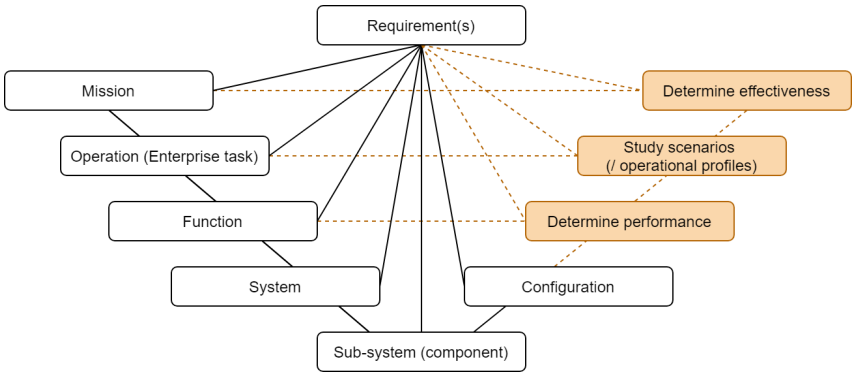
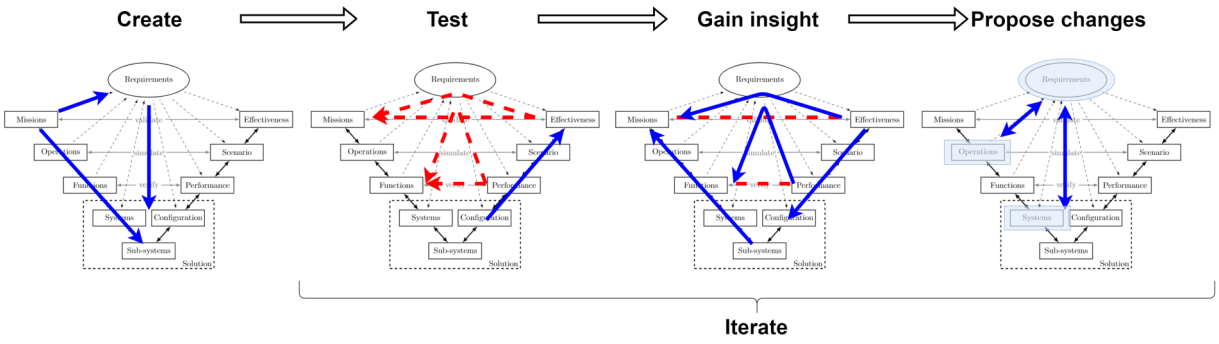
Defence Materiel Organisation
Ministry of Defence

Traceability in ship design

Connecting need, requirements and design in one information model

H.W. Van der Weg





TU Delft

Traceability in ship design

Connecting need, requirements and design in one information model

by

H.W. Van der Weg

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on **Wednesday** April 8, 2020 at **15:00**.

Report number:	SDPO.19.043.m	
Student number:	4157737	
Project duration:	April 3, 2019 – April 8, 2020	
Thesis committee:	Prof. Dr. ir. J.J. Hopman,	Delft University of Technology
	Dr. A.A. Kana,	Delft University of Technology
	Dr. W.W.A. Beelaerts van Blokland	Delft University of Technology
	Dr. ir. B. Van Oers	Defence Materiel Organisation

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Traceability in complex ship design involves keeping track of the evolving relations between need, requirements and design. Such traceability is not fully developed in the current design process at DMO due to divisions in the design process and between design tools. This thesis proposes to use a single information model to facilitate traceability by connecting the need, requirements and design in a single database. Besides enabling traceability, such an information model can also be used to support modern modelling design tools.

In the early design stage of naval vessels – the preliminary design phase – the capability, technical feasibility and affordability of a new ship are defined. Interactions between these three aspects are opaque and rely on the need for a new naval vessel and supporting requirements and preliminary design(s).

Determining the need, requirements and design is in this thesis considered to be a 'wicked problem'. In such problems, information gained by developing solutions helps define the problem itself. However, these solutions can only be determined with a certain interpretation of the initial problem. This results in many (evolving) interactions between the need, requirements and design in the ship design process as one is needed to define the other. With the help of Systems Engineering and Requirements Engineering information and relations in a ship design process can be determined and structured. These engineering fields are widely applied, and current technology enabled the development of computer aided approaches incorporating design theory, including computer aided traceability.

One of the current developments in design approaches is the introduction of model-based approaches, which creates the possibility to actively manage more relations and interactions in the design process. Model based approaches such as the Design Structure Matrix are used to visualise relations between information, and Axiomatic Design enables the decomposition and creation of complex relations. A different approach to modelling in design processes is the use of Knowledge Based Engineering. Knowledge Based Engineering field aims to capture not only the relations and interactions between information in design processes, but also 'knowledge' about the 'how' and 'why' in these processes. Each of these model-based developments relies on, or enables, traceability of information. However, having a program capable of supporting an information storage to support these approaches remains difficult to achieve. For this purpose, Shipbuilder software was created to support transparent storage and management of information.

In this thesis, theories for modelling approaches, Systems and Requirements Engineering and the DMO design process have been combined. This resulted in one information structure, applied in the Shipbuilder application, which enables traceability. To show that the proposed information model can model a ship and supports traceability, a case study has been performed. A small model of both an S- and L-frigate has been created to show how traceability can be performed in this information model.

With the help of an information model representing the design process in a single database, it seems possible to trace interactions between need, requirements and design. This will result in having better understanding of the relations and interactions between capability, feasibility and cost. The method in this thesis can help to better understanding of the interactions in less time, freeing time to make more and/or better improvements during the preliminary design phase.

Preface

Since I was a little kid, I wanted to design ships. This is one of the reasons I enjoyed studying Maritime Technology at the TU Delft, where I learned about the complexity and underlying theories of ship design. In the last few years of my study I got interested into two more specific aspects of ship design: model-based developments and determining the need of a customer for a design. These interests drove me towards the subject of this thesis which I have performed at the Defensie Materieel Organisatie (DMO). They wanted to introduce a new feature in their design process in order to keep track of the evolving connections between design, requirements and need.

During my research period at DMO, I learned about the complex nature of designing naval vessels, tools and ways of supporting this and the enthusiastic engineers working together to create new ships. Therefore I would like to thank the team of Afdeling Maritieme Systemen (AMS) I worked with at DMO for their support and enthusiasm.

My supervisors, Dr. ir. B.J. Van Oers of DMO and Dr. A.A. Kana of the TU Delft, I would like to thank for their guidance, supervision and valuable discussions during my research.

I would also like to thank G. Schouten and M. Van Loenen of Shipbuilder for their effort in helping me with the research and the inspiring conversations we had.

Finally, I would like to thank my family and friends for their support and help with the research by means of sparring and discussions. A special thanks here is for my dad, for aiding me during the writing of the thesis and in his help to clarify my story.

I hope you will enjoy reading this report and learn from it - as I did in writing it.

H.W. Van der Weg
Delft, March 2020

Contents

List of definitions

Definitions from literature

Capability The ability to complete a task or execute a course of action under specified conditions and level of performance [33]. It relates to *effectiveness* as used by Duchateau [15], Van Oers et al. [44].

Feasibility A feasible ship (or system) fulfils physical (and technical) requirements and design constraints. For example the need for a ship to float (upright) [43, p. 8]. In this thesis a feasible ship does not necessarily fulfil the need for this ship. It relates to *performance* as used by Duchateau [15], Van Oers et al. [44].

Cost The initial (building) cost of building a ship (or system). It relates to the *cost* and *financial feasibility* as used by [15, 44]. A ship costing less than the budget is affordable.

Need The need is the problem that a customer has which initiates the design of a ship. The need can be captured in ”stated requirements” which are requirements given by the customer[47, p. 2].

Requirement A statement is an attribute in a system, a statement to help guide the design of a system, in a way that is useful for different stakeholders and applies value of the system to a customer or user[31, p. 94][47, p. 2].

Configuration Integration of requirements, systems and components into a solution including a proposed general arrangement with spaces, fulfilling the need [15, p. 7].

Definitions as used in this thesis

Suitable ship A suitable ship is capable, feasible and affordable (see definitions above).

Insight Knowledge or understanding of information, relations and interactions.

Partial solution A partial solution is a (combination of) system(s) which can be incorporated in the configuration of a design. A partial solution fulfils a part of the complete set of requirements.

Configuration The configuration of a ship is a integration of several partial solutions. This will lead to an arrangement of spaces and systems. Based on Duchateau [15].

Design The design of a ship is the combination of the configuration, partial solutions and requirements. This design is then the whole ’solution’ for the design problem.

Configuration design cycle
The part of the design cycle concerned with creating a configuration (to fit the requirements).

Requirements cycle
The part of the design cycle concerned with creating requirements.

Creating (phase) The phase in a design cycle where a complex system is created and/or defined.

Testing (phase) The phase in a design cycle where the complex system is tested on fulfilling capability, feasibility and cost.

Insight phase The phase in a design cycle where insight in a ship design. This phase is used to propose changes to the complex system if needed.

Propose changes (phase)
This phase in a design cycle is where the proposed changes to the complex system are defined with the help of the insight gained in the previous phase.

1 Context

1.1. Buying a naval vessel

Buying a ship is like buying a new car. When buying a car, you only have a rough idea of what you want the new car to be - except when you want to buy that one specific (dream) car. While searching for the new car you get a better view of what car you actually want. This can be because you have seen enough candidates, or because a salesman tried to sell you one of his models.

During this search there are two things to which you will test a car: capability and cost. The capability of the car is the extent to which the car fulfils the need you have for the new car. If a car is not able to fulfil your need, this car is an unsuited option. A car is similarly unsuitable if it is too expensive to buy. A new car can also be unsuitable if it is not feasible. In the example of a car, this feasibility is far fetched as all considered options are already designed. Yet, a car can be unsuitable due to feasibility in constraints. Take for example a restriction in the width of a car. Living in a historic, small, city centre with narrow streets will make the option of an old school American car infeasible, as this car is simply too wide to navigate through the streets.

So the buying process of a car actually relies on three things: capability, feasibility and cost. These three can be determined by relating a possible car to your need, situation, and wallet. What it exactly is you want and can afford becomes more clear with each possible car you examine.

When buying a ship, the process above can be used in a similar way. There is a need for a ship; and this new ship needs to be capable, feasible and affordable. In the beginning it is not completely clear what a suitable ship looks like, but this will become more clear during the process.

When the Dutch government wants a new ship, they too go through this process. One big difference here - one that makes it a seemingly completely different problem - is that there are generally no complete, suitable, off-the-shelf options. The options for new naval vessels have to be created during the searching process for a new ship; they have to be designed.

Designing a vessel with the need known would then surely lead to the new design to be perfect, as it can be made to match the need. This, unfortunately, has been proven not to be possible. Having a design to be created to match the need will most likely result in the need to be altered slightly during the designing. This is the result of the need becoming more clear and being better understood during the buying process. As a design is created from the need, and not only matched to it, this will result in the design having to change. With this new design possibly leading to more knowledge on the need, and so forth.

This led to the use of the iterative design process, guiding the definition of the need and the creation of a design in the same process. These iterative processes are not only used to determine a possible solution to fulfil the need, they are also used to design these possible solutions. Designing in these processes is just as big a part as the buying process.

So the major difference between buying a car and a naval vessel, is that the latter does not

only rely on a buying process. In this buying process the need becomes more clear through looking at alternatives. The addition to this, when buying a naval vessel, the alternatives are to be designed in this process as well.

1.2. Designing the naval vessel

It is the mission of the Defence Materiel Organisation (DMO), and more specifically the department for marine systems ([Afdeling Maritieme Systemen] AMS), to support the Dutch government in determining their need for a new ship. The DMO is also responsible in ensuring that the determined need can result in a feasible and affordable ship. The resulting task for DMO is then to elucidate requirements for a new vessel. This process is supported by the creation of preliminary and concept designs (Figure 1.1). These resulting preliminary designs are generally not designed to be built. This process phase (concept exploration and definition) is called the preliminary design phase [15, 43]. In this phase the goal is to determine a set of requirements which can be set out to shipyards. From this set of requirements a design for a ship can be created by the shipyard.

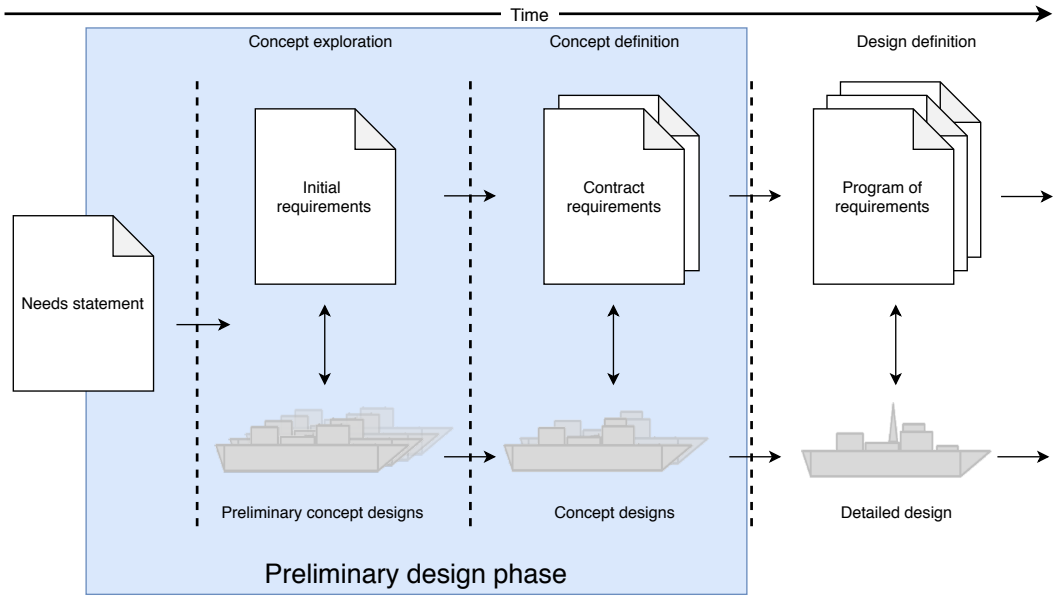


Figure 1.1: A simplified view of a design process, where requirements and designs are created and improved over time. A design in this figure is the combination of partial solutions and configuration.

To support the (complex) process of designing a new ship, DMO uses the Systems Engineering (SE) approach. This approach helps in providing a framework which can be used to create a design and also supports the structuring of created information in the process. In Figure 1.2 is shown what one (high level system) cycle in the design process might look like, based on Systems Engineering and the design process as adapted at DMO[1]. In the process the determination of feasibility and cost is called determining the performance and the determination of capability and affordability is called effectiveness. The cost of the conceptual ship is therefore determined separately from the affordability. The latter is the result of the cost in combination with the performance and capability of a ship. DMO wants to improve this process by having better ways of following information and changes during design iterations. By doing so they aim to improve the process of determining the balance between capability, feasibility and cost to support the Dutch government in their buying process.

1.2.1. The difficulty in this process

Finding a balance between capability, technical feasibility and cost relies on the relations between need, requirements and design. With the right requirements and design it is possible

[1]This is an interpretation based on the DMP as used by DMO [32]. Need = A, Requirements = B, Partial solution + design = C

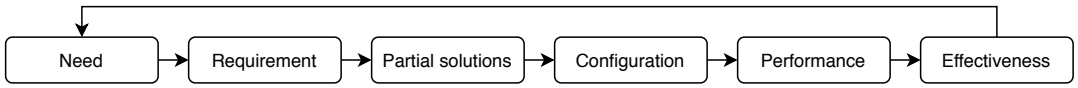


Figure 1.2: Steps in a design cycle, stating per step what is determined. Based on the process at DMO, which is built on SE theory

to represent the need for a naval vessel. However, determining the 'right' need, requirements and design for a ship can be seen as a 'wicked problem' as said by Andrews [1] [15]. Such 'wicked problems' have no definitive answer and can be redefined an resolved in different ways over time and according to whom interprets the problem [10]. This is because information on the solution is needed to understand the problem [37].

With this definition the need can be seen as part of the 'problem' and the requirements and design as part of the 'solution'. Since these now rely on each other for clear definitions the interactions between the need, requirements and design have to be understood as well. The line between 'problem' and 'solution' is blurred in wicked problems, meaning it is difficult to trace 'cause and effect'. To be able to do so, it is important to define and capture the relations between need, requirements and design. These relations can be used to trace information and find interactions.

1.3. The research

In current design practices the design process and design tools are well developed. However, traceability of relations between need, requirements and design remains problematic. While these relations can be defined with the help of design approaches, it is difficult to actually capture them [44]. It is this not being able to capture the relations that prevents traceability.

For this thesis the goal is to see how traceability between need, requirements and design in the DMO design process can be improved. This is to better understand, and optimise, the trade-off between capability, feasibility and cost. This traceability is being able to follow connections in information; these traces can help in analysing a system [28].

In order to realise this goal both the theory behind creating a design and the practical approach for this at DMO are analysed. During the process it was possible to generate and answer the research questions which led to this thesis.

1.3.1. Research questions

To reach the goal of this thesis the following main question has been stated:

> How can we capture the relations between need, requirements and design to improve the traceability of information on a naval vessel in the preliminary design phase?

To find an answer to this main question several research questions have been determined. With the problem now known, first the theory behind designing - and the relations between need, requirements and design - has to be discussed. After that the current practice at DMO is examined, with theory as a base line. The next question to be answered is how to use theory and practice to capture the relations, before looking into the improvement in traceability.

1. How does Systems Engineering (and Requirements Engineering) define the relations?
2. How are the relations currently managed during the design process at DMO?
3. What could be a method to be able to capture and manage these relations?
4. How does this method help in providing traceability, and what insight can be gained through this?

The answers to these questions are found in consecutive order in this document to work to an answer to the main question. The first question is answered in chapter 2 and is about the theory behind the design process. Capturing relations (with connections), question two, is about modelling ship design and is the subject of section 2.2. The third question is about the implementation of this theory at DMO and is discussed in chapter 3. With the proposed

method in this thesis in chapter 4 the fourth question is answered. In this chapter the theory behind the fifth research question is found, but the answer for the fifth research question is mostly the subject of chapter 5.

1.3.2. Research outline

In the beginning of the research the impact of having requirements and design separate in the process was investigated. One of the challenges that was found is that defining requirements is difficult enough to have its own research field: Requirement Engineering (RE). A separate field for the designing of (the configuration of) ships within the theory has not been found, but separate methods and theories for this have. The difficulty in managing the relation between requirements and design was therefore not only a problem encountered by DMO, but is present in the theory as well.

During the research on design tools and requirement management tools, the term 'traceability' was found. At the same time some approaches for 'model based development' were studied. The idea to build a model capable of managing relations between requirement and configuration came up, and became the goal of the research. This model would then be used to study if such relations management could work and if extra information could be captured within it. To be able to study this, a model first had to be designed and created. While there are some theories and approaches to create such data models, they are often still in development or privately owned. One program which DMO is currently implementing is Shipbuilder. This software is created to capture the (complex) data model of a ship during the design phase. Using this software created the possibility to store, manage and connect both requirements and design in one database.

During the course of the research it has been determined that going to "model based development" is a step in the direction of a improving [the information flow in] the design process [7, 17, 31, 45]. For these models several approaches were analysed: Axiomatic Design [17, 22], Design Structure Matrices [7], the Engineering Structure Matrix [4] and Knowledge Based Engineering ([26, 45]). In each approach the connection of information was the most applicable item for this research. Finding relevant research about the connection of information and the impact changes can have in this information through having connections, briefly led to looking into change impact of systems [9, 19]. While having the possibility to determine the impact of changes in the complex system is very useful, it is impossible without having a complete connected complex system in the first place. To capture a complete complex system in one model will require storing all known information and connections of a complex system. As this is - within the coming years - far from possible, it is the question how to take a step in this direction.

The research then moved towards finding out how a interactions of a complete, high level, complex system can be captured in one information model. With this question in mind the design process of DMO was yet again analysed, this time with the focus on information storage and flow.

2

Systems Engineering in ship design

In this chapter the first sub-question, 'How does Systems Engineering (and Requirements Engineering) define the relations?', is to be answered. The first part of this chapter goes into the theory of Systems Engineering (SE), applied on (naval) ships. It will focus on how Systems Engineering defines the relations between need, requirements and configuration. For this the design process is discussed in two parts: Requirements Engineering (RE) and configuration definition.

The second part of this chapter is about the shift towards model based development, or Model Based Systems Engineering (MBSE) [21]. The focus for these models will be to capture, manage and use the relations. One of the difficulties in using a computer to manage relations is that data for a computer has to be linked for a relation to exist. The architecture in which the data is linked in a computer has a great impact on the flexibility of the data. Having a too rigid architecture will help in maintaining overview in data, while a flexible architecture will better support changes in the data. The difficulty then is to have a solid, comprehensible, data model to support the relations while at the same-time requiring this data model to be flexible.

This chapter will only go into the theory behind design processes, the practical side of a design processes and the resulting limitations is the subject of chapter 3. In that chapter the theory is used to explain the process and problems which arise by using the theory will also be discussed.

