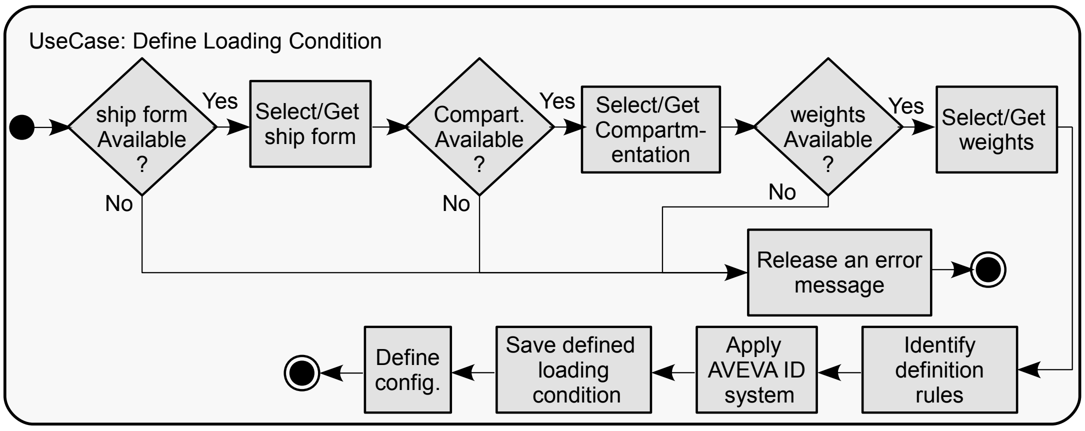


Figure A.22.: UseCase: Define Loading Condition

Publication related to this dissertation

- R. Bronsart, W. Jabary, An Approach for the Compartmentation in Early Ship Design, Proceedings of the 12th International Symposium on Practical Design of Ships and Other Floating Structures (PRADS), 2013

Declaration of Primary Authorship

With this statement I declare that I have independently completed the above thesis entitled *An Enterprise Modeling Approach for the Early Ship Design*. The thoughts taken directly or indirectly from external sources are properly marked as such. This thesis was not previously submitted to another academic institution and has also not yet been published.

Rostock, 2015
Wisam Jabary

Acknowledgement

Several years have been spent at the Chair of Naval Architecture at the University of Rostock while completing my PhD-thesis. My doctoral journey has been filled with happy and sad moments, which have shared with many people. To these people I owe my heartfelt appreciation. Firstly, I would like to express my sincere gratitude to Prof. Robert Bronsart. Furthermore, I would like to voice my sentiments of very deep appreciation to you for granting me this opportunity, and I am also deeply thankful for your encouragement and support throughout the entire time. Your advices and help with my research and career, as well as your recommendations regarding personal matters have been invaluable. I would also like to thank Prof. Patrick Kaeding for his comments. I am heartily thankful to the staff of the the Chair of Naval Architecture for many valuable discussions, particularly Jonas Conradin Wagner. Also, thanks to Jonas for proofreading the manuscript.
I would like to offer my special thanks to my parents who have sacrificed their lives for my brother and sisters and myself, and provided unconditional love and care. I owe them everything and words can not express how grateful I am to them. I would also like to thank my brother. Samer, you have been my best friend all my life, you deserve my sincerest thanks for your advices and support. You are really my special soul-mate. I am particularly grateful for the unconditional love and support given by my sisters, and I wish I could show them just how much I love and appreciate them. I would like to express my profound gratitude to my brother-in-law, sister-in-law, my wife's wonderful family (father, mother, sister and brother) who have all been supportive and caring. To the small angels, my nephew and nieces, I would like to say that I love you so much and thank you for being my eternal sunshine. Finally, to my wife, Boushra Assaf, who already has my heart and whose love and encouragement allowed me to finish this journey, even though she had been thousands of miles away physically for a long time, I would like to say that I owe you my heartfelt appreciation, especially for keeping up with my moods during the last year. I would like to ask you, Boushra, to share with me this moment, and to together dedicate this work to our small beloved flower, our daughter "Zenab". I love you both very much.

Rostock, 2015
Wisam Jabary

Curriculum Vitae

Personal Information:

Surname: Jabary
First name: Wisam
Date of Birth: 21.06.1983
Gender: Male
Nationality: Syrian
E-mail: wjabary@yahoo.com

Work Experience and Education:

2009-2015: PhD student at Rostock University, Germany
2007-2009: Scientific assistant at Tishreen University, Latakia, Syria
2001-2007: Studying of marine engineering at Tishreen University, Latakia, Syria
1998-2001: Secondary school, syrian school leaving certificate, Latakia, Syria
1989-1998: Elementary school and junior high school, Latakia, Syria

Language Skills:

Arabic: Native language
English: Fluent in speaking and writing
German: Fluent in speaking and writing