2.1. Systems Engineering

The design process as pictured in Figure 1.2 is a good starting point to explain why this process is difficult. In the picture there are some relations which are missing. The first is the relation between performance and requirements. This relation is the direct result of the performance being the combination of requirements and design; this feasibility is about how and if the design fulfils the requirements. A second relation which is missing is the one going from design back to partial solutions. It is uncommon for a combination of partial solutions to be incorporated into one feasible design in one go. Going from partial solutions to one design is therefore also an iterative process in itself. To show these missing connections in the same picture as the design process could look as in Figure 2.1. The third missing relation is between effectiveness and need, or determining the capability [15, 43].

The resulting figure does not quite capture the information that has to be shown in it. To have a better figure showing the design process a new shape has been used in SE to picture the process: the V-model (Figure 2.2a). This V-model is used to visualise an iterative process going from determining to integrating and testing. By introducing a second dimension in the visualisation of the process it is possible to capture more information [25]. Not only is the cycle performed from left to right each iteration, it is now also visible that the 'lower' cycles are covered by the 'higher' cycles as well. Each 'lower' cycle is a part of a 'higher' cycle. Using this shape to visualise the design process in Figure 2.1 results in a representation as

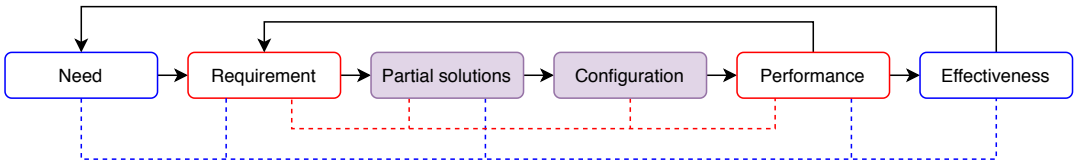
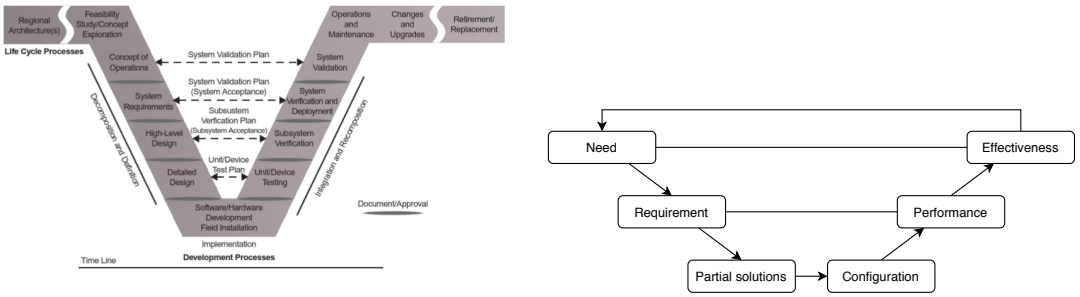


Figure 2.1: Connections in the design cycle. Blue is the ”need cycle”, red the ”requirements cycle” and purple the ”designing cycle”. The last cycle is subordinate to the first two in the preliminary design phase, therefore combining the two colours.

in Figure 2.2b.



(a) An example of a general V-model. From *Systems Engineering Principles and Practice* [25]

(b) The design cycle shaped to the V based on the process in 2.1

Figure 2.2: Two implementations of the V-model in SE

Using SE to structure the process led to more than only better visualisation of the process. It also helps in structuring the information model which is used to describe a ship. This information model will help to determine how a ship is tested to be suitable or not. This testing is about determining if a new design is capable, feasible and affordable. The terms creating and testing of a (complex) system will be used throughout the rest of the report; they are based on the terms *design* and *analysis* in Figure 2.3 [34]. With ”creating” in a complex system the steps of determining the requirements, partial solutions and design are meant. The term ”testing” in a complex system is about determining the fulfilment of effectiveness, performance and cost; or about determining capability, feasibility and affordability. In Figure 2.4 it is shown which steps in the design process are about creating (blue) and testing (red). The need in the figure (green) is said to be assumed fixed. While this is technically not the case in the preliminary design phase, it is not the task of DMO to determine this. They will help the Dutch government in shaping their need, which is still the starting point of a design cycle at DMO.

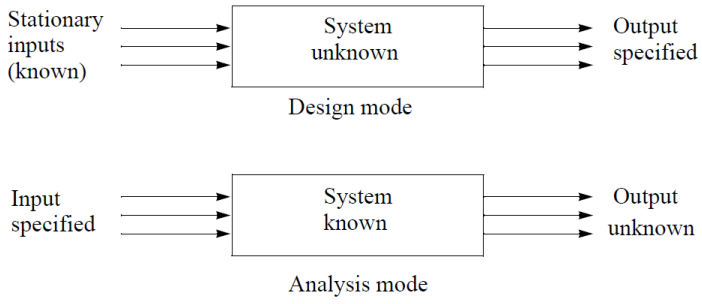


Figure 2.3: The difference between *creating* and *testing* a system. Based on the *design mode* and *analysis mode* (as seen) in the illustration from Pedersen and Engja [34]

Creating and testing a complex system now seems to be performed in that order in the

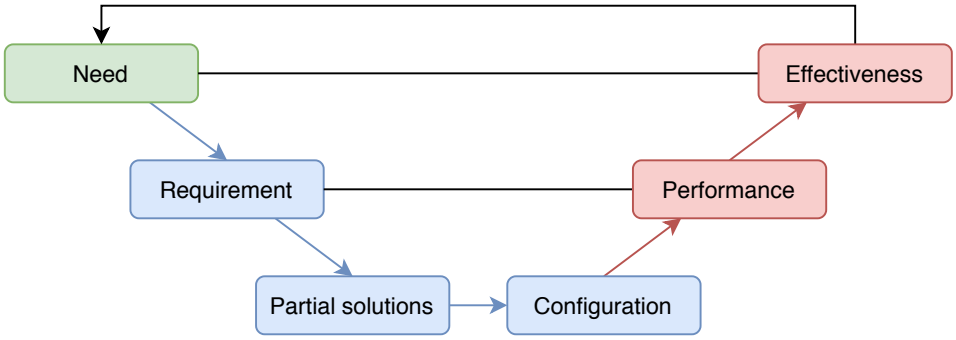


Figure 2.4: Show which steps in the design cycle are about creating (blue) and testing (red). The need (green) is assumed to be known.

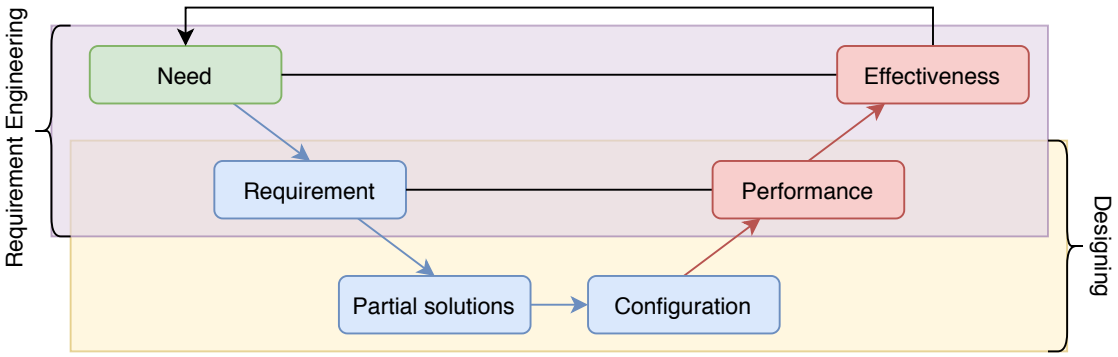


Figure 2.5: The design cycle (continued from Figure 2.4) where the focus of Requirement Engineering and designing of configuration is shown. Based on information from [47]

design cycle as pictured in Figure 2.4. Whereas this would probably lead to practical problems, it is also not entirely possible in theory as well. The step to create requirements - from only the need - has proven to be very difficult. Not only is this the reason that a preliminary design is determined in the first place, this difficulty in the process resulted in a new engineering field within SE: Requirements Engineering (RE) or requirement analysis. The goal of RE is to create, store, allocate and manage (supporting) information of a requirement during its life-cycle [47]. This can done irrespective of a design for the complex system but should not limit the design freedom to the extent that only one design is possible. In Figure 2.5 the purple section shows where the focus of RE lies with respect to the design cycle. This shift in focus in a design cycle, first requirements and then design, is also mentioned by Singer et al. [40].

The creation of partial solutions is the first step in creating a configuration design to fulfil the requirements. This is a step in the design cycle which relies on the creativity of engineers; here it is possible to apply the most innovative design solutions. Combining these partial solutions into one design, or configuration, is also a creative process, but is limited by the choice of partial solutions to incorporate. Creating a design from requirements is then iterated in itself as well to ensure the correct partial solutions are used. In Figure 2.5 the yellow section shows where the focus of ”designing” of a ship lies with respect to the design cycle.

Within SE a complex system is divided into separate pieces of which the following are relevant in order to structure the information in a preliminary design phase:

- Need
- Requirements
- Functions
- Systems
- Sub-systems

- Components [31, sec. 3.5.7]

By structuring information for a ship in the above mentioned parts, a ship is easier divided in partial solutions. These partial solutions do not always include a configuration in order to be designed, they can also be (high-level) functionally specified or described using a set of (design) information [17, 40]. Using this approach will also facilitate the 'configuration design cycle' by now only having to create the configuration aspects of partial solutions into one design. Before starting to visualise a design all necessary partial solutions could be determined functionally. This functional determination is the result of having suitable requirements which are derived from the need. How the creation of requirement works can be found in section 2.1.1 and further theory behind the creation of a design from requirements is found in section 2.1.4.

Having the separate parts in a ship and using them does not automatically result in a structured (information) model for the complex system; as a model is a representation of a thing [41]. While there is a thing, the ship, captured in several parts, there is no 'base model' for this ship connecting information. For this (information) model to be useful it is needed to connect parts to each other in a comprehensible way. What the benefits are of connecting the information in one computer model is the subject of section 2.2.

2.1.1. Requirements Engineering

Requirements Engineering (RE) is about creating, storing, allocating and managing (supporting) information of a requirement during its life-cycle [47]. The first step in a design cycle (Figure 2.4) is about determining requirements to capture the need. This step includes two important terms: need and requirements. This section is about determining what these terms mean, how they can be structured and how they can be managed. The definitions of need and requirement are taken as follows:

Need The need is the problem that a customer has which initiates the design of a ship. The need can be captured in "stated requirements" which are requirements given by the customer [47, p. 2]

Requirement A statement is an attribute in a system, a statement to help guide the design of a system, in a way that is useful for different stakeholders and applies value of the system to a customer or user [31, p. 94][47, p. 2]. A collection of requirements defining the constraints of a specific physical entity can be called a *specification* [8, p. 29]. For more information on this, and a breakdown of requirements, see Appendix A

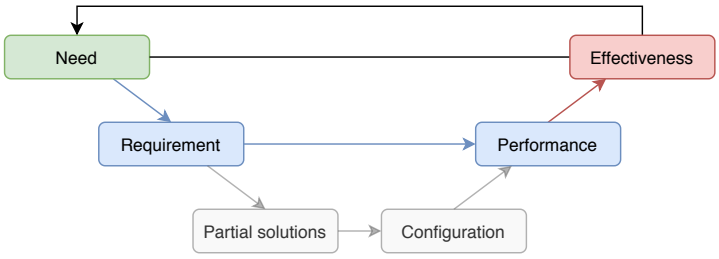


Figure 2.6: The part of the design cycle (Figure 2.4) focusing on creating requirements.

For this thesis the need comes from the Dutch Ministry of Defence, which is the problem owner. They state some requirements with which they try and capture their need for a new ship. These requirements can change slightly over time, they too have to be formed during the design process. A change of a requirement can actually lead to a better understanding of the need. Requirements help in capturing the need; they do not define the need [47]. Requirements themselves develop/evolve during the design process in a similar way

as a configuration develops/evolves during this process. While this change can help in understanding the project, it is required to keep track of these changes to keep order. For this *traceability* can be used, see Section 2.1.3.

Capturing the need for a ship can be done by using requirements. These requirements then state what the problem owner wants and does not want. The document where these requirements are printed is then the starting point for the design process (e.g. DMP-A brief [32]). It does not mean that there can be no changes in this document during the design process [47]. Especially in the preliminary design phase the need is still to be shaped. Saying then that the preliminary design phase is then only to elucidate requirements is a simplification as it is also about defining the need.

To still refer to the preliminary process as being focused on requirements, a distinction has to be made in requirements. For this the terms *stated* and *real* requirements can be used. Where the stated requirements are about capturing the need and are stated by the problem owner and the real requirements are those that reflect the need of a customer for a specific capability or system [47, p. 2]. The real requirements are the requirements as they are mentioned in the design cycle for this thesis.

The number of requirements is growing at a rapid pace in the beginning of a design process (Figure 2.7), while the knowledge about a design is still low [5, p. 45]. Not only are documents generated to capture the need, there are also multiple documents generated to capture the 'real' requirements as well. These latter documents are mainly used as starting point for designers, as the requirements in these documents are constraining their design freedom and design space. This transition between 'stated' to 'real' requirements is an interpretation of the 'need domain' and 'functional domain' in Axiomatic Design (see Section 2.2.2) [17]. Having to manage many documents adds to the complexity of having to manage the large number of requirements in them as well. Adding to the complexity of managing the shear amount of requirements is the number of stakeholders or large project teams [16, 31].

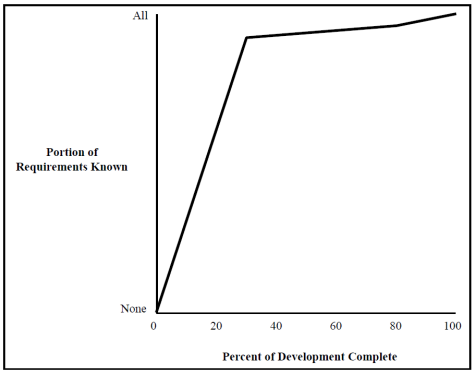


Figure 2.7: The number of requirements grows rapidly in the early design stage. (From Bernstein [5, p. 45])

Capturing the need and defining requirements prior to starting a design process may be subject to a similar iterative process of searching. When stating the need it is easier to say what you want, and more difficult to say what you do not want. This latter is due to you not knowing what you do not want at the beginning, because you do not know what is possible. Stating what one wants can lead to infinite possibilities to fulfil the need, the number is being constrained by possibilities having to be feasible as well. These constraints come from physical aspects (design constraints) and also from what the problem owner does not want. Describing a need with both what the problem owner wants and does not want will be a good starting point for a design. To have the best start this need has to be as compact as possible while still containing the essence of the problem. This means that both defining the need prior to a design process and designing a ship in itself can be seen as 'wicked problems', where a solution is required to define the problem [10, 37].

> *"Perfection is achieved not when there is nothing more to add, but when there is nothing left to take away."* - Antoine de Saint-Exupery

2.1.2. Implementing (ambiguous) requirements

Implementing the meaning of a requirement is an important task of an engineer. Take for example the requirement ”include a large number within this report”. While it seems easy to fulfil this requirement, the requirement is ambiguous. The definition of a large number can be fulfilled in at least two ways: 2 and 1e10 are both large numbers - in a different manner.

Since there is no further goal given to support this requirement it is impossible to conclude which of the two numbers is the proper implementation of the requirement. For an engineer the *rationale* (or context) for a requirement might help guide towards a (good) design [12]. It is important to store the supporting arguments supporting a requirement. Whether it is in a requirement document or via relations to other information should not matter; if the relations in a complex system are correct.

Not having very definitive or clear requirements can also have other unwanted results. One of these results is often called *requirement creep*. This is a change of scope or requirements as a result of engineers not being able to keep track of high-level requirements. Such requirement creep can lead to a project verifying all requirements while not meeting the needs of a problem owner.

Just as with the SE approach in a ship design process, requirements rely on context. While only some requirements are able to be implemented and validated without context, most requirements are better understood, and therefore supported, with context [24].

The insight gained through trying to relate requirements to context led to the following conclusion for this thesis: Having the option to connect requirements to one another it is less important for a requirement to be unambiguous. Ambiguity is created by a difference in interpretation which relies on context. By capturing the context in the relations to a requirement (object) ambiguity is counteracted.

2.1.3. Tracing requirements

Tracing information through requirements (for complex systems) has been of interest for researchers some time [28]. This documenting and managing of relations between (layers) of information is named *traceability*. Such traceability within a complex system can help to increase understanding of a design and make an impact analysis [14]. It can also help in understanding changes during a design project.

Using traceability to describe and follow the life of a requirement can help to communicate, integrate changes, preserve design knowledge and support quality assurance. Proper requirement traceability could also help a design team in finding looming problems in an earlier stage, as it can help identify these problems in the requirements [16].

Within the requirements engineering the traceability between entities can be seen as three-dimensional as can be seen in Figure 2.8. O Grady describes each type - axis - of traceability as follows:

- Vertical - the parent-child relation, this is the traceability between different levels of requirements
- Longitudinal(horizontal) - the traceability to a design and verification of a requirement
- Lateral - the traceability to methods or the rationale used for a requirement [31]

Or, as seen for this thesis, more general:

- Vertical - within the same ’domain’ (within a ship)
- Longitudinal(horizontal) - to (implementation in) another ’domain’ (within a ship)
- Lateral - supporting documents/methods/context (not necessarily within a ship)

The traceability information can best be generated simultaneously to the generation of the item it is used for. For this to work, time and money have to be invested in a system capable of creating, storing and managing a system which can capture both the item and traceability [31]. This means that some structure has to be created to be able to interpret all the generated information. Such structure can be called a *tracery* [20]. By having both the traceability information and a tracery it is possible to gain insight in requirements or a complex system.

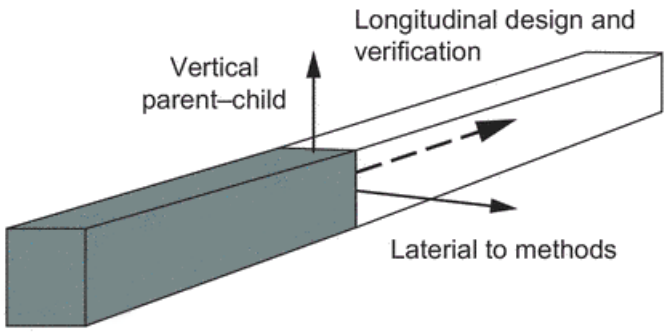


Figure 2.8: Three-dimensional traceability in requirements [31, p. 109]

Implementing traceability does not have to imply huge changes on a design process. Information on the relationships between entities is already used in the design process: each time a review of a design comes to pass [14]. Storing this information, having a capable program to do so and having a structure to interpret this information, is what traceability is about.

To see how such (3D) traceability could help, a small example will help. Take the specification section B 200 *Field-of-vision from within the wheelhouse* as stated by Det Norske Veritas (DNV) (in 2011) [13]. Within this specification there are several other specifications (and requirements) such as B 201. These can be interpreted as the 'children' of B 200. The specification section B 200 does also have a parent, the section 'B. *Bridge Design*', which in turn has a parent as well 'A. *General*' [13]. These are the vertical traces for the specification section B 200. Longitudinal and lateral traces are not mentioned in the same document.

Longitudinal and lateral traces, however, do apply to the specification section B 200. The lateral trace is knowledge present at DNV, and holds information on why the specifications are as designed. The longitudinal trace will be generated as soon as an engineer uses these rules to design a wheelhouse, both in requirements and in configuration. One of the longitudinal traces leads then to the 'design of the wheelhouse' and one to the 'verification' of this design. This trace is thus split into two.

2.1.4. Designing the configuration of a ship

To determine if a set of requirements describes a ship which is capable, feasible and affordable, it is useful to have some sort of design - ranging from a combination of parameters to detailed arrangements. This design may help in supporting the requirements and to find answers to uncertainties and assumptions [2]l. So, while in the preliminary design phase a design is indeed created, this design is not necessarily intended to be built: it is to support requirements and concept exploration [15, 43]. This then also leads to this preliminary design to incorporate assumptions and uncertainties.

Creating a preliminary design comes after the (initial) definition of requirements (Figure 2.9). Without a set of requirements guiding the preliminary design, the chance of a design fulfilling the requirements is nought. For this reason the defined requirements are taken as the starting point for a design. There are still assumptions and choices to be made during the creation of a preliminary design. These assumptions and choices are most likely the points of improvement in a next design iteration.

The preliminary design is the first translation in the design process between need and design, with the requirements as guideline and bridge. In a preliminary design the first geometrical shapes and physical aspects of a new ship can be determined and studied.

2.1.5. Iterating with Systems Engineering

Performing a requirement and designing cycle is part of the bigger design cycle covering a process from need to design to effectiveness. The resulting design cycle is therefore more complex than pictured in Figure 2.2b.

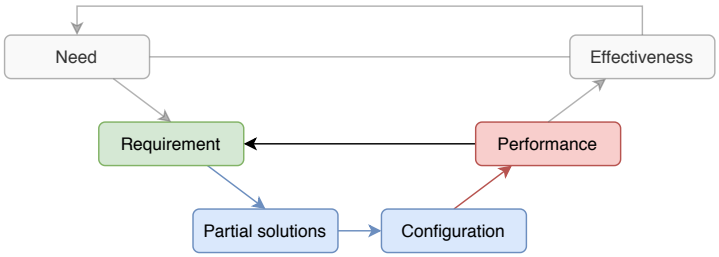


Figure 2.9: The part of the design cycle (Figure 2.4) focusing on creating a design for a ship.

The iterative process can be divided in four stages in this thesis:

1 Create (/improve)
2 Test
3 Gain insight
4 Propose changes

The creating and testing stages have been discussed briefly in the first part of section 2.1. During these stages in the ship process, information and relations are defined at first and later tested for compliance with the initial need. Understanding of the relations, is needed if (quick) insight has to be gained on a design [14]. This insight can be used to propose changes or learn from a design. In the iterative process, the insight is mostly used to find improvements in a design.

As Systems Engineering is used to define the steps in a design process, a V-model has been used to visualise what happens in each of the four stages in Figure 2.10. This V-model acts as a tracery to understand information flows in a (high-level) ship in the stages.

These steps iterate during one design phase. Such phase can be 'concept exploration', 'concept definition' or 'design definition', see Figure 1.1. At the start of a new phase more detail is added to a design. For the first phase this detail is actually just the first information.

In more detail the four stages can be described as follows:

Create In this stage a ship is created, meaning that information about the ship is defined. Here the requirements, partial solutions and configurations are defined - according to the level of detail in a design phase. In the V-model, this part is about defining need to design and about defining when such design is effective and performs.

Test The created ship from the previous stage is now *tested*. Now the capability, feasibility and affordability are determined. This testing is about seeing if constraints and requirements are fulfilled; why they are fulfilled as they are is not yet of importance. In the V-model, this part is about determining (testing) the performance and effectiveness of a ship.

Gain insight In this stage insight in a ship is sought, based on existing information in the design process. This can either be to find out why the test results are as found or to learn from a design for future projects. In the V-model, this stage goes 'through' the V-model backwards. It is about finding out how the result came to be, and why this happened.

Propose changes This stage is about determining what changes to the ship can be applied to ensure the next test phase to not repeat an unwanted result. Here change propagation can be helpful to quickly see the impact of a change, however this relies on information being (actively) connected [9, 19, 29]. In the V-model, this part is about determining what, and where, can be changed to change the results.

Iterating the design of a ship using these stages could help in structuring and understanding information flows in a design process. It is a way of thinking used by the author of this thesis to later specify the use of a new method to 'connect' requirements to configuration. These stages help to identify where the most difficult part in gaining the insight in a ship design lies.

2.2. Model based design

Systems Engineering, and the design cycle (Figure 2.2b), is created to support a (ship) design process through structuring information [25]. Within the design cycle are other iterative processes: defining requirements and defining configuration (see Figure 2.5). This adds to the complexity of iterating according to the V-model during a design cycle. With the information for a ship being created, stored and managed with the help of the V-model structure, it should be possible to find information for a ship with relative ease. However, this has proven to be more difficult in practice. Capturing the information concerning a "complete V" means storing and managing a vast amount of information. Whereas this first led to information being separated in documents and later in databases, computers can currently help in creating a model for this information. This is a change in the design process from being a document based development towards model based development [31, p. 81].

The model based approach resulted in approaches such as the Design Structure Matrix (DSM), Domain Mapping Matrix (DMM) and Engineering Structure Matrix (ESM) [4, 7]. Another approach that has been developed for this purpose is Axiomatic Design [17, 22].

The matrix based models (DSM, DMM, ESM) aim to create an overview of relations in the information in a design process [7]. Axiomatic Design is a more model based approach, aiming to connect domains of a system [17]; which can be used to create overviews such as in matrix based approaches [17, sec. 1.2.5]. These approaches have been researched for some time now, and as research progressed new tools improving these methods have been developed . Some of these tools aim to improve traceability of information in the models. One of these tools is Capra, which creates a framework to be able to trace information in a model [27, 28]. However, traceability programs such as these still require a connected information model; meaning that a computer can see and interpret relations between information.

Tracing information in a design has been done with Design Matrices and Axiomatic design giving information on the relation between entities but no information on the underlying structures of those entities [20].

Another approach to modelling a design process and complex system is Knowledge Based Engineering (KBE). The goal of KBE is to create tools which can manage knowledge in a design process [12, 45]. The focus of these KBE tools is then on the structure and the connections of all the information in a design process, to capture knowledge and have models able to perform design steps. Having a model of the design process, supporting processes, knowledge and tools, is therefore a large part of the KBE field [39].

2.2.1. Matrix frameworks

One way to increase the insight in a ship design and related data while at the same time make the information comprehensible for engineers are matrices. In a matrix it is possible to show connections or relations between entities by marking the associated column and row. This is the basis for the Design Structure Matrix (DSM) (Figure 2.11), which was introduced to gain more insight in the architecture of a ship design [7, 11]. According to Browning [7] "the DSM brings the advantages of simplicity and conciseness in representation, and, supported by appropriate analysis, can also highlight the important patterns in system architectures (i.e., design structures), such as modules and cycles". In the traceability problem the visualisation of found 'traces' or relations is important to keep information comprehensible.

From the DSM, new types of frameworks evolved to further increase the gained insight in systems: the Design Mapping Matrices (DMM) and Multi Domain Matrix (MDM) (Figure 2.12)[7]. These DMMs are capable of connecting information in multiple domains as the name suggests. In some cases a DMM has been used to connect functions to components [7]; this imposes that this approach could be applied for traceability. Bartolomei et al. [4] proposes

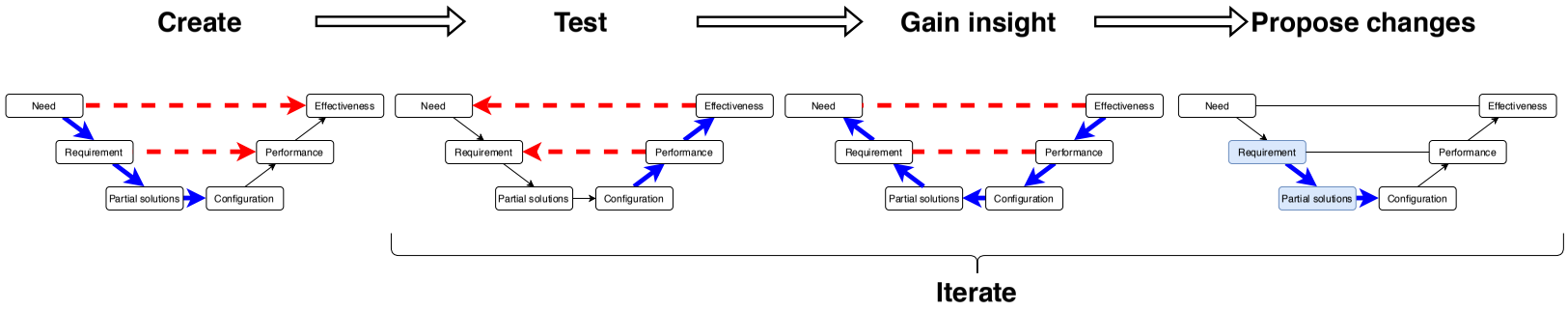


Figure 2.10: The information flow ”in” the V-model during a design cycle. Blue arrows are defining relations which are able to be directly interpreted. Red arrows are used for testing and can be indirect relations; these relations often rely on context.

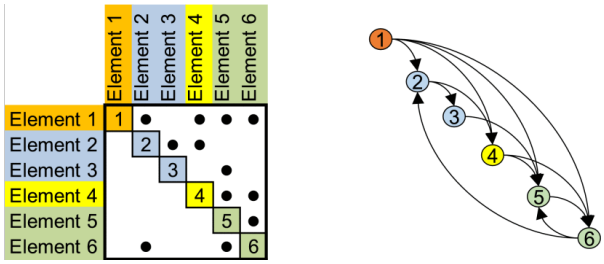


Figure 2.11: An example of a Design Structure Matrix, and its equivalent node-link diagram (from [7])

a new approach for the evolution of modelling frameworks: the Engineering Systems Matrix (ESM) (Figure 2.13).

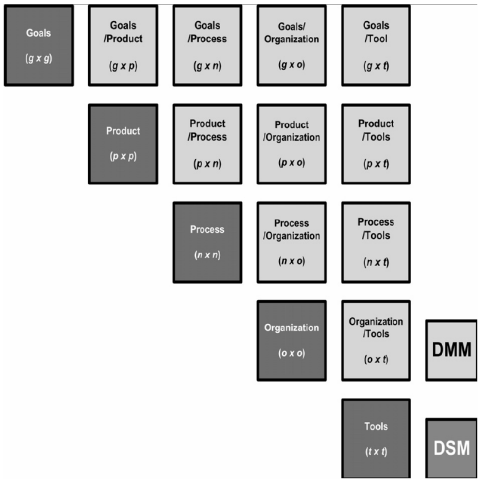


Figure 2.12: A MDM framework comprised of DSMs and DMMs (from [11])

The definition of Axiomatic design (AD) is that it will connect the functional and physical domain [22]. Using this method can prove helpful in many occasions and it has been used in some occasions to connect requirements to design parameters [7]. However the data used in the approach can be stored in several ways. It is possible to use this approach - and data - separate from a design process; for example, it can be used to validate decisions.

Bonjour et al. [6] use a DMM to couple functions to systems in order to create a complex system structure. As it is vital to have relations in a complex system, the DMM and DSM can be used to visualise and manage these. However, these matrices are currently limited in managing and visualising large quantities of (complex) design data. Underlying data models are becoming more complex and more difficult to manage; which is a fundamental problem in improving matrix based models (and other approaches) [7]. With a smart database application, such as Shipbuilder, it is possible to manage relations within a complex system as

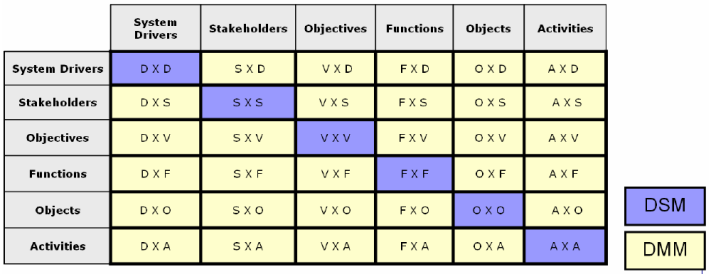


Figure 2.13: The structure of an ESM, decomposed to DSMs and DMMs (from [4])

well as store additional data on each element. By using such databases for the information model, it is likely that matrix based models can help in gaining insight in this information [7].

Matrix based structures have also been applied to change propagation in complex systems [9, 19, 29]. This is another way of gaining insight in the system a (matrix based) model describes.

2.2.2. Axiomatic design

Axiomatic design aims to connect the physical and functional aspects of a design through coupling Design Parameters to Functional Requirements. This connection is one of the connections between the four domains in Axiomatic Design: Stakeholder requirements, Functional architecture, physical architecture and process architecture (Figure 2.14) [17]. This coupling is an important step in understanding the development of a (ship) design. It can therefore be seen as a method connecting RE to configuration design. While configuration design can also implement the functional aspects of a design, these aspects are generally only a few specific requirements of a system. In RE the coupling between the first two domains are covered.

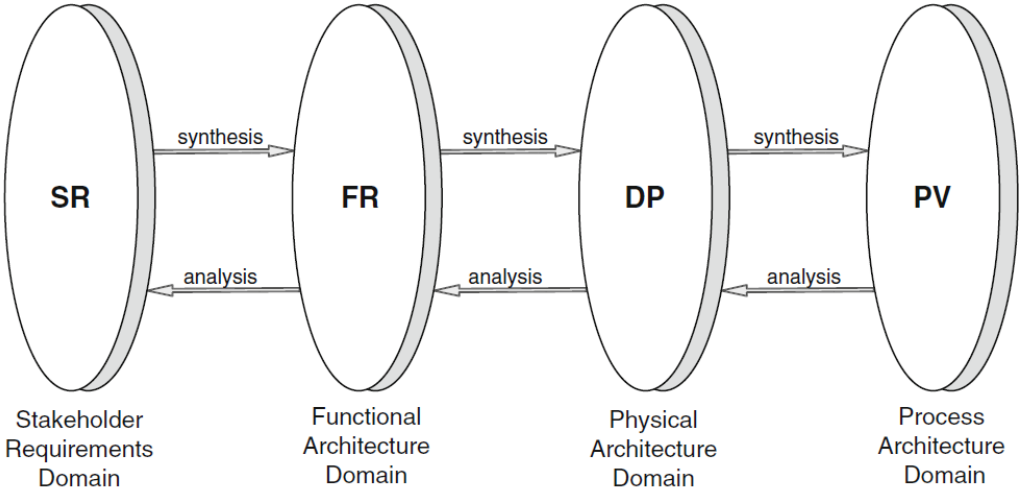


Figure 2.14: The four domains in the Axiomatic Design perspective (from [17])

Axiomatic design is built on two axioms (hence the name): the independence and information axiom. The independence axiom is about mapping relations in a way that one DP corresponds to one FR and the information axiom is about minimising the amount of information required for a design [17, 22]. Axiomatic design was first applied in the Mechanical Engineering field, later it expanded to other disciplines as well [17].

In Axiomatic design a design in synthesised in a so called ”zig-zag” approach (Figure 2.15). In this approach first the high-level parameters are determined across domains, after that lower-level parameters are determined for the ’starting domain’. By using zig-zagging and mapping a design can be defined in different levels of detail [17, 42].

The use of Axiomatic design does help engineers in gaining insight and structure information in a marine system [22]. For this thesis the basics of Axiomatic Design can be applied to manage the relations between need, requirements and configuration.

2.2.3. Knowledge Based Design

Knowledge Based Engineering (KBE) is mostly used in combination with CAD applications, or at least this is were the idea started [26, 35, 45]. KBE was thought to help engineers to develop a design in a CAD program by having knowledge stored in the computer. This stored knowledge would support, correct or predict design solutions. An other aspect of the KBE development is that corporate knowledge can be stored, thus being less dependent on that

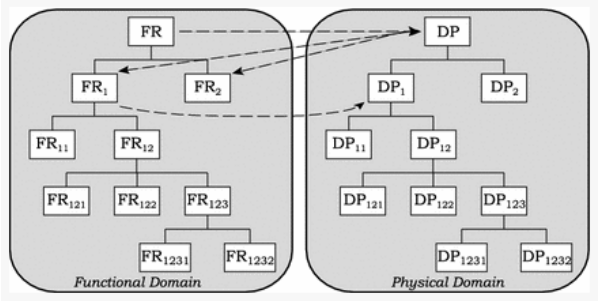


Figure 2.15: Synthesis according to the zig-zag approach in the functional and physical domain (from [17])

'one senior engineer' with all the knowledge and losing parts of this when the engineer leaves the company.

In the ideal design process key design decisions will be made as late as possible, as this generally means that the decision can be based on more knowledge. This view can be seen in the articles by Verhagen et al. [45] and Mavris and DeLaurentis [30]. Figure 2.16 from Verhagen et al. [45] illustrates what what the goal of KBE is: increase design freedom, delay committed cost and advance (design) knowledge in a process. Mavris and DeLaurentis [30] describe the same problems and goals in his research. The idea here is that by creating a general approach for a design process, through applications, it is possible to reduce cost and duration of projects while getting better results. One of the encountered problems is that most information is not available for all relevant parties, not that the information is non-existent. This problem can be limited by using a single database to store design data; having a way of easily gaining insight from this data even further.

Where Mavris and DeLaurentis are looking from the design perspective of aeroplanes, Verhagen et al. are taking on the problem from a broader perspective. Early KBE systems exists since the 1970's, but only since the 1990's has the development of such systems gained increased interest. This is the result of acknowledging the difficulty of knowledge models and the corresponding research required for further development and use [39, sec. 2.1]. KBE aims to use (corporate) knowledge to create artificial intelligent programs which can design products. The ultimate goal is that such a system will design a product, create all necessary data and documents, make the production process and check products against industry rules. In order to create such systems it is required to divide products in as generally applicable parts as possible. Reusing these parts will speed up a design process as a lot of information is thereby already known in later products [26].

In KBE the DIKW (Data, Information, Knowledge, Wisdom) pyramid is used to define knowledge (Figure 2.17, which has been used for several years to structure the 'wisdom hierarchy'. In the paper from Rowley [38] it is found that there are fairly clear definitions of 'data' and 'information':

Data Data items are elementary representations of observations which are meaningless without context.

Information Information is processed data. It adds meaning and structure to data, mostly aimed to be understood by human interpretation.

Knowledge Knowledge is a collection of information and data. It can lead to understanding, learning and being able to relate information.

Wisdom Wisdom can be a result of knowledge and knowing what to do with this. It is mostly a human skill but has no general, clear, definition.

The definition of the higher levels in the pyramid are less obvious and differ between researchers [38]. However, it is at this level that it becomes interesting to be able to let a computer help engineers. If a computer is able to work with, and manipulate, information and data it could be called 'advanced knowledge modelling' [39].

To have an idea of how a DIKW structure could help, the following example is given:

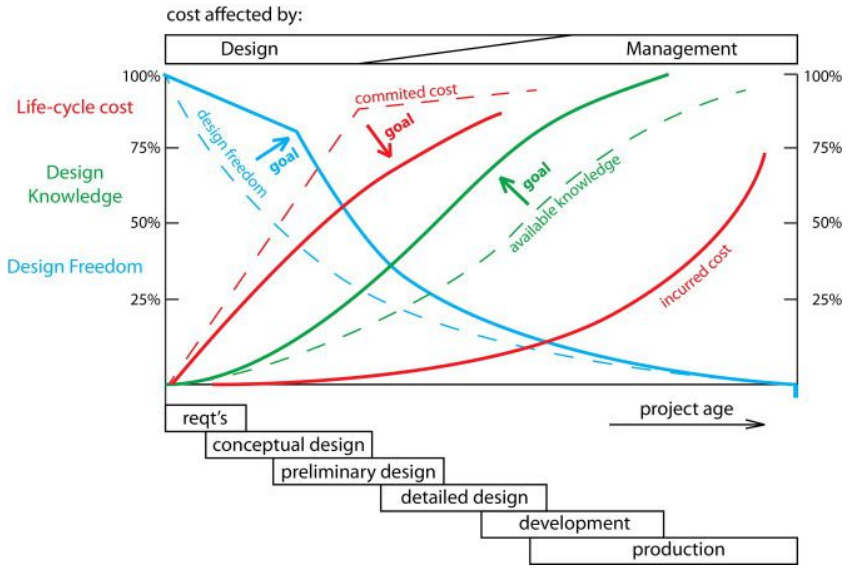


Figure 2.16: Product life-cycle cost, design knowledge and freedom related to design process. By Verhagen et al. [45]

- Data: 128
- Information: the length in metres of a S-frigate
- Knowledge: the L-frigate is equally as long, as it is built on the same hull
- Wisdom: the length of these frigates is the result of a design process for a 'standard frigate' and suited the requirements at that time [23, 46]

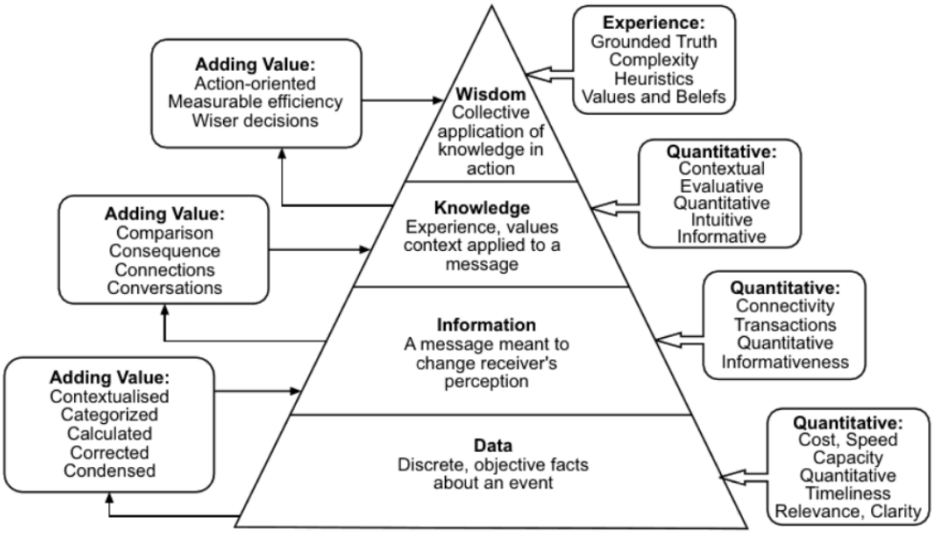


Figure 2.17: Relations between data, information, knowledge and wisdom (from [3]).

With the help of the DIKW structure, one of the goals of KBE is to have a knowledge model to help engineers with tedious jobs [45]. As there is knowledge in this field in creating and managing complex models (containing and relying on context), a KBE system might be helpful in connecting requirements to configuration.

2.2.4. Moving to model based development

Using an information model in the design process of a ship could make it easier to store and manage relations between information in a design. By using one information model to store

information some indirect relations can actually be managed. Having two separate information models - and databases - can rely on implied relations [17, 27, 31, 41]. Generally one information model is manged in a database or program. Information models crossing several databases or programs can be difficult to manage as this requires programs to (actively) communicate.

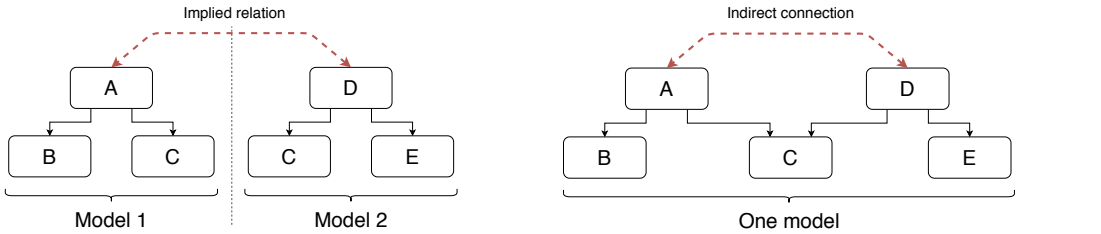
Take for example system A in Figure 2.18a. System A has to be supported by components B and C, and is stored and managed in database 1. Now take another system, D, which has to be supported by C and E, and is managed in database 2. Both systems A and D thus have to be supported by component C: there is an indirect relation between A and D. By using separate databases to manage A and D this indirect relation can become an implied relation. This is because component C is in both databases. While C remains the same component, it is now managed by two separate data entries. As these data entries both refer to the same component, but are not the same object, there is now an implied relation between A and D, based on the assumption that component object C in database 1 is the same component as object C in database 2.

Such implied relations are common in information models in a ship. Same requirements are stated in the ConOps, Program of Requirements and entry fields in design programs, yet these requirements are not the same object. It is possible to manage these implied relations through the knowledge of engineers working on the project and documents storing information for these implied relations (references between documents/databases). However, finding information in this construction can be tedious and errors can occur by interpreting information differently.

As finding information in data models based on separate databases can be difficult or tedious, a solution could be found by using a combined database - or model. In a combined database it is possible to actually connect pieces of information (data) to each other.

For the example above, using one model to manage both systems A and D will provide the possibility to only have one data object for component C (Figure 2.18b). As a result the implied relation between A and D is now an indirect connection, captured in the actual relations between A and C and C and D.

Having the implied relations captured through other relations in a model can help in gaining (quicker) insight in the information. This is the underlying principle to proposing the stated method.



(a) A representation of an indirect (implied) relation between two objects in different information models (each in a database)

(b) A representation of an indirect relation (connection) between two objects in one information model (one database)

Figure 2.18: One of the aspects in relations which is covered by moving to a model based development process

2.3. Conclusion

With the help of Systems Engineering and the 'sub-field' Requirements Engineering a good framework supporting the development of a ship has been created. By using SE and RE it is possible to structure information on a ship design as well as define this information. As the discussed frameworks were all high-level, similar to the level of detail found in the preliminary design phase. For this reason the ship as a whole will be taken as the complex system in this thesis, systems such as propulsion engines will then be 'components'.

Relations between need, requirements and design are part of the defining structures for information in both SE and RE. Where in SE the decomposition is often more focused on the 'design' part of a complex system, the RE is about the structure of the requirements for

such system [31, 47]. It is in RE that the term traceability has been introduced [14]. This traceability is the result of using computers in the design process, or going to model-based development.

In the model-based developments there are programs and approaches created to help in defining and managing relations. With the Design Structure Matrix it is possible to visualise connections in a ship, which helps in understanding relations [7]. Another approach is called Axiomatic Design, which helps in defining relations and complex breakdown structures. Here the decomposition of a ship can be structured across domains and in different levels of detail [17]. To see what such relations are capable of, and what more can be gained from them, the research field of Knowledge Based Engineering has been developed. This field aims to capture 'knowledge', which is a level above 'information', to aid engineers in developing designs [39]. All of these approaches benefit from having a solid information source. In a solid data source relations should be stored and managed, enabling digital traceability. While the relations are well defined in theory, and they can be used in model-based approaches, it remains difficult to create a single information source to capture the relations.

The issue that exists in the design process at DMO is that it is tedious and difficult to trace information between need, requirements and design. With the underlying theory and possible solutions to manage and use these now known, the next step is to evaluate the current design approach at DMO. This will be done in the next chapter.

3

The DMO design process

Without the ability to trace relations between need, requirements and design it is impossible to create a capable, feasible and affordable ship. Yet, at DMO they are capable of creating a ship which is capable, feasible and affordable, without this traceability. This means that the relations between need, requirements and design are present and known in their design process. How they are currently defined and managed is therefore analysed in this chapter to answer the second sub-question: 'How are the relations currently managed during the design process at DMO?'.

To find the answer to this question, first the theoretical side of the DMO design process will be discussed, before the practical implementation is described. This will give a good view of how the relations should be and are currently managed in their design process. Here, a division between requirements engineering and designing can also be seen, just as in theory in the previous chapter. This division is the result of programs not being able to cover all information and them being used in different design stages.

By having both the information from theory and practice it is possible to see what could be used to improve traceability. Some requirements are stated to be fulfilled by a new method in order to improve traceability. How such method can look like is discussed in the next chapter.

3.1. The theory behind the DMO design process

Acquiring new materiel for the Netherlands Ministry of Defence follows the "Defensie Materieel Process" (DMP) [32]. This process comprises of five phases:

A - Statement of requirement and budget
The A phase is about stating the initial need for a ship. This is about the 'what' and 'why'. In this phase the first operational requirements are determined and a budget is defined. The Dutch parliament has to approve a project by looking at the results of this phase.

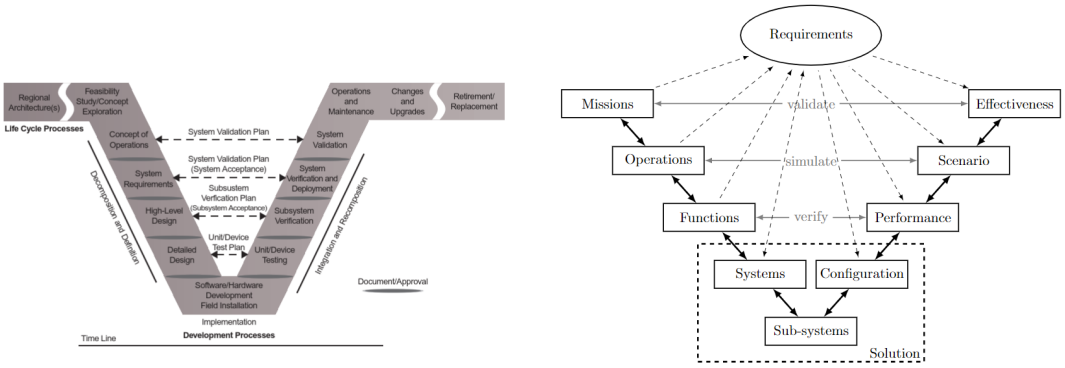
B - Study-phase After aproval, DMO starts a project to acquire a ship. In this phase more detailed operational, functional and some technical requirements are determined. This is supported by different preliminary designs, in order to determine which alternative is capable, feasible and affordable.

C - Detailed study & development phase
If at the end of the B-phase it is determined that there are no off-the-shelf ships available, this more detailed study phase is started. This is generally the case for a warship [44]. During this phase the functional and technical requirements are determined in more detail. With this a more detailed design is made.

D - Procurement preparation
This phase is to determine a contract specification which is used to put out a quote to the industry. After approval of the parliament a contract is signed and a ship can be built.

E - Evaluation Large or complex projects are evaluated after some years of service.

Supporting this DMP, the design process at DMO is based on the Total Ship System Engineering Process of the NATO, which is supported by their own V-model (Figure 3.1b) [44]. Compared to a general V-model (Figure 3.1a), the biggest difference is that the "requirements" are not in the V itself.



(a) A generic V-model in SE [25]

(b) The V-model as used by DMO for a ship. This model is proposed by Van Oers [43] and adapted by Duchateau [15]

Figure 3.1: The process to the V-model

One of the reasons that the V-model as used by DMO is different compared to a more general V-model (Figure 3.1a), is that this V-model is used to support the elucidation of requirements. A design of a ship then is not leading in the process, it should reflect the requirements to test them on describing capability, feasibility and cost.

Using the System Life Cycle from Kossiakoff et al. [25], the design process at DMO covers the "concept development phase" and a part of the "engineering development phase". The projection of the design phases and DMP documents on a timeline is seen in Figure 3.2. This figure shows a difference in DMO (blue) or a shipyard (grey) being the leading stakeholder in the design process. During this process, the DMP is used to inform the Dutch politics of the progress. In the figure the documents used for this are shown as the 'decision points' in a process, where one phase changes to the consecutive phase.

In the lower part of Figure 3.2 there are several V-models shown. This has been done to indicate that during each phase the same V-model is used to define all design aspects of a ship in the process. However, in each design phase the focus within this process is on different parts of the V-model. This focus is highlighted by circles; as used before the colour blue indicates what is 'created' and the colour red indicates what is 'tested'. While the focus in these stages is on these aspects, the other aspects in the V-model are also defined. These aspects are mostly assumed or determined in low detail to support findings of the highlighted aspects.

In the first two stages the focus is on capturing the need, setting the requirements and determining some functionalities. During these phases the scenarios and effectiveness of the 'ship' are tested. In the third and fourth design phase, the focus shifts towards the lower part of the V-model. This is to indicate that in these phases the focus lies in creating a configuration (or solution) for the design problem. Here the performance is the leading 'test' for a ship. However, during each phase the V-model is used in total as the preliminary design phase (Concept design phase and the advanced development phase in Figure 3.2) is still about elucidating requirements for a complete ship.

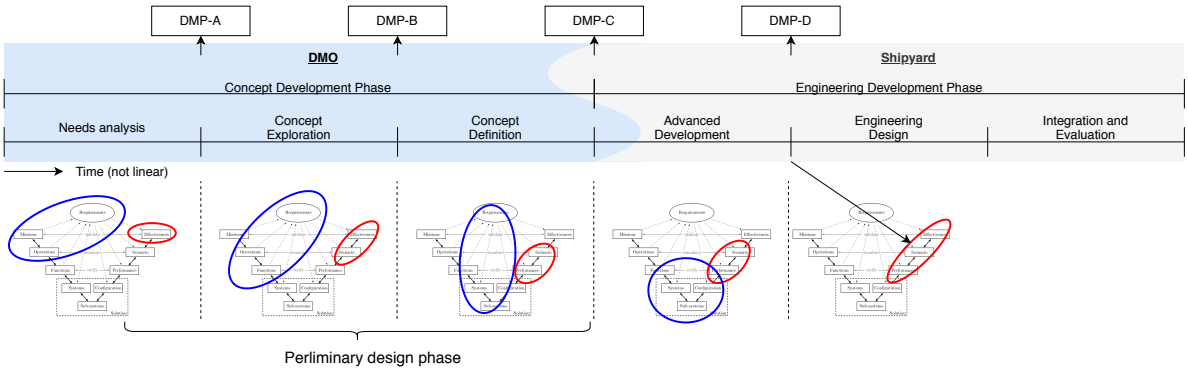


Figure 3.2: A system life cycle as seen in the DMO design process, with points indicating the DMP documents, and the V-model highlighting the points of interest. The highlighted ovals are blue for 'creating' and red for 'testing' parts of the V-model. Based on information in [15, 25, 32, 44]

3.2. The implementation of the DMO design process

The implementation of their theoretical design cycle model (Figure 3.1b) is limited due to practical limitations. These limitations lead to the process not being as streamlined as hoped, this is where the issue starting this thesis initiated. One of this limitations is that there are multiple stakeholders needed in the process to support the input and generation of information, which is a standard issue within System Engineering [36]. Storing, managing and using this information is also divided. This is the result of a ship containing a vast amount of information, which covers many aspects of a design (e.g. functional, operational, hydrodynamics). This leads to several programs used to manage the information [44].

Both these limitations have one side-effect, it blurs the line between requirements and design. A design is created to test the need and requirements before the DMP-A and B briefs, after which the need and requirements are supposed do be as good as fixed - to some level[32]. Only if there is no possible design able to fit the need and requirements these two can be altered. Therefore it is important to know where in the process a choice, trade-off, decision or assumption has been made. Finding this information becomes tedious by having multiple stakeholders and programs, which could lead to an uncertainty regarding having to change the need, requirements or design; thus blurring the lines between them.

3.2.1. Stakeholders

In the DMP there are three main stakeholders: Defence staff (DPLAN), the Royal Netherlands Navy (CZSK) and DMO [44]. However, there are more stakeholders in the design process of a warship. In total there are at least seven stakeholders which have impact on the design process in the preliminary design phase. Who these stakeholders are, and what their interest in the design process is, is stated below.

Defence Staff (DPLAN)

DPLAN is responsible for the functional requirements and the budget; for the functional requirements they get support of DMO.

The Netherlands Royal Navy (CZSK)

CZSK is responsible for the operational requirements and the concept of operations. They are also the user of the new ship and also take care of the maintenance of a ship; this is done by a separate section "Directie Materiële Instandhouding" (DMI).

Naval Maintenance and Sustainment Agency (DMI)

DMI is the responsible unit for the maintenance of the ships and systems of CZSK. With this they have knowledge on current used systems, which may be valuable information in a design process.

Defence Materiel Organisation (DMO)
DMO is responsible for the materiel of the Dutch Ministry of Defence. Within DMO there is a section responsible for naval systems. Their main task in the DMP is to determine the technical requirements and they are responsible for the procurement of the ship.

"Specialist" Within DMO there are multiple "specialists". For this thesis these "specialists" are engineers which are specialised on a certain field (e.g. marine engineering, hydrodynamics, weapon systems). The interest of a specialist is thereby only on a part of the model of a ship as proposed in this thesis.

Shipyard A shipyard does not have direct influence on, or responsibilities in the DMP. However, keeping the shipyard in mind during the design process could help structure a design to ensure feasibility of the design.

NATO The NATO has rules and regulations for new warships. These rules affect the design process of a ship. Furthermore, as the NATO conducts missions with the aid of national forces, the NATO has to be informed during the design process.

Here it can be seen that all the stakeholders are, to some extent, responsible for the requirements of a new ship. Only DMO is responsible for a design of a ship. Not all stakeholders play an equally large role in each of the DMP phases. For this thesis these stakeholders are mentioned to show that it is not only DMO which is concerned with the design of a warship. The other stakeholders also have interest and input in this process. This makes the design process more complicated [36].

3.2.2. Programs

The relevant programs for this thesis are FIDES, PACKING, QUAESTOR, Rhinoceros and DOORS, as they are used during (different phases) the design process of a ship at DMO.

DOORS IBM's DOORS is a requirement management program, used at DMO.

PACKING PACKING is a bin-packing model used to generate and change a ship model in the design exploration phase. It is able to generate a large amount of high-level ship designs, which can be used to study trade-offs.

FIDES FIDES (Functional Integrated Design Exploration of Ships) is used in the design definition phase. With this program, and the link with Rhinoceros, it is possible to generate 3D models of a ship, with variable levels of detail.

Rhinoceros A 3D-CAD program used to visualise the design of a ship. The input for a 3D model are the blocks as defined in FIDES.

QUAESTOR This is a knowledge management shell, managing a numerical model of a ship. FIDES can be used to generate input for this program. This program is also used to couple several other programs, such as SARC's stability program PIAS and MARIN's SHIPMO seakeeping code [44].

Shipbuilder Shipbuilder is an application currently used as a requirement management program in one project. This program is a semantic database capable of storing information of a ship [18].

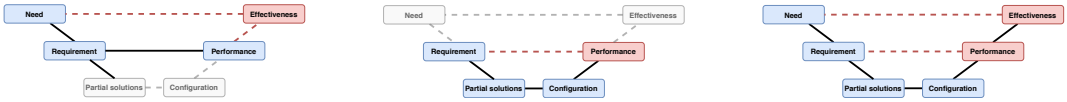
Table 3.1 shows where in the DMP each program is used. Here it can be seen that there are two programs used during the whole process, DOORS and Shipbuilder. Both these programs are currently used as requirement management tools. For the design of a ship, several programs are used. PACKING is used for concept exploration in DMP-B, and the other design programs are mainly used in the DMP-C phase.

In this table, the software tool Shipbuilder is shown to be used during a complete process for requirements engineering. This program, however, is also capable to be used as a

design tool. This program can then be used for this purpose as well during the design process, already partially closing the gap between information management of requirements and design.

Program	Need definition	Concept exploration	Concept definition
DOORS	x	x	x
Shipbuilder	x	x	x
PACKING		x	
QUAESTOR			x
FIDES			x
Rhinoceros			x

Table 3.1: Programs used in different DMP phases, determined from Van Oers et al. [44]



(a) The parts of the design process supported by DOORS

(b) The parts of the configuration design process supported by FIDES, PACKING, QUAESTOR and Rhinoceros

(c) The scope of the Shipbuilder software, applied to the model in this thesis

Figure 3.3: Parts of the design process and their supporting programs. Similar to Figure 2.4, blue indicates what is 'created' and red indicates what is 'tested'. Dashed lines are indirect relations, solid lines are connections

In Figure 3.3 the discussed programs are shown with respect to their focus on the design process. The division between Requirements Engineering (Figure 3.3a) and designing a configurations (Figure 3.3b) can be seen in these figures. Only DOORS, and in a new project Shipbuilder, is used for the requirement management. The other discussed programs are used to define a configuration. As such configuration is used to represent the requirements there should be a connection between the two. Such connection is currently often an indirect relation, which has to be interpreted by an engineer. There is currently no program used to cover the information of both requirements and configuration. Shipbuilder is capable of doing this, but is now only used for RE. When DMO wants to move to model based design, this program can prove to be very useful, that is one of the reasons it was chosen.

The division between requirements and configuration design tools results in 'implied relations' between the two. There is no connection between items on either side of this division. The consequences of this is are discussed in Section 2.2.4.

Shipbuilder

To improve the flow of information DMO is implementing the Shipbuilder software in the workflow of one of their projects. Shipbuilder is a web based data management tool based on the SE approach. With this their goal is to make the information in a ship more transparent.

Shipbuilder software is build on the database platform Relatics and is dedicated to the maritime industry. The configuration of a ship and the project are stored in a semantic network based on a maritime knowledge base. Knowledge is stored based on entities, parameters, relations and constraints. The software is supporting the Systems Engineering method [18].

The scope of the Shipbuilder software is to create and manage a ship in one program. This includes both the requirement and configuration parts of a design process. The ship is a high-level complex system, meaning that systems such as engines can be considered as a component of the ship. The software is built to be adaptable, making it possible to incorporate the design approach as used by DMO. This possibility for adaption is one of the benefits of using a model based approach, such as Shipbuilder [18].

As Shipbuilder is already applied at DMO, the information model in Shipbuilder has been adapted to fit the process at DMO. This was the information model which was encountered at the beginning of this research. The current use of the software is to create the initial set

of requirements the relations between requirements and relations, and relations and other objects in a ship are more important than relations between objects in a ship.

For support in designing a ship, Shipbuilder already implements some information structure and management to support this process. The software is built to also support the structural design of a ship. For this purpose the arrangement of components is done by adding spaces and decks to a ship. These components, spaces and decks are supported by the construction of a ship. This construction is not a part of this thesis, as it is also not a part of the preliminary design phase. The required level of detail in a ship design is too high. The use of spaces and decks, however, is suitable to be used for this thesis. With these two objects it is possible to (partially) define the configuration of a ship. Spaces and decks are also useful in placing components, and indicating placement of systems, in a configuration [18]. The information model used for this configuration is seen in Figure 3.4. In this figure it is shown with dashed lines that there are indirect connections between a space and a system and between a component and a deck. Having the option to see these relations is useful for engineers, as this increases the context for an object. For this thesis, this information model is not part of the scope. However, this information model does make it possible to connect the 'configuration' of a ship to the rest of the information. The information model for a configuration can also be used when a ship is visualised in a 3D CAD program.

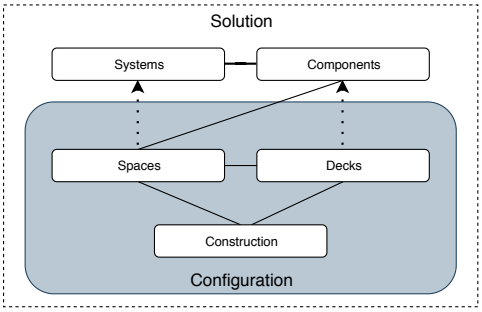


Figure 3.4: A possible interpretation of the information model for the configuration in Shipbuilder, applied to the 'configuration' and 'solution' as found in the V-model of DMO. This part of a design is therefore covered in the Shipbuilder software

Using Shipbuilder as a leading program in the design process, means also the 'knowledge management shell' of QUAESTOR needs to be replaced. This can be done by using the 'Knowledge Base' (KB) of Shipbuilder. The KB can be used to store general information (knowledge) for the ship design process. As this KB is implemented in the software it is possible to directly apply stored information to a new ship [18, 44].

3.3. A small example of a naval ship

To give a small example of a naval vessel, using SE, may help to identify why it can be difficult to have traceability in the information for a ship. In the case study performed for this thesis (chapter 5) the example of a frigate, based on the standard frigates once used by the Dutch Navy, is taken. The standard frigate came in two variants: the S-frigate and L-frigate. The first is an anti-submarine warfare vessel and the latter an anti-aircraft warfare vessel. More details on these vessels can be found in chapter 5.

To examine the use of the Requirement Engineering and Systems Engineering as described above, it is useful to create a small example. For this example the steps performed as in the design cycle will be followed.

The example is short and is only to indicate how the terms mentioned above can be used. For this means there is only one stated requirement, two requirements and one function used as output from the "defining requirements" cycle. For the design only one, still high-level, partial solution is mentioned to fulfil the requirements. In the design this partial solution is used to indicate how this system can in another complex system take the role of a component.

As there is knowledge on how to build, operate and use ships the result stated at the effectiveness will generally be tackled in the first design meetings. It is still a good indication of how the need can become both more clear as well as change during the course of a process.

Need

Stated requirement: Have a ship capable of providing submarine defence at sea.

Requirements

Two requirements will be stated, supporting the need (represented as a stated requirement) above. These requirements are both high-level system requirements.

- The ship shall be able to defend itself from nearby submarine threats
- The ship shall have a system to take out a submarine

Functions

Disable hostile submarine close to the ship

Partial solution

To fulfil the function, one of the possible solutions is to fire a torpedo from the ship. This is a self-defence system.

- System: Torpedo launch tube
- Sub-system: Pressurised air system
- Component: Control pad of the torpedo tube

Design

For the sake of this example only one partial solution has been used. This results in the design only having one partial solution. Generally, a design is the integration of several partial solutions, each solving different requirements.

- System: Self-defence system
- Sub-system: Torpedo launch system
- Component: Torpedo launch tube

Performance

The ship shall able to disable a submarine effectively within the range of X nautical miles from the ship in conditions until sea state Y.

Effectiveness

The ship is able to provide submarine defence for itself against a nearby hostile submarine, with a survival chance of 90%.

This provided defence however, is only for the ship itself. If this vessel would be a true ASW frigate it should be able to provide support for other ships as well. As this is not stated in the initial need, a new requirement has to be added to the ”need”; the current requirements do not have to be changed.

In this example, all the information is located in one location. This enables the user to quickly identify what the change in specifying the need would indicate: change the solution fulfilling the function. This can either mean to change the system, or add another system. To have an answer to which of these two options is better, it would be logical to also include the cost of such systems. In the example the cost is left aside, but the cost generally plays a large role in decision making[44]. In a standard design process the budget, however, is set. So let’s assume that the ship is currently on the cost limit, so adding a new system would be too expensive. This assumption, with the argument is now a driver for the design choice which has to be made. If later on in the process there is more budget, this could mean that two systems for the function could also work. However, if the assumption on the cost is not properly stored, the solution of one system for the function is no longer a variable but a set solution.

So, information on the budget and cost should now also be included in the breakdown above. This changing of stored information, and input, creates difficulty in tracing information through a design process if there is a lot of information involved.

3.4. Combining theory and practice

Looking back at the process using a V-model (Figure 2.10) and projecting the practical approach at DMO on this a new figure can be made. This is Figure 3.5. In this figure the V-model is used as an information model, defining a ship. The problem they encounter in these steps is that the current information model is distributed over several programs, whereas there can only be one ship. With the approaches as described in this chapter and looking at this figure it can be seen that the process of creating a ship is very complex and can be confusing. This is the result of the process being divided in itself and practical limitations of programs amplifying this.

The issue which started this thesis - the difficulty in tracking, and back-tracking, information - can be seen in this figure. The difficulty in tracking the information in the process has several causes:

- The complexity of a ship
- Uncertainties in the design process [2, 44]
- Separate processes (see Section 2.1)
- Relying on interpretation (see Section 2.2.4)
- Separate programs (see Section 3.2.2)

None of these causes individually are preventing the information flow. However, relying on engineers to find separate pieces of information and interpreting implied relations themselves is time consuming and is considered a tedious job. It is the tediousness of finding each piece of information and creating the traces that is considered a problem, this could be solved by using a tool capable of traceability management [14, 31, 44]. To decrease the time used to trace information the underlying problem has to be solved. This problem lies in the storage and management of information and relations. Solving this problem can lead to improvements which can reduce the impact of the above mentioned issues.

By using the Shipbuilder software the segregation in the ”designing cycle” is already mostly covered. Using this software creates a possibility to connect requirements to design. As external programs can be coupled to this software to both read and create information, a complete model of a ship should be possible to be created [18]. The remaining question is then to find out if the model used in this software can be changed to further implement the theoretical model of a design cycle. The design cycle based on literature is mostly covered in Shipbuilder, but there are some differences between this model and the design cycle used at DMO.

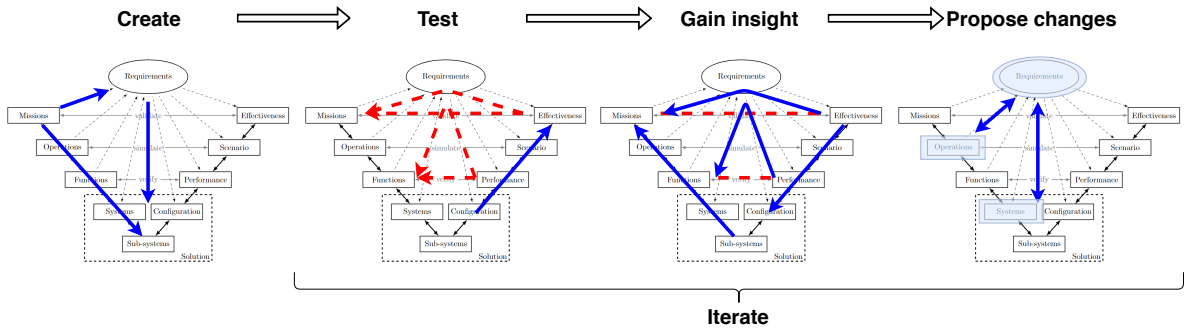


Figure 3.5: The information flow ”in” the V-model used by DMO during a design cycle. Blue are information flows for defining (or creating), red are information flows for testing.

3.5. Possible improvements

To solve the issue DMO encounters in tracing information throughout a preliminary ship design a solution could be found in actually connecting the separate items present in their V-model (Figure 3.1b). By solving the problem of having one ’definitive’ database to store and manage information, issues preventing traceability could be resolved. Implementing

the Shipbuilder software seems to move towards a solution to their problem, as it is based on Systems Engineering and creates the possibility to create one database for important information.

As Shipbuilder can be seen as a KBE program, and is focused on storing and managing (transparent) information of a ship, it should then be possible to build a model of a ship in this program. This model should be built in a way that is implementing both theory - SE, already in Shipbuilder - and practice - V-model of DMO. The last part will be the key to helping DMO to solve the issue they have with finding information in their process.

To ensure that using the Shipbuilder software in combination with the DMO design approach will improve traceability, some requirements have been stated:

- Quick access to information
- Quick insight in the gained information
- Have the ability to apply changes in the information model
- Use the software for different ships [44]

These four requirements are already partially fulfilled by DMO using the Shipbuilder software. Having quick access to the information and making this transparent is the goal of Shipbuilder, so checking these requirements should come as no surprise. Since the Shipbuilder software is Object Oriented the information model that is used can be changed, but most importantly the information in this model can be changed without losing the model. However, there are still possible improvements to be made.

Quick access to information For the first requirement - quick information - the missing connections take away the possibility to create traces from the configuration to mission. So another requirement has to be added to the first requirement(which is a stated one to capture the need): *Quick creation in a trace from need to component and back).*

Quick insight in the gained information This requirement is fulfilled through the possibility to add background information about an information entry in Shipbuilder. This background information (e.g. source or rationale) can help in gaining insight on an object. Yet, as the connections as seen in the V-model are not all present, the lack of some of the connections could hinder an engineer in seeing the reason why the model incorporates a certain connection. This will lead to not having insight in the overview of the information. To solve this problem two requirements are added to the second requirement: *Store information in the model which tells how information can be interpreted* and *Use an information architecture fitting the DMO approach of a design cycle.*

Have the ability to apply changes in the information model This third requirement can use some extra information as well. Here, only having the option to make changes to information in the model can actually slow the traceability of information later on. This is because it might not be known what the impact of this change is or will be. The direct impact of a change is difficult to define, this is the subject of the research field Change propagation [9, 19]. Still, having the option to see what a change in the model might do is helpful for engineers in going through iterations. This insight is built on having correct traces for information, having insight in these traces and having a known structure to compare the traces and information to. These are the extra requirements as stated above. An extra requirement then added to the third requirement, and incorporating all other requirements, could read: *Quick insight in the impact of changes in the model.*

Use the information model for different ships This requirement is applicable as concept exploration is a part of the preliminary design phase. Exploring different concepts, can be interpreted as designing different ships. In the Shipbuilder software it is possible to create different ships. These ships can contain information stored in the Knowledge Base. However, this does not directly result in a comprehensible information model, this has to be created in Shipbuilder by the user. As it is not defined what a design looks like at the beginning of a design process - neither is a design defined at the end of the preliminary design phase - the method should be able to work for 'any' ship design. The actual requirement for the method will then read: *Be reusable for different ships*

With these extra requirements in mind there are now five requirements to be fulfilled in order to get to a solution for the issue of insight in information. All the requirements, possible solutions and fulfilment are stated in table 3.2.

	Need	Solution	
1	Use an information architecture fitting the DMO approach of a design cycle	Use the same information structure as the V-model used by DMO	±
2	Quick creation in a trace from need to component and back	A program capable of creating these traces using the existing information	?
3	Store information in the model which tells how information can be interpreted	Link information in a way which can be used to interpret the information and links	?
4	Quick insight in the impact of changes in the model	Use stored information and relations to identify the impact of changes	?
5	Be reusable for different ships	Have an information model capable of describing different ships	?

Table 3.2: Requirements that may help solving the issue of gaining quick information from a ship model

3.6. Conclusion

The theoretical design approach at DMO, which incorporates Systems Engineering, defines the relations between need, requirements and design. However, in the practical implementation a division between requirements engineering and designing can be seen. This creates difficulty in tracing information from requirements to design, while in theory these relations between both are defined. The difficulty in traceability is then not part of the design process itself, it lies in the practical implementations of computer programs. Having one single source to store information could create the possibility to improve traceability.

By solving the requirements set in the section above it should be possible to create a method which will improve this traceability. The main issue that has to be solved is to use one database for information and connections with a comprehensive structure. This will most likely improve traceability; opening the way to resolve issues which exist because of the difficulty of determining cause and effect.

How a method fulfilling the stated requirements can look like is described in the next chapter. The proposed method will be shown and explained. Difficulties encountered with implementing this method are also discussed.

4

A new method

Using an information model, covering need, requirements and design, in one database will help to capture and manage the relations between these three aspects. This will help to improve traceability. In this chapter, the answer to the third sub-question 'What could be a method to be able to capture and manage these relations?' is answered by the previous sentences.

The method in this chapter is created to fulfil the requirements mentioned earlier. In this chapter the outline of the method is discussed and the building and early implementation of the method is described. This will not directly lead to fulfilling the set requirements. In order to see if the method actually does this a case study has been performed, this is the subject of the next chapter.

4.1. The proposed method

To fulfil the requirements stated in the previous chapter in table 3.2 the following method is stated for this thesis:

> Build a model of a ship in Shipbuilder, following the structure of the V-model as used by DMO, to capture relations between configuration and requirements to improve traceability in the design process

The steps in examining this method are the following:

1. 'Visualise' the information model in Shipbuilder, according to the V-model
2. Change the information structure to represent the V-model
3. Create a model of a ship with the improved database model
4. Show traceability between information of requirements and configuration

The first two steps for this method will be discussed in this chapter, the last two are the subject of the case study in Chapter 5. In Section 4.2 first the proposed information structure is visualised and related to the current Shipbuilder structure. Changes to this structure to represent the proposed information model can be seen in Section 4.3.

The objective of the method is to support information traceability and the creation of insight in this information. This is done so that in the insight phase the best possible changes can be proposed. By having good traceability in the information of a complex system it should be possible to determine what changes are possible and which are not. The traceability should lead to finding an uncertainty, assumption or choice in the design which can be changed. Finding information on where and how to change a design to improve it - while not imposing a new problem - is then the goal of the method.

For such traceability to be possible, the easiest way is to use one program to model the ship. Why this can be useful has been discussed in Section 2.2.4.

The information model, following the V-model of DMO, will look as pictured in Figure 4.1. In this figure the information model exists of the objects shown in white, the yellow blocks are activities to evaluate the complex system. The complex system then comprises of objects and the activities produce new information (in documents or databases) stating the test results.

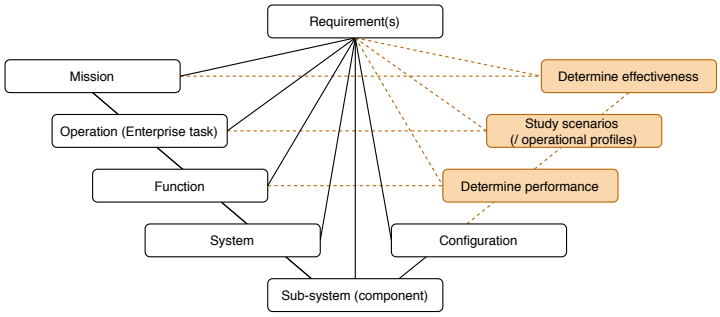


Figure 4.1: The proposed information model, based on the V-model by Duchateau [15]. The yellow objects are not really objects in the information model, they are more activities resulting in documents which contain information.

4.2. Visualising the information model in Shipbuilder

To see how the current information model (structure) relates to the proposed model, the structure in Shipbuilder has been visualised to represent the V-model (Figure 4.2. This figure shows some differences related to the proposed model. One difference is that not all connections between the shown items are present, the other is not directly visible but is shown with the red marking of the 'function' object.

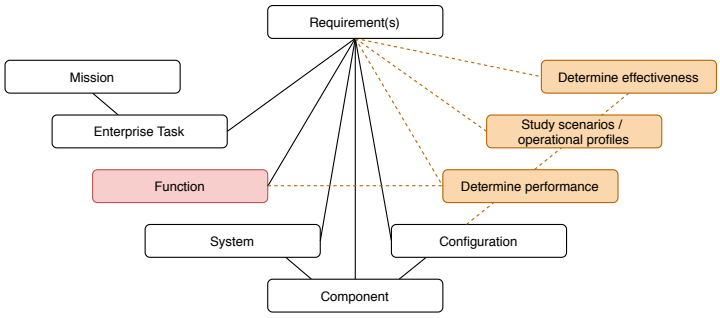


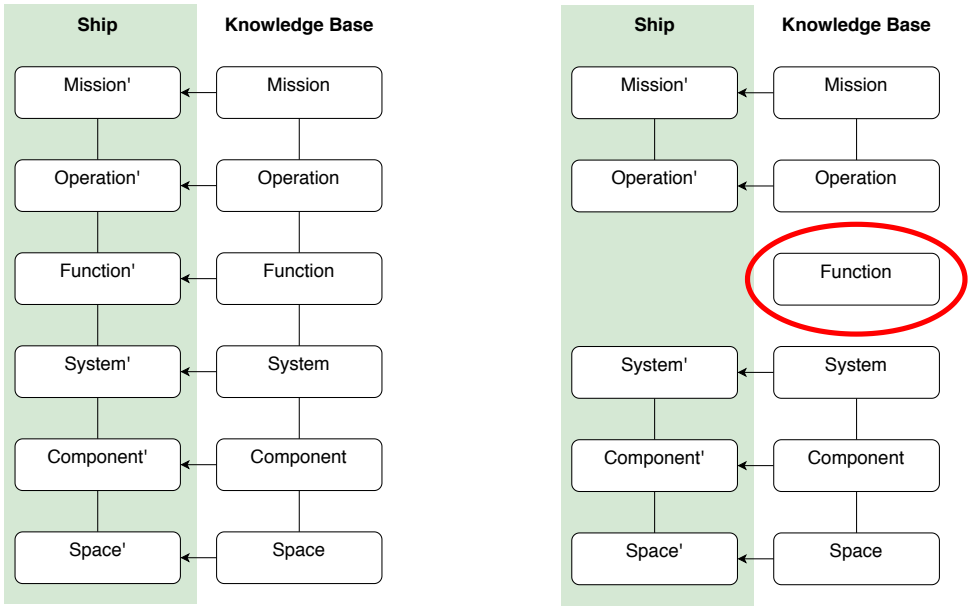
Figure 4.2: The information model in Shipbuilder using the V-model as reference. The 'function' is red as this object is only present in the Knowledge Base and not in the model of a ship

4.2.1. Implementation of the 'functions' object

For all but one object that has been discussed in the method it is currently possible to apply this object to a ship. Only for the 'function' object this is not possible. The problem this poses however can be seen in Figure 4.3b. Here it can be seen that all but one object has been placed in the green field. This green field represents the boundary of a ship. Within this field information is specific for the ship, outside this field is non-specific information. Such division is helpful to prevent clutter in the KB.

In the red circle in Figure 4.3b is a function object. This object is not placed in the field of the ship but is connected to objects in the ship. This should not prevent tracing information through a model, so it does not pose a big problem for understanding the relations in a ship. If functions are specified specifically enough in the database this could prove to be no problem at all. However, such solution will undermine the thought of the KB being the place to store general information as the stored information is specific for one (type of) ship. In addition to that, the relation between a system in a ship and a function (for the ship) is now only indirect. This is the result of a function only being in the KB. Having this function only in

the KB currently implies that a relation can only exist in this KB, not outside of it. Meaning, that a function can be connected to a system object. The system in a ship is an instance of such a system object, it is not the object itself. Now, there is no connection between the instance of the system object and the function, only between the parent object of the system and the function.



(a) The proposed implementation of the model in the software.

(b) The initial implementation of the model in the Shipbuilder. Here it can be seen that a 'function' is not implemented in a ship, and has no connections to other objects

Figure 4.3: Both the proposed and initial implementation of the model of a ship (green) in Shipbuilder as built from objects from the Knowledge Base.

One of the problems that arises by not resolving this problem is that the line between 'knowledge' and 'project information' blurs. With the current size of the models of the ship this is not a problem, but having large projects will be hindered by this problem. If someone thinks a function has to be altered for one ship, the current implementation will also change the functions for all other connected ships. Having a 'base function' in the KB and implementing 'specific functions' for a ship will prevent this. This structure can also be seen in the requirements in the Shipbuilder software. Here are also 'requirements' and 'ship specific requirements', of which the latter is built on a 'requirement' but is altered to fit the structure of one specific ship.

Solving the current implementation through defining each function to be specific enough to be only used by one ship, may result in the database for functions becoming too large.

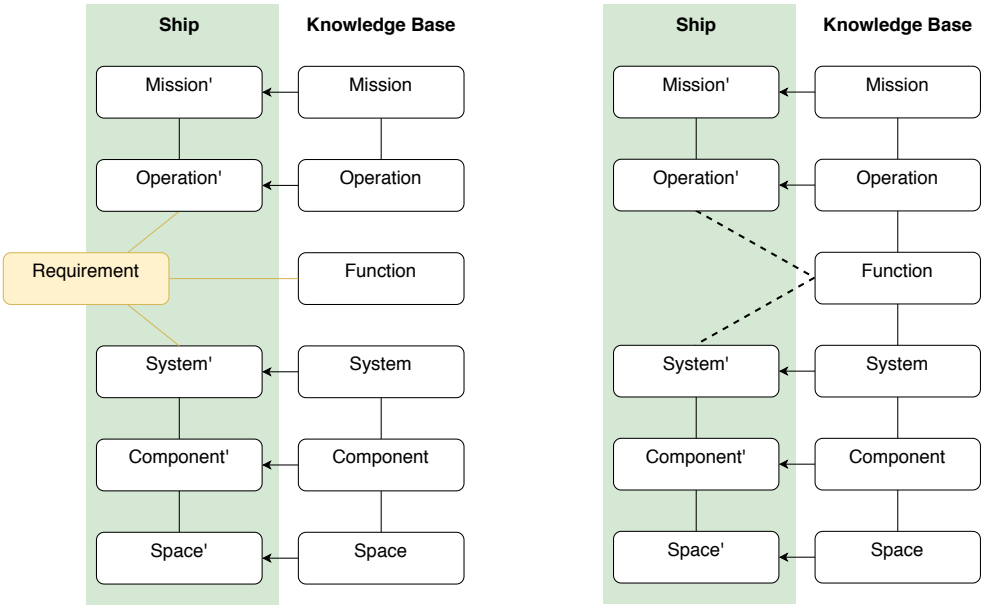
4.3. Changing the information structure

The found information model in Shipbuilder is missing some connections compared to the proposed model (Figure 4.1). The connections for the 'functions' have an extra difficulty due to the current implementation of this object, as discussed above.

To be able to test the working of the proposed method, and later implement it, this model has to be connected. Therefore are the missing connections to be created. The missing connections are:

1. Mission - requirements
2. Enterprise task - function
3. Function - system

By implementing these connections, some indirect connections as shown in the information model will also be able to be implemented. These connections are more the result of the



(a) With help of requirements a function can be connected to a ship

(b) The final implementation of a function at the end of this thesis. A function can now be (indirectly) connected to objects in a ship model

Figure 4.4: The way a function is currently connected in the information model in Shipbuilder. The dotted lines are indirect connections, related through requirements

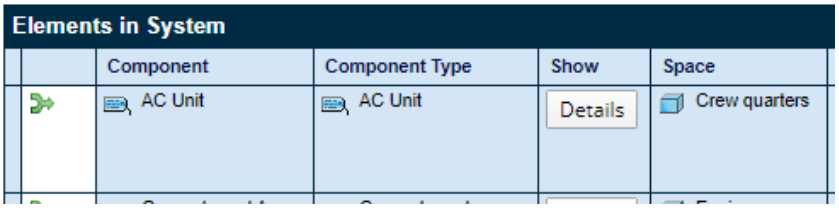


Figure 4.5: The list of components ('elements') in the electrical system of the S-frigate. In this list the space connected to a component is also shown, this is an indirect connection from the system to this space

connections between a requirement, an object and this object in the configuration (applied object).

In the Shipbuilder software there is also an interface. With this interface it is possible to create and manage the data. To do this there are visual fields to show what connections are present for a certain object. Most of the connections of an object are visualised in the interface, and even some indirect connections can be shown. An example for the latter is the connection to a space in the interface for a system (Figure 4.5). A system in itself is not bounded by one space, but the components in this system are. As the components in a system are shown in the interface, the relation between this component and a space is shown with it. While this does not have a direct impact on the design of a system, it does give valuable insight in the application of this system for engineers.

The final implementation of the information model in Shipbuilder is not exactly as proposed. An overview of the final connections in the information model can be found in Appendix C. The first missing connection has not been implemented. This is the result of two reasons: 1 - only in a late stage in the research it became clear for all involved parties what exactly was this connection and 2 - the current implementation of the object mission has practical limitations. These practical limitations meant that, for the time being, the connection was not implemented along with adding more information to this object as well. However, this will be solved if DMO and Shipbuilder agree on the working of the model.

Such practical limitations are also present for the object functions. For the best implementation of this object, the structure of the underlying information model and this object

itself have to be changed. As this is a time consuming task, and a quick work around (adding the connections) did work, this limitation has not been resolved. Since it was still possible to add some missing connections to the object, this did not result in a problem.

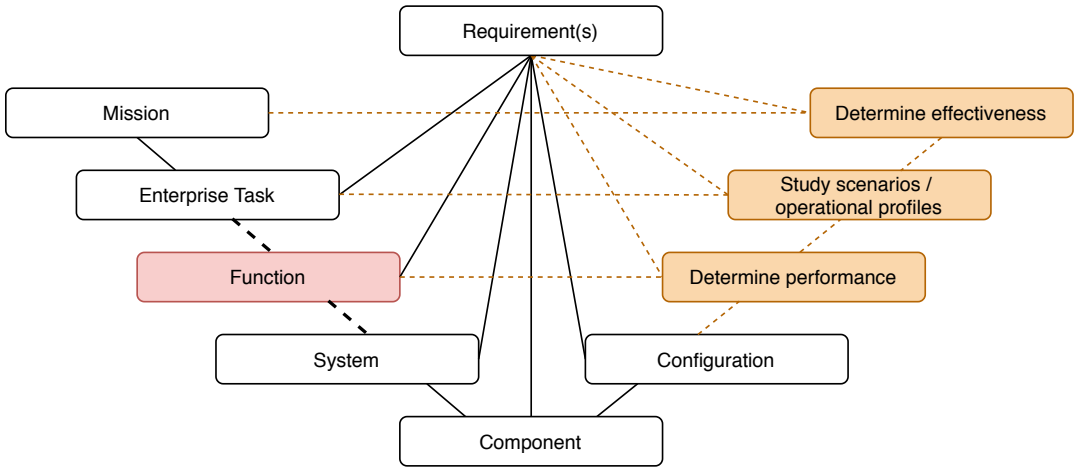


Figure 4.6: The final version of the model at the end of this thesis. Yellow objects are activities and the red object cannot yet be placed in a ship.

4.4. What does the method solve

As stated at the beginning of this chapter, the goal of this method is to provide traceability in information of a complex system. The main issue to solve is to close the gap between information storage and management between requirements and design. Defining requirements and creating a design can be performed separately by assuming information in the other process. These assumptions are generally the subject of discussion and possible points of change. Getting quicker to these points of attention, and having insight information on these at the same time, may be possible with the proposed method.

This method is then to be used to create a model for a ship which can be used in each of the design steps in the process V (Figure 4.7). It therefore fulfils requirement 3 of the stated requirements in Section 3.5 (see Table 4.1).

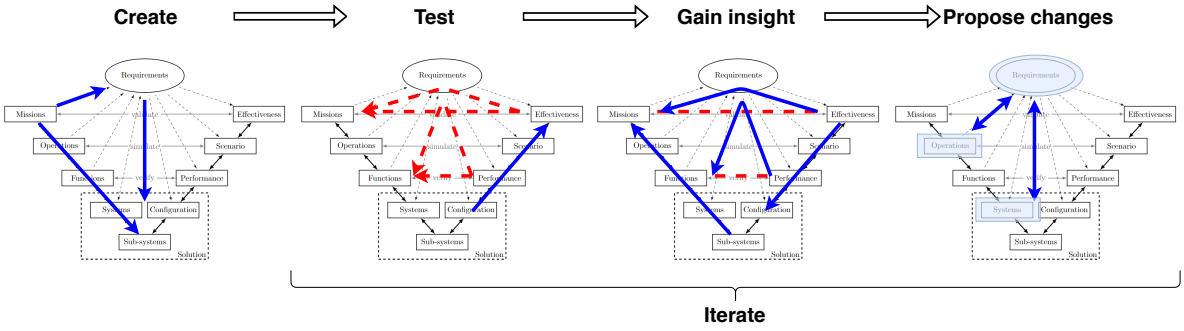


Figure 4.7: The information flow 'steps' as used in this thesis. By using one information model in one program (Shipbuilder) it is possible to perform each of these steps directly with Shipbuilder. The information model is currently distributed over several programs

To prove that this method is capable of reaching the goal, it has to be shown that it fulfils its requirements. To do this, a case study has been used. With this case study it will be shown that:

- A model can be created according to the V-model
- A trace can be created from this model
- Insight on such trace can be gained from the model

- This information and insight could help in proposing a change

To prove some of the general applicability of the method, two ships will be modelled. This will result in proving that the information model can be used for multiple vessels, as well as proving that insight from these models can be used to interpret the vessels. For this purpose the S-, and L-frigates will be used. Having two models will also help in proving if the method is capable of fulfilling the fourth item in the list above. By having the ability to determine - through insight in information - if a trace is for a specific ship it might be possible to use this model to distinguish vessels. As design exploration is part of the preliminary design process, this can be helpful to compare concepts. By seeing differences between designs more insight may be gained in the impact of assumptions, without having to use change propagation or altering of the model.

	Need	Result		Section
1	Use an information architecture fitting the DMO approach of a design cycle	Use the same information structure as the V-model used by DMO	✓	4.3
2	Quick creation in a trace from need to component and back	A program capable of creating these traces using the existing information	±/?	-
3	Store information in the model which tells how information can be interpreted	Link information in a way which can be used to interpret the information and links	±/?	-
4	Quick insight in for possible changes in the model	Use stored information and relations to identify possible impact of changes	±/?	-
5	Be reusable for different ships	Have an information model capable of describing different ships	±/?	-

Table 4.1: Requirements that may help solving the issue of gaining quick information from a ship model

4.5. Conclusion

The method in this chapter is built to improve traceability in the design process of DMO. For this purpose the information model which has been proposed resembles the V-model structure as used by DMO, connecting need, requirements and design. This could be altered to fit another design process, by using the bespoke design process outline for that process. By storing this information model in one database (in Shipbuilder) traceability should be improved. This method is the result of combining theory from Systems Engineering and Requirements Engineering and underlying theories from model-based approaches discussed in Chapter 2.

Building the information model in Shipbuilder meant some changes to the software. While most changes were possible to achieve, some were not. The changes that have not been implemented should not hinder the application of the method for the purpose of this research. However, they should be resolved if this method is to be used.

To see if the proposed method does indeed fulfil the requirements, and solve the traceability issue, a case study has been performed. In this case study two (small) models of frigates are built which are used to show how traceability works in this method. The case study does not stop at proving traceability is possible, it is also used to show why such traceability is useful.

5

Case Study

The case study in this chapter is to see how traceability works in the method proposed in the previous chapter. This method takes the first step in solving the issue of traceability being slow and tedious. How this method exactly does this, and what more can be achieved by having good traceability is described in this chapter. It will answer the sub-question: 'How does this method help in providing traceability, and what insight can be gained through this?'.

To prove the working of the method several requirements have been set at the end of Chapter 3, in Table 3.2. To show that the method can fulfil these requirements the case study should show that:

- A model can be created according to the V-model
- A trace can be created from this model
- Insight on such trace can be gained from the model
- This information and insight could help in proposing a change

In Sections 5.1 and 5.2, models based on the S-, and L-frigates, once part of the Dutch Navy, are defined and built. How information can be traced through the information models in Shipbuilder, is shown in Section 5.3. Such traces are defined to help an engineer in gaining insight in a ship design. This is done by connecting requirements to configuration, and also by understanding relations in the information model of a ship. The subject of Section 5.4 is how such insight can be gained. Section 5.5 is about 'using' such insight for understanding (and building) an information model of a ship. There is a brief study of seeing how the information model can help in proposing a change to the model, in Section 5.6.

5.1. The S-, and L-frigates

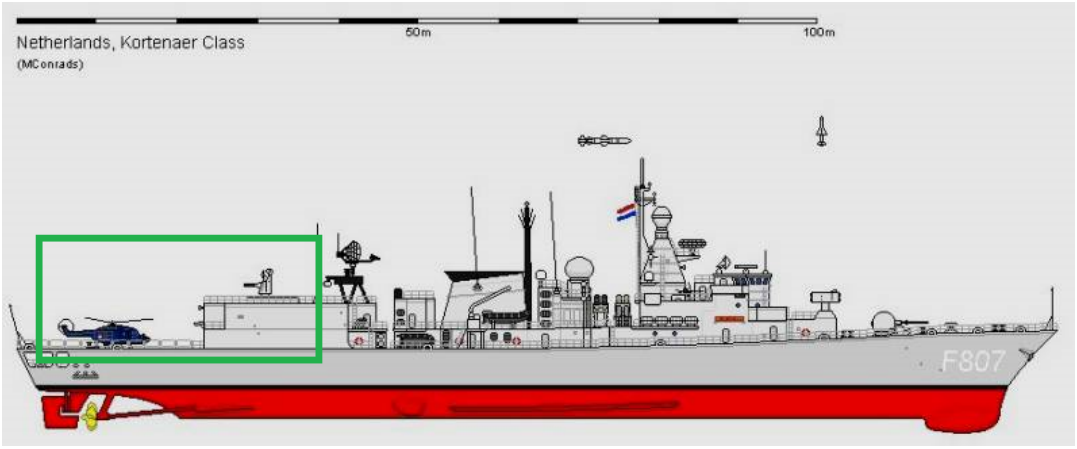
The Kortenaer-class frigates (Figure 5.1a) from the Dutch navy, also known as the standard frigate or S-frigate, were designs resulting from the replacement need at the end of the 1960's. In 1978 the first (Hr. Ms. Kortenaer) came into service, this was an anti-submarine frigate. Since 2003 all of the 10 S-frigates are out of service. The L-frigates (Figure 5.1b) operated from 1986 until 2005 for the Dutch navy. The hull is based on the Georges Leygues class from the French navy, the gas turbines are the same used in the Tromp-class (guided weapons) frigates. The German Bremen-class frigates are based on the design of the S-frigates.

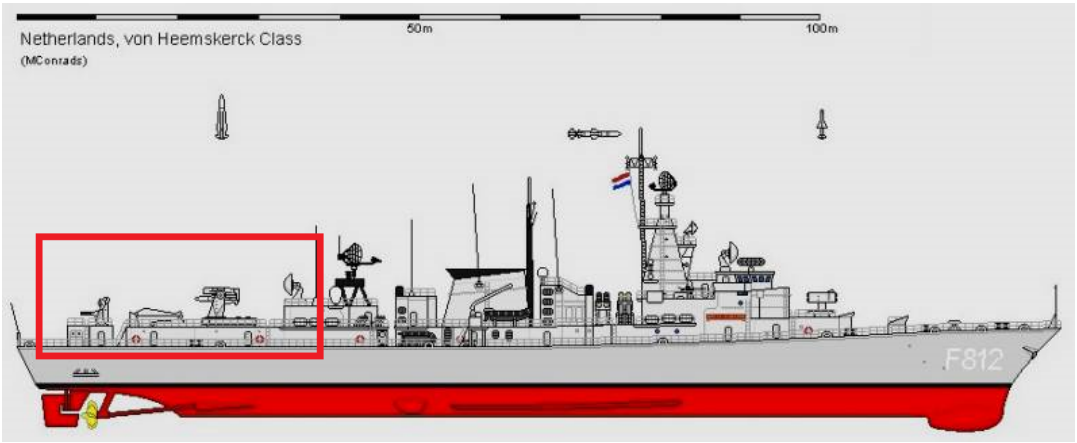
After the design was approved twelve S-frigates were ordered and a thirteenth one with altered - air defence - specifications. During the building of the 12 ordered ships two were sold to Greece. This opened the possibility to change the plans for one altered design to change two vessels. The single extra vessel now became two L-class frigates.

The design of the S- and L-frigates are considered excellent designs by the Netherlands and foreign navies alike. The success of the altered design and the fact that the design is based on the standards as set by the NATO this design is perfect for a study to show how a design can fulfil its requirements.

Where the S-frigate was designed for anti-submarine warfare with anti-air defence and anti-ship warfare capabilities; it was designed to be an escort vessel. The L-frigate focused on anti-air warfare and operating as flagship; anti-submarine and anti-ship were reduced to be used for self defence purposes only.

(a) Side view of the Hr. Ms. Kortenaer (S-frigate) as it was built[1]





(b) Side view of the Hr. Ms. Jacob van Heemskerck (L-frigate) as it was built[2]

Figure 5.1: Both the S- and L-frigates as they were built

The fact that both ships have many similarities will help to use these ships as example for preliminary designs. Both vessels are built on the same basis, but have different missions and components. Having both ships and comparing them on mission or component should then lead to being able to distinguish the difference in the vessels. This can be used to simulate the uncertainties and different design solutions as can be encountered in the preliminary design phase.

For the early models to be built both the aft parts of the ship will be taken as point of interest. The aft ship is where the biggest (visual) differences between the S-, and L-frigates are. For the S-frigate the aft localises the helicopter (green square in Figure 5.1a), while the L-frigate localises a guided missile launcher (red square in Figure 5.1b). To also include a similar system for both vessels a part of the electrical system of the vessels will be used. Later in the test cases, more systems and components will be added. Initially these three different systems are used to model both ships.

[1]Source: `http://www.shipbucket.com/drawings/3375` (06-2019)
[2]Source: `http://www.shipbucket.com/drawings/3380` (06-2019)

5.2. The model of a ship

The first step in building a model in Shipbuilder is to divide the parts of interest in the ships according to the System Engineering approach. This goes according to one breakdown using the items as found in the V-model: mission, operation, function, system and component (sub-system). The resulting breakdown resulted in items as shown in Figure 5.2.

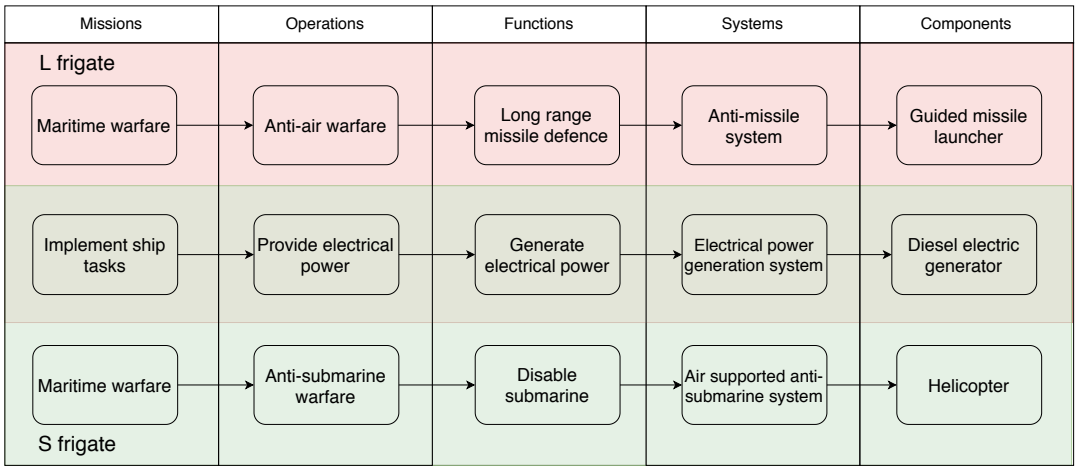


Figure 5.2: Breakdowns of some systems on board of the S-, and L-frigates, using an SE approach

The breakdown as described above, is then put into the information model in Shipbuilder. This means that a ship is built from objects in the Knowledge Base. By adding such objects to a ship, it is possible to adapt said object for a specific ship. For the model of the S-frigate, Figure 5.3 shows how the information model looks like in Shipbuilder. Note, that as described in Section 4.2.1 the function is not part of the ship. For the purpose of connecting and storing information, and finding if the method suits its requirements, this is acceptable.

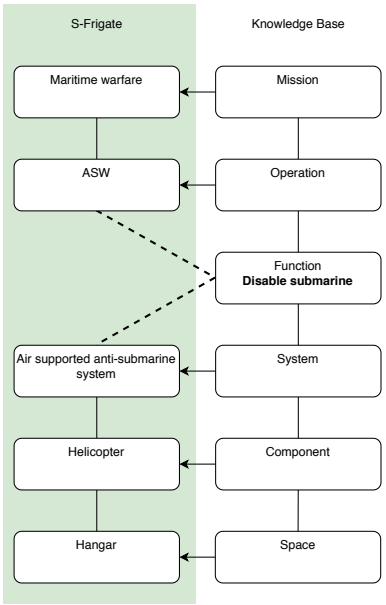


Figure 5.3: The model in Shipbuilder, using the V-model as structure, of the S-frigate specific breakdown in Figure 5.2

5.3. Traceability - in the model of a ship

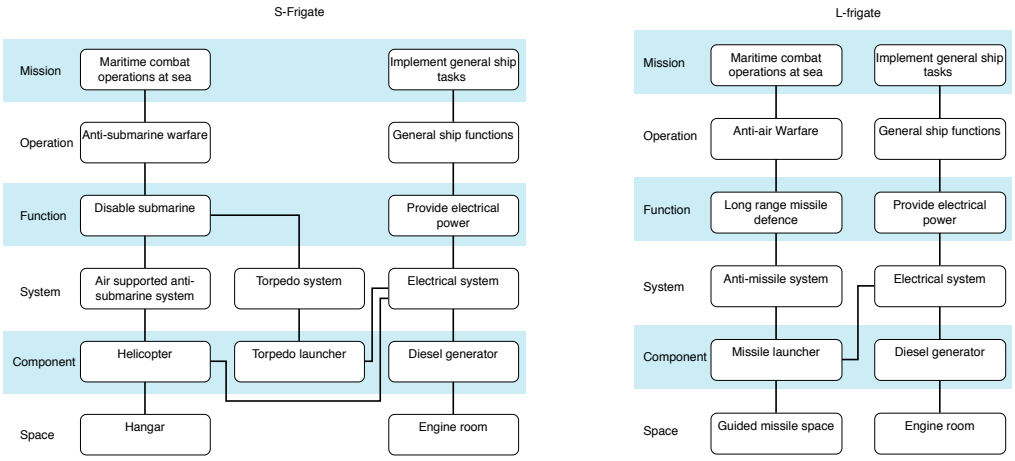
One of the goals of the information model of the method is to (quickly) create a trace in the model. Such trace has to be able to be built from either mission to component (or even

spaces) or created the other way around (component to mission). The best approach for creating such a trace is to use the 'layers' of the information model. For this the structure of the V-model has to be used. Creating the trace is about following connections to another 'layer'. Following connections in one 'layer' does also create traces and may lead to insight, but it does not give information concerning the object in the V-model structure.

So, in order to get the trace which is currently difficult to obtain is the vertical traceability [31, p. 109]. The lateral and longitudinal traceability, whilst giving insight in the background information, does not provide traceability in accordance with the structure as seen in the V-model. The traces can therefore be said to be either top-down (mission to component) or bottom-up (component to mission), with respect to the information model.

To give an example of a trace which can be created with this traceability approach, a trace in the S-frigate has been created. While the connections leading to the creations of such traces are in the software, there is currently no automated way to create the traces. The created figure which shows this trace is therefore nothing more than a representation of this trace.

Following the connections as presented in Shipbuilder led to traces as seen in Figure 5.4. For all traces there is no direction given for the trace, it is not stated if this trace is top-down or bottom-up. This is because there is no real distinction for this in the information model. There either is a tracing possibility or there is not, and if the possibility exists it goes both ways. So it is possible to go to a space when starting at a mission, and at the same time it is possible to go to a mission starting from a space. Note that the problem with the current implementation of functions in the software means that an engineer has to have knowledge of the model in order to create such traces or requirements have to be used to create the traces. The second option is somewhat of a proof that the design and requirements are indeed (good) connected in the software.



(a) Traces in the model of the S-frigate (b) Traces in the model of the L-frigate

Figure 5.4: Representations of traces in the data of the S-, and L-frigate

5.4. Insight - connecting requirements to design

Now that it is possible to create traces in the model of a ship, representing the complex system, the next step is to see if it is possible to get insight in such traces. This is where the connection between design and requirements comes into play. In the figure showing the traces in the S-frigate (Figure 5.4a) there are two branches below the function "disable submarine". One branch leads to a torpedo system, the other to a helicopter. This is because both systems can be used to perform this function, so both systems can be present in a ship. But the trace does not tell why one (or both) of them is in this trace, it just shows that it is in the model.

The reason that both systems are in the model of the S-frigate, is that the S-frigate has two requirements connected to the ship:

1 Perform offensive ASW operations
2 Perform defensive ASW operations

These two requirements come from the ASW operation which has to be performed by the ship. Having these two requirements is the reason why the traces split below the function. The proposed design solution for the first requirement is to use an on-board torpedo system, while the second requirement has been fulfilled using a helicopter with an air-lifted torpedo system.

To actually see the requirements on the traces, Figure 5.5 has been created. In this figure it can be seen that the first requirement is connected to the air-lifted torpedo system, and the second requirement is connected to the torpedo system. With this information, these requirements applied to the ship, there is now a reason why both of the systems are in the model of the ship.

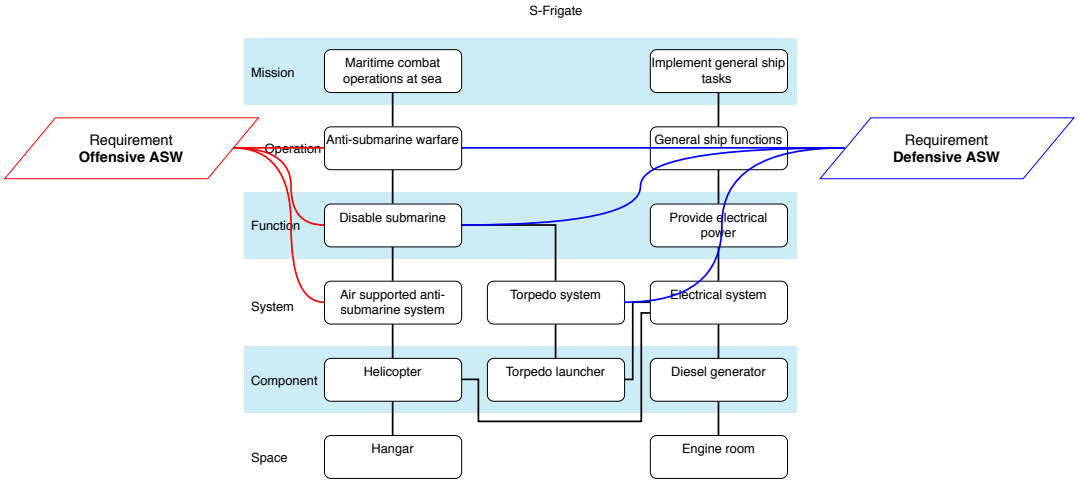


Figure 5.5: Requirements projected onto the trace from Figure 5.4, to show some context

In the figure only showing the trace (Figure 5.4a), the function is still ambiguous. It leads to two different systems, and the figure holds no information to why this is. However, in the model of the ship, this information is stored. This information is the presence of requirements and connections of these requirements to the shown objects. These connections to an extra piece of information then counteract the ambiguity which could arise from the figure. Note then, that this figure is nothing more than a possible representation of the ship.

The use of connecting information can be described with the following statement: Having the option to connect requirements (and other objects) to one another it is less important for a requirement (or object) to be unambiguous. Ambiguity is created by a difference in interpretation, which relies on context. By capturing the context in the relations to a requirement (object), ambiguity can be counteracted.

5.5. Using insight

With the ability to create insight in the model, by capturing some context, it might be possible to built a system with seeming ambiguity. Such ambiguity can be using one object for multiple solutions. Only seeing the object then does not give an answer as to why it is there; it is ambiguous. Knowing the context for this object will counteract this ambiguity, and exactly this can be done by connecting other pieces of information to this object.

In this section two possible ambiguous objects are analysed: a function and a system. For both objects two implementations are determined, which will be supported by extra information on these objects.

5.5.1. For an ambiguous function

To show how a system can be ambiguous, and information can counteract this, the function as discussed in section 5.4 is taken as example. How the function ”disable submarine” can

be interpreted in two ways, and thus 'solved' in two ways is shown in this section. In Figure 5.6 the breakdown for the function leading to two systems and corresponding components is shown. This breakdown is based on the same underlying information as the traces earlier: there are two requirements each leading to one of the branches.

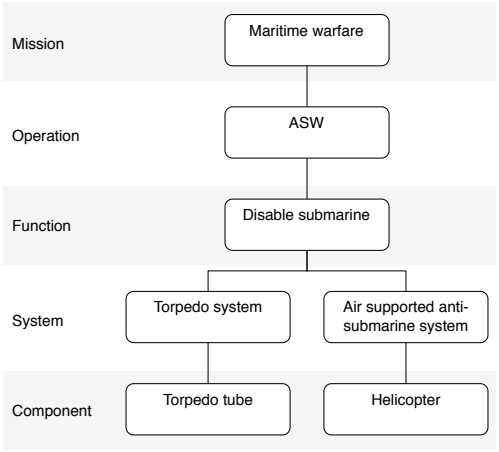
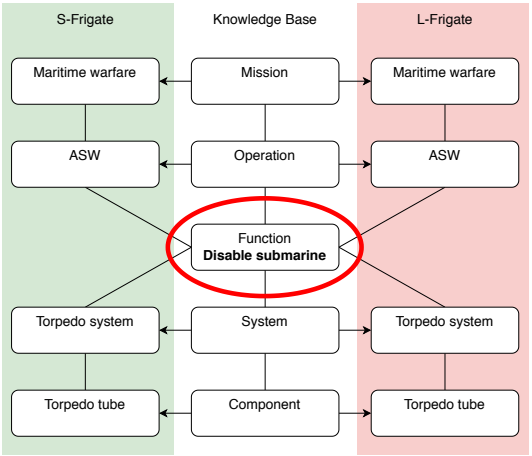


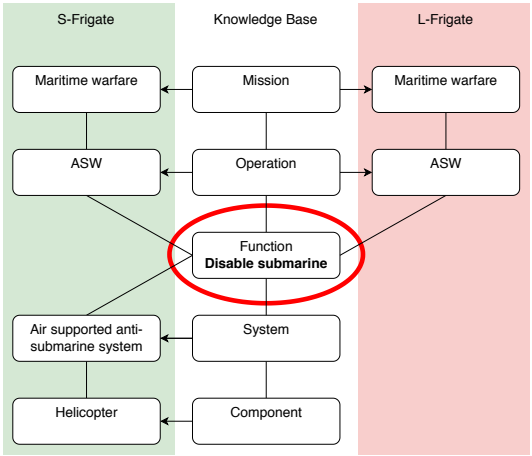
Figure 5.6: A breakdown with one function leading to two systems

In the S-frigate the application of the function with the offensive and defensive requirements is now known. To show that not having one of the requirements does indeed lead to another model of a ship, the L-frigate is also observed. The L-frigate, being an AAW frigate, does hot have the offensive ASW requirement placed in the model of the ship. The defensive requirement is, however, placed in the model. This is because of the fact that this requirement does not come from either the ASW or AAW missions or operations. The defensive requirement is the result of the frigate needing to have a general defensive capability. Since both the S-, and L-frigate do need this defensive capability the requirement remains.

To show how the model of a ship is affected by the implementation of a requirement, the Figures 5.7a and 5.7b were created. In Figure 5.7a the defensive requirement is taken as the context and in Figure 5.7b the offensive requirement is taken as context. What can be seen is that both models of the vessels do have the torpedo system incorporated, while the helicopter is only present in the S-frigate model. However, the function (and higher level objects) is present in both models.



(a) The implementation of the function for the defensive ASW requirement



(b) The implementation of the function for the offensive ASW requirement

Figure 5.7: A visualisation of both the SE models of the S- and L- frigates where both figures include the same function (disable submarine) connected to the vessels but with different implementation

5.5.2. For an ambiguous system

Similar to ambiguous functions, it is also possible to use systems for different purposes. Again, it is the context, in the form of requirements, that helps resolve the ambiguity. This example, however, is somewhat more complicated than the example with the function. This is because the observed system does not only lead to different branches, but is also at the end of two different other branches. This can be seen in Figure 5.8.

Again the distinction in the breakdown is the result of two requirements: offensive AAW and defensive AAW. For both requirements, as with the ASW requirements, it is the case that the defensive requirement is related to survive-ability of the ship.

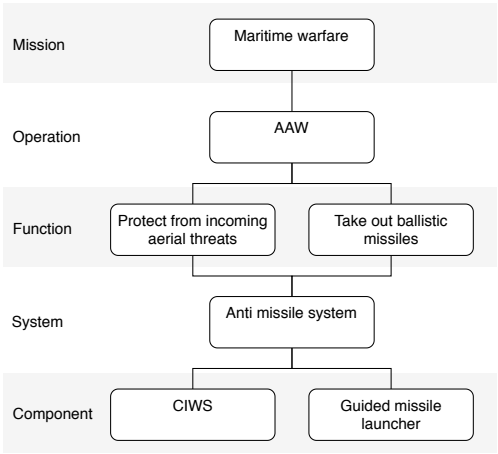
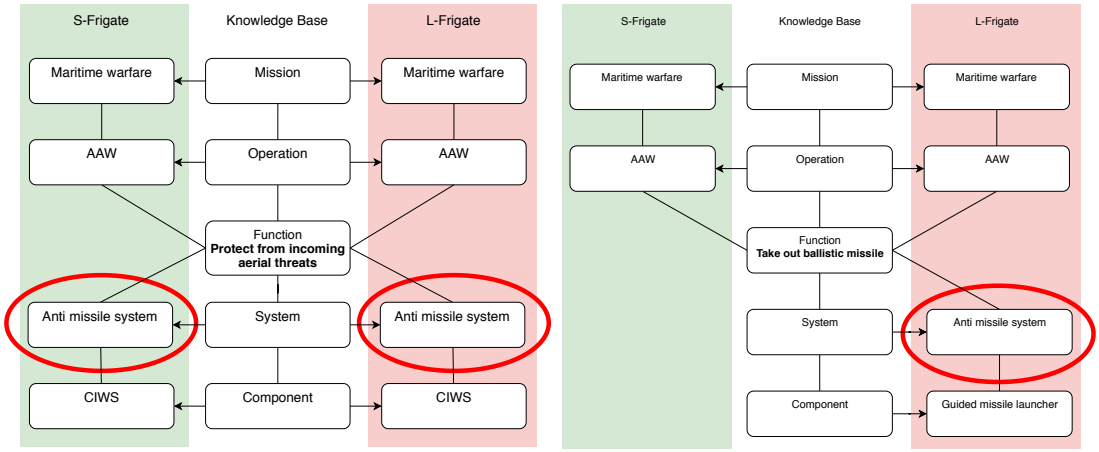


Figure 5.8: A breakdown with one system leading to two functions and thus components

As with the ambiguous function, the visualisation of the ambiguous system is based on two different requirements: offensive and defensive. The resulting figures are Figure 5.9a, for the defensive requirement, and Figure 5.9b, for the offensive requirement. Both the models of the S-, and L-frigate are shown.



(a) The implementation of the function for the defensive AAW requirement

(b) The implementation of the system for the offensive AAW requirement

Figure 5.9: A visualisation of both the SE models of the S- and L- frigates where both figures include the same system (missile launch system) connected to the vessels but with different implementation

5.6. Help in proposing a change

The impact of a change to the model is still up to an engineer to determine. It is, however, possible to create traces in the model to determine where a choice has been made. It is also

possible to gain some insight in this choice as the requirements create context and supporting information can be linked to objects in the Shipbuilder software. This information can help an engineer in proposing a change to the design, in the model.

For seeing how this would work, a small example is used. This example is based on one of the breakdown branches in Section 5.5.1: the torpedo tube (see Figure 5.6). This branch comprises of: Maritime Warfare, ASW, Disable Submarine, Torpedo System and Torpedo tube. Each of these objects has requirements, and the component (torpedo tube) is placed in a space in the configuration of a ship. When testing a ship on fulfilment it can happen that one of the requirements connected to this branch is not satisfied. For this example, this is that the torpedo tube cannot deliver the required force to fire a torpedo.

In the model in Shipbuilder, an engineer is able to see where the torpedo tube is connected to. At the 'other end' of the connection is also an object, containing information and connections. So is the torpedo tube part of the torpedo system and the electrical system (see Figure 5.4a). It is then brought to attention of the engineer, that changing the torpedo tube does not only impact the torpedo system, but also has an impact on the electrical system. Proposing a change to the torpedo tube to increase its power, will also influence the electrical system of the ship (for the sake of this example the pneumatic system used by the torpedo tube is also part of the electrical system). If the electrical system has been optimised for a specific load, it could be possible that applying a change to this system is difficult or expensive.

The example given is short, and somewhat obvious. However, having the possibility to actually see how these objects interact is important. Such interactions can also be those that are not obvious. In those cases, having the possibility to trace this information can be helpful for engineers to make an assessment of the impact of a proposed change. To see how this can really benefit engineers, more research is required. The models created for this thesis are too small: making the interactions 'too obvious' and not very complicated.

5.7. Checking requirements

In Chapter 4, a method was proposed to see if traceability can be increased. For this to be concluded some requirements were set up. In this chapter, the method has been tested to see how it fulfils the requirements. The results can be found in Table 5.1.

	Need	Result		Section
1	Use an information architecture fitting the DMO approach of a design cycle	Use the same information structure as the V-model used by DMO	✓	4.3
2	Quick creation in a trace from need to component and back	A program capable of creating these traces using the existing information	✓	5.3
3	Store information in the model which tells how information can be interpreted	Link information in a way which can be used to interpret the information and links	✓	5.4
4	Quick insight in for possible changes in the model	Use stored information and relations to identify possible impact of changes	✓	5.6
5	Be reusable for different ships	Have an information model capable of describing different ships	✓	5.2

Table 5.1: Requirements that may help solving the issue of gaining quick information from a ship model

Performing the case study resulted in seeing how the requirements are fulfilled. With this it can be said that the proposed method can indeed help in solving the issue of tracing information in a ship design process. However, the case study has now only been used to prove that the method could work. To see how good the method works and if it can handle large, complex, models more research has to be done. This can also be in the form of using this method during a (small) project.

With the current test case, it has not been tested how much time can be gained in the insight phase. Neither has been proven that gaining insight works with real complex models. While it has been shown that it is possible to trace information, gain insight with this and even possible determine impact of change, this is only the first step in applying the method.

Nevertheless, has the case study provided positive results. By proving, with a small example, that this method does work the first step in proving the use of this method has been taken.

5.8. Conclusion

The case study in this chapter shows how traceability works with the proposed method. What can be done with these traces and traced information has also been shown. Combining the theory from Systems Engineering, Requirements Engineering, Model based approaches and current practice of the DMO design approach led to the method and the here discussed case study. By seeing how the method fulfils the set requirements with this case study, it can be said that this method does look like a solution to capturing and using the relations between need, requirements and design. Seeing an example of how this could support the design effort, in determining changes, shows that this method does support traceability in order to improve the design effort.

6

Conclusions

This thesis started with the issue at DMO of having difficulty in tracing information - quickly - during a design project. As DMO operates mostly in the preliminary design phase, where requirements are elucidated and configurations are used to support this, the focus of tracing of information was between requirements and design [44]. This led to the following question to be answered in this thesis:

> How can we capture the relations between need, requirements and design to improve the traceability of information on a naval vessel in the preliminary design phase?

To find an answer to this question, four separate sub-questions have been answered. With the help of these questions the author was able to understand the context of the problem and work towards a possible solution for the issue. These questions are:

1. How can Systems Engineering (and Requirement Engineering) help to capture the relations?
2. How are the relations currently managed during the design process at DMO?
3. What could be a method to be able to capture, manage and use these relations?
4. How does this method help in capturing these relations, and what knowledge can be gained through this?

Each of the sub-questions have been answered in a chapter, of which a summary of the findings are stated below.

How can Systems Engineering (and Requirement Engineering) help to capture the relations?

Systems Engineering and the incorporated field Requirements Engineering, are both research fields for creating and managing information and configurations of a design process. From these two fields it is possible to see how an information model could help the design process; after all a generic information model for a design is what these two research fields are about. One of the used process models is the V-model. This model covers the relations between need, requirements and configuration. In a V-model, Requirements Engineering can be seen as having focus on the 'top-half' while the configuration design is located at the 'bottom-half', see Figure 6.1.

With the help of modelling programs and computers, it is possible to create large, complex, information models. However, these 'models' are generic and do not have one specific application which can be used, resulting in difficulty in finding research for applied models with lessons learned. To use such a model based approach, an applied model for DMO has to be created. While building such model takes a long time, it could save time in an iterative process. In an iterative process it is useful to have one information model capable of growing with it, instead of having to redefine information for a ship in each iteration or design phase. For more information, see Chapter 2.

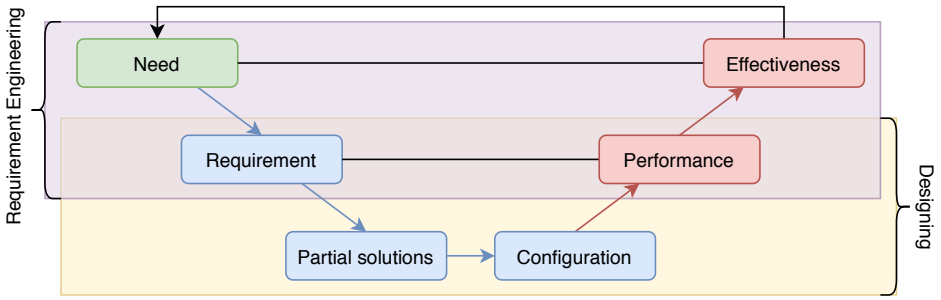


Figure 6.1: An interpretation of a Systems Engineering V with highlighted sections focusing on Requirements Engineering and 'design of configuration'. The colours of the rectangles should be interpreted as follows: green is the 'given' description leading to a solution, the blue is the 'creation' of this solution and red is the 'testing' of the solution.

How are the relations currently managed during the design process at DMO?

At DMO the Systems Engineering approach is used to define the information structure and process architecture in their design process, in the form of a V-model. Their information model is spread over several programs (Figure 6.2), which are used in different design stages and by different stakeholders. This can make it difficult to find information and relations in the complexity of programs, stakeholders and uncertainties. This does not result in unsuited designs, but it hinders the search of information when looking for insight in a design. To improve the information storage, the Shipbuilder application is integrated in the DMO design process. This program can be used to capture and manage large quantities of information, ranging from need to configuration. For more information, see Chapter 3.

(a) The parts of the design process supported by DOORS

(b) The parts of the configuration design process supported by FIDES, PACKING, QUAESTOR and Rhinoceros

Figure 6.2: The focus of the current most used programs in the design process. The programs are separated by focus on Requirements Engineering and 'design of configuration'. The blue colour is on 'creating' a solution and red is the 'testing' of a solution.

What could be a method to be able to capture, manage and use these relations?

One method to capture and manage the information and relations for a design process is using the V-model as used by DMO as the architecture for an information model, which can be seen in Figure 6.3. As the V-model, at DMO, is used to define the design steps in a ship design process, but also to define objects for decomposition of a ship, this model is perfect to use as base for an information model. As Shipbuilder is being implemented at DMO, and this program is based on Systems Engineering, and information (objects) can be defined in Shipbuilder, this program is a useful base to create an information model as proposed.

To prove the working of this proposed method the following steps have been defined:

1. 'Visualise' the information model in Shipbuilder, according to the V-model
2. Change the information structure to represent the V-model
3. Create a model of a ship with the improved database model
4. Show traceability between information of requirements and configuration

The information model in Figure 6.3 is the result of the first two steps. With one information model in the Shipbuilder software covering requirements and configuration, the two last steps are performed with a case study. For more information, see Chapter 4.

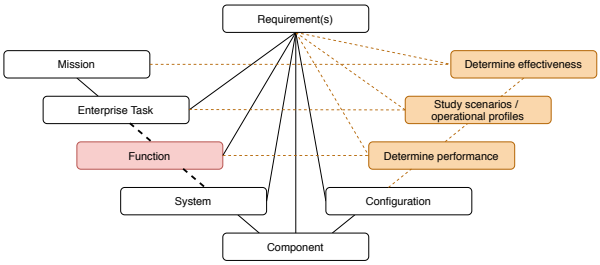


Figure 6.3: The information model proposed in this thesis. This model is able to capture information of a ship ranging from need to configuration while being applied in one program (Shipbuilder).

How does this method help in capturing these relations, and what knowledge can be gained through this?

To see how the proposed method actually works, a case study has been used. With this case study, (small) models have been built which were then used to see how traces can be created from these by showing the following steps:

- A model can be created according to the V-model
- A trace can be created from this model
- Insight on such trace can be gained from the model
- This information and insight could help in proposing a change

By showing the results for each step, the fulfilment of the requirements for the method (stated in Chapter 3) can be checked. In Table 6.1 the fulfilment of each requirement can be found, along with a reference to the section where the solution can be found.

	Need	Result		Section
1	Use an information architecture fitting the DMO approach of a design cycle	Use the same information structure as the V-model used by DMO	✓	4.3
2	Quick creation in a trace from need to component and back	A program capable of creating these traces using the existing information	✓	5.3
3	Store information in the model which tells how information can be interpreted	Link information in a way which can be used to interpret the information and links	✓	5.4
4	Quick insight in for possible changes in the model	Use stored information and relations to identify possible impact of changes	✓	5.6
5	Be reusable for different ships	Have an information model capable of describing different ships	✓	5.2

Table 6.1: Requirements that may help solving the issue of gaining quick information from a ship model

By having an application capable of capturing information (both for requirements and configuration) and connections between these pieces of information it is possible to create traces. The traces built for the case study are focused on the 'configuration' of a ship, where requirements create context. Having the requirements connected to the traces can give insight for an engineer. Not only is it possible to connect requirements to a configuration in a design and trace information between them, it is also possible to gain insight with these traces.

With this the proposed method does seem to manage the relations between requirements and configuration, and knowledge can be gained from this. In the steps of 'creating', 'testing', 'gaining insight on' and 'proposing changes to' a model the proposed information model can be used as seen in Figure 6.4. For more information, see Chapter 5.

How can we capture the relations between need, requirements and design to improve the traceability of information on a naval vessel in the preliminary design phase?

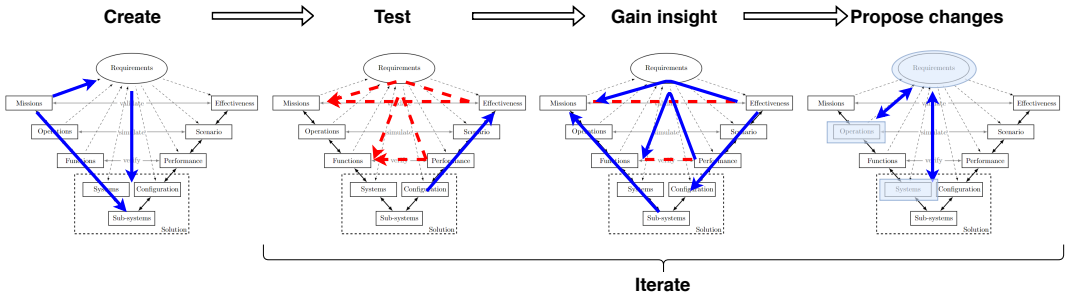


Figure 6.4: The steps that can be performed with the model. By using the method with these steps it should be easier to gain insight and propose changes during a ship design process.

With these answers it is possible to state an answer for the main question in this thesis. The relations between requirements and configuration can be captured in one information model (Figure 6.3) - in one program - and thereby the traceability of information is improved. This information model is based on Systems Engineering and Requirements Engineering. With this model it is possible to define a 'ship' and store information on this ship accordingly. The information model, can be used to create, test and gain insight and can help in proposing changes in a design (see the steps in Figure 6.4). As the model is based on the V-model used by DMO, it covers requirements and design. The need is also covered, as the need is captured in requirements. This then creates a connection between need, requirements and design. Having one information model to cover all these in one program, it is possible to trace information through a design (captured with an information model).

6.1. Discussion and limitations

Where the use of one information model in one database has been shown, there are several limitations and implications with this method. One of which is that in this thesis the structure of an information model has been proposed. Models of ships build with this structure proved to be able to support traceability.

Using one ship model might also help when going through different phases a design process in general. Using this ship model as a base for the next design phase could improve traceability throughout the design process. By adding more detail to the ship model corresponding to each phase, the model should better describe a design fitting the phase. To be able to do this, the ship model should be tested to work if it is far more complex (i.e. contain more interactions). The information structure to build the ship model should not increase in complexity. Next to this added complexity, the approach (and information model) also has to be tested to see if it holds when a design project proceeds to a next phase.

Even with more complex ship models describing the wanted level of detail, there is still room for improvement. What is not really mentioned in this thesis is that the design of a naval vessel is not really about one vessel, it is about designing a vessel class. This is partially covered in the Shipbuilder software by using a separate project for each class. While this does divide stored, and related, information per class, it does not provide means of using 'partial solutions' for the all concepts for this class. In a project it is possible to built ships, but these do not have to be related to each other; they are all built separately on the Knowledge Base. Here it might be beneficial to create an extra 'information layer': a ship class. This layer is placed between the KB and a ship. With this it might be possible to use (design) solutions for each concept (ship) in the class, while keeping the knowledge in the KB as generic as possible. Such approach might help to separate general knowledge (KB), applied knowledge for a class (new 'information layer') and applied knowledge for each concept (ship).

Besides the added complexity to this model in the approach, there are other aspects with are of interest as well. One of this is to find an answer to the question of how much faster tracing information with this approach actually is. It has been shown that tracing of information does work in the model, but no time restrictions have been used. Also were the models for the case study built purposely to show traceability, not to show that the method

supports development of a design. Tracing information in a ship design process is currently a tedious job, but it is not impossible. The proposed method seems to be able to quickly trace information. However, it has not been proven that it is actually faster than in the current approach.

Using one model seems like a good step in the direction of creating better traceability through the ship design process. However, there is one problem with an approach like this. That is that for defence purposes it is often practice to divide information in order to prevent leaking of information and access to 'unnecessary' information. In the Shipbuilder software there are possibilities to give each individual access, or deny access, to pieces of information. While this prevents the access to 'unwanted' information, the traceability through the information should still be based on the connections to these 'unwanted' pieces of information. In the end, all information is still related to each other. This might pose difficulty in maintaining the 'kneed-to-know' basis of information, or at least this has to be implemented from the start when implementing such model based approach.

6.2. Recommendations

To get a solution to the traceability and change propagation issue in balancing capability, feasibility and cost, some more research has to be done. Some aspects have been mentioned throughout the report and some have been mentioned above in the discussion. The proposed method seems to be a step forward in working towards a solution for the current issue. Traceability between need, requirements and design will be easier with the proposed method. This should help engineers to balance capability, feasibility and cost.

The information model and ship models built for this thesis did help in proving the initial working of the method. To be able to further prove the working of the method during a design process and understand the impact of having connections between relations, some aspects are to be researched.

Create visualisations for the model to show traceability and 'insight'

While the model is capable to store and manage connections, there is no (quick) adaptable visualisation possible of traces in the model. To provide engineers with the insight of seeing these connections Design Structure Matrices could help. However, these matrices could prove to be too large to be helpful in a large complex model. Therefore, it is useful to research what kind of information stakeholders need from an information model of a ship. When this is known, then the step of visualising this information can be taken. Having only a model capable of connecting a complex model could still be beneficial, but if insight can quickly be gained from it by using visualisations it is even more useful.

Build bigger models to see if the structure works

The current models of the S-, and L-frigates are small. They can be small, because they were only used to prove the basics of the method. To see if this method really works in a design process, larger (and more complex) models have to be created using this method. Shipbuilder can work with large and complex models, as this is their business. The next step is to use the information model as proposed to build and manage such models.

Research if the method can move along with consecutive phases

For this thesis only the preliminary design phase has been used to define the level of detail for a ship. However, the design process covers several phases. To see if the proposed method can also be used in different phases has to be researched. Along with that research, it is also useful to determine if the model is able to 'grow' along with the phases. If the method works with each phase, but cannot evolve with consecutive phases, a new model has to be created in each phase. This would require time in each step, but could also help in removing unnecessary information from the model. As said, this requires further research.

See if the method does speed up the process of traceability

As said in the discussion, the current case study only proves that it is possible to create

traces through the information. There is no mention of time in this test case. For this method to be truly useful for engineers, it has to be faster in giving engineers insight in the information in a ship.

While it might be possible with this method to see 'new' relations in a ship, this is not the main goal of this method. This is to speed up gaining insight in information and changes in a ship.

Gain more knowledge on the 'captured context'

During the course of this research it has been found that it is possible to use ambiguous information to clearly define information. This can be done by relating pieces of information together, thus giving 'ambiguous' information context.

This context has slightly been used in the test case in Sections 5.4 and 5.5. Here it shows that it might be possible to use related information in the model to define context. How, and if, this can be used to provide context in a large model could be interesting to research. When this does work it might mean that stored information can be better reused in other projects, as this information does not have to be defined project specific.

See how the method can determine impact of change on a model

In Section 5.6 it is briefly mentioned how the method could help in determining changes, and impact of these changes, to a model. However, it still relies heavily on engineers with knowledge to be able to help here. Determining impact of change in the model with the software itself will require implementation of change propagation software, whereas now this comes form the engineers. It might prove to be difficult to have the application itself being able to determine the impact of change, as change propagation software is still being developed and researched. Even with the application being able to follow changes, it starts with researching how the current traces can help stakeholders in determining changes and change impact.

Build a requirement architecture in Shipbuilder

In the case study the focus was not really on the 'requirements side' of a ship. Also, the proposed information model contains no information for a requirement structure. DMO is currently working in Shipbuilder to define requirements, lessons learned here can be used to provide a requirement architecture. Such an architecture then has to be connected to the proposed information model (and configuration model) in Shipbuilder.

Create an extra information 'layer' for ship classes

As mentioned in the discussion, it could be helpful for the design process (at DMO) to have the information model work on a ship class as well. Currently there is the Knowledge Base and ship 'layer' in the information structure of Shipbuilder. Most design projects for naval vessels, however, focus on the design of a class and not only on one design. Each design should fall within the needed class, they are not completely separate ships. This is currently not implemented in the software or the method. How the method could include the use of classes between the stored knowledge and applied knowledge at ships is an interesting subject for further research. This is more towards an IT subject, as it is mostly concerned with the structure of the uses program (Shipbuilder in this thesis).

Besides further research on this method, improvements can be made on the actual implementation of the method into the workflow of DMO. The current model and implementation of it in Shipbuilder are currently only set out to prove the basic workings of the method. For the method to be used by DMO and Shipbuilder, some changes are in order. One of which is the implementation of the 'functions' object, as mentioned in Section 4.2.1. Besides changes such as this, there might be some changes to terminology, definitions or implementations.

A last note to add here is the addition of the 'cost' of a concept. When testing a ship on performance, the building cost of such ship can be determined. In the effectiveness of a ship the affordability of said ship can be determined as this is a combination of cost and other aspects. Being able to trace information though the model should also imply that cost can be

traced. This will help in determining the cost in the first place, and elaborate traceability can help determining the affordability. Proposing changes based on cost should also be possible with an information model as used in this thesis. However, this has not been implemented or tested in the current model and could prove to be a great addition.

6.3. Personal reflection

At the beginning of this thesis there was an idea to improve the feedback (of information) in the design cycle (see Figure 1.2). While searching for information on what such feedback could be, and what the difficulty of defining a ship is, the 'separation' between requirements and configuration caught my attention. Therefore, my literature research has been on model-based approaches (Design matrices and Axiomatic design), Requirements engineering and Systems Engineering.

In the early stages of the research I set out to build a model capable of providing 'traceability' in ship design. This meant this model had to incorporate Systems Engineering, Requirements Engineering and model-based approaches. During the course of the research I kept digging in theories on these subjects to support decisions I wanted to make for a model. It proved difficult to create a satisfactory new model capable of doing this, and it turned out that did was not needed. This resulted in the use of the V-model as used by Van Oers [43] and Duchateau [15] to function as the architecture of the information model.

Only at a late stage in the research it became clear to me that I had done enough to say something for the initial problem. At this point in time I started working on writing a report to capture my research. The report that followed was created in too little time and with too little cooperation. This resulted in my research being delayed.

The delay in my research meant that I had time to really think about the story I wanted to tell with my research. I also had to do this because this was not clear from the initial report. In this phase of the research I had a lot of contact with my supervisors and others to talk about my research. From this I noticed that my story became clearer and better each time I sparred with someone about it.

In the end I think the research has been beneficial for me and for others. Besides learning what research really encompasses, I also learned more about working 'alone' on a project with others. While there were some less pleasant times for me during my research, the overall feeling that I hold now is quite positive. In the end I think I have done some interesting work and gained valuable insights in ship design and the corresponding process.

Bibliography

[1] D.J. Andrews. Marine design - requirements elucidation rather then requirements engineering. Proc. 8th Int. Marine Des. Conf. (IMDC), 2003.

[2] D.J. Andrews. Marine requirements elucidation and the nature of preliminary ship design. International Journal of Maritime Engineering, 153(1):23 – 39, Jan. 2011.

[3] A.B. Bahtia. Database Management System. Alpha Science International, 2014. ISBN 9781783321919.

[4] J.E. Bartolomei, M. Cokus, J. Dahlgren, R. de Neufville, D. Maldonado, and J. Wilds. Analysis and applications of design structure matrix, domain mapping matrix, and engineering system matrix frameworks. Technical report, Massachusetts Institute of Technology, Cambridge, MA, 2007.

[5] J.I. Bernstein. Design methods in the aerospace industry: looking for evidence of set-based practices. Master's thesis, Massachusetts Institute of Technology, 1998.

[6] E. Bonjour, S. Deniaud, and Jean-Pierre Micaëlli. A method for jointly drawing up the functional and design architectures of complex systems during the preliminary system-definition phase. Journal of Engineering Design, 24(4):305–319, 2013. doi: 10.1080/09544828.2012.737457. URL https://doi.org/10.1080/09544828.2012.737457.

[7] T. R. Browning. Design structure matrix extensions and innovations: A survey and new opportunities. IEEE Transactions on Engineering Management, 63(1):27–52, Feb 2016. ISSN 0018-9391. doi: 10.1109/TEM.2015.2491283.

[8] D.M. Buede and W.D. Miller. The Engineering Design of Systems : Models and Methods. John Wiley & Sons, 2016. URL https://ebookcentral-proquest-com.tudelft.idm.oclc.org/lib/delft/detail.action?docID=4391537.

[9] P.J. Clarkson, C. Simons, and C. Eckert. Predicting Change Propagation in Complex Design . Journal of Mechanical Design, 126(5):788–797, 10 2004. ISSN 1050-0472. doi: 10.1115/1.1765117. URL https://doi.org/10.1115/1.1765117.

[10] R. Coyne. Wicked problems revisited. Design Studies, 26(1):5 – 17, 2005. ISSN 0142-694X. doi: https://doi.org/10.1016/j.destud.2004.06.005. URL http://www.sciencedirect.com/science/article/pii/S0142694X04000626.

[11] M. Danilovic and T.R. Browning. Managing complex product development projects with design structure matrices and domain mapping matrices. International Journal of Management, 25:300 – 314, 2007.

[12] T.W. DeNucci. Capturing Design: Improving conceptual ship design through the capture of design rationale. PhD thesis, TU Delft, 2012.

[13] Det Norske Veritas (DNV). Rules for classification of ships - nautical safety, pt.6 ch.8 sec.2, July 2011.

[14] J. Dick. Design traceability. IEEE Software, 22(6):14–16, Nov 2005. ISSN 0740-7459. doi: 10.1109/MS.2005.150.

[15] E. Duchateau. Interactive evolutionary concept exploration in preliminary ship design. PhD thesis, TU Delft, 2016.

[16] A. Egyed and P. Grunbacher. Identifying requirements conflicts and cooperation: how quality attributes and automated traceability can help. IEEE Software, 21(6):50–58, Nov 2004. ISSN 0740-7459. doi: 10.1109/MS.2004.40.

[17] A.M. Farid and N.P. Suh. Axiomatic Design in Large Systems: Complex Products, Buildings and Manufacturing Systems. Springer International Publishing, Cham, 2016. ISBN 978-3-319-32388-6. doi: 10.1007/978-3-319-32388-6. URL https://doi.org/10.1007/978-3-319-32388-6.

[18] M. Van Loenen G. Schouten. Ict strategie b.v. - shipbuilder software, 2019.

[19] M. Giffin, O. de Weck, G. Bounova, R. Keller, C. Eckert, and P.J. Clarkson. Change propagation analysis in complex technical systems. Journal of Mechanical Design, Jul 2009. URL https://doi-org.tudelft.idm.oclc.org/10.1115/1.3149847.

[20] O. Gotel and S. Morris. Requirements tracery. IEEE Software, 28(5):92–94, Sep. 2011. ISSN 0740-7459. doi: 10.1109/MS.2011.105.

[21] INCOSE and Wiley. INCOSE Systems Engineering Handbook: A guide for Systems Life Cycle Processes and Activities. Wiley, 2015. ISBN 9781119015123. URL http://ebookcentral.proquest.com/lib/delft/detail.action?docID=1895890.

[22] Beom-Seon Jang, Young-Soon Yang, Yu-Suk Song, Yun-Seog Yeun, and Sung-Hee Do. Axiomatic design approach for marine design problems. Marine Structures, 15(1):35 – 56, 2002. ISSN 0951-8339. doi: https://doi.org/10.1016/S0951-8339(01)00015-6. URL http://www.sciencedirect.com/science/article/pii/S0951833901000156.

[23] J. Karremann. Kortenaer klasse (s-)fregatten, 2019. URL https://marineschepen.nl/schepen/kortenaer.html.

[24] M. Kirsch-Pinheiro, R. Mazo, C. Souveyet, and D. Sprovieri. Requirements analysis for context-oriented systems. Procedia Computer Science, 83:253 – 261, 2016. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2016.04.123. URL http://www.sciencedirect.com/science/article/pii/S1877050916301466. The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016) / The 6th International Conference on Sustainable Energy Information Technology (SEIT-2016) / Affiliated Workshops.

[25] A. Kossiakoff, W.N. Sweet, S.J. Seymour, and S.M. Biemer. Systems Engineering Principles and Practice (2nd Edition). John Wiley & Sons, 2011. ISBN 978-0-470-40548-2. URL https://app.knovel.com/hotlink/toc/id:kpSEPPE006/systems-engineering-principles/systems-engineering-principles.

[26] G. La Rocca. Knowledge based engineering: Between ai and cad. review of a language based technology to support engineering design. Advanced Engineering Informatics, 26 (2):159 – 179, 2012. ISSN 1474-0346. doi: https://doi.org/10.1016/j.aei.2012.02.002. URL http://www.sciencedirect.com/science/article/pii/S1474034612000092. Knowledge based engineering to support complex product design.

[27] S. Maro and J. Steghöfer. Capra: A configurable and extendable traceability management tool. In 2016 IEEE 24th International Requirements Engineering Conference (RE), pages 407–408, Sep. 2016. doi: 10.1109/RE.2016.19.

[28] S. Maro, J. Steghöfer, J. Hayes, J. Cleland-Huang, and M. Staron. Vetting automatically generated trace links: What information is useful to human analysts? In 2018 IEEE 26th International Requirements Engineering Conference (RE), pages 52–63, Aug 2018. doi: 10.1109/RE.2018.00-52.

[29] M. Masmoudi, P. Leclaire, M. Zolghadri, and M. Haddar. Change propagation prediction: A formal model for two-dimensional geometrical models of products. Concurrent Engineering, 25(2):174 – 189, 2017. doi: 10.1177/1063293X17698192. URL https://doi.org/10.1177/1063293X17698192.

[30] D.N. Mavris and D. DeLaurentis. Methodology for examining the simultaneous impact of requirements, vehicle characteristics, and technologies on military aircraft design. Proceedings of the 22nd Congress of the International Council on the Aeronaudical Sciences (ICAS), 2000.

[31] J. O Grady. System Requirements Analysis. Elsevier, 2014. ISBN 978-0-12-417130-5.

[32] Netherlands Ministry of Defence. Defensie materieel proces bij de tijd, Feb 2017. URL https://www.rijksoverheid.nl/binaries/rijksoverheid/documenten/brochures/2017/02/01/defensie-materieel-proces-bij-de-tijd/web_201750128_Brochure+Defensie+Materieel+Proces.pdf.

[33] Office of the Chairman of the Joint Chiefs of Staff. DoD Dictionary of Military and Associated Term. Washington DC: The Joint Staff, 2020.

[34] E. Pedersen and H. Engja. Lecture notes in course tmr4725 modelling, simulation and analysis of dynamic systems, Aug 2014.

[35] J.A. Penoyer, G. Burnett, D.J. Fawcett, and S.-Y. Liou. Knowledge based product life cycle systems: principles of integration of kbe and c3p. Computer-Aided Design, 32(5):311 – 320, 2000. ISSN 0010-4485. doi: https://doi.org/10.1016/S0010-4485(00)00014-2. URL http://www.sciencedirect.com/science/article/pii/S0010448500000142.

[36] D.T. Rigterink. Methods for Analyzing Early Stage Naval Distributed Systems Designs, Employing Simplex, Multislice, and Multiplex Networks. PhD thesis, University of Michigan, 2014.

[37] H.W.J. Rittel and M.M. Webber. Dilemmas in a general theory of planning. Policy Sciences, 4:155 – 169, 1973. doi: https://doi.org/10.1007/BF01405730.

[38] J. Rowley. The wisdom hierarchy: representations of the dikw hierarchy. Journal of Information Science, 33(2):163–180, 2007. doi: 10.1177/0165551506070706. URL https://doi.org/10.1177/0165551506070706.

[39] G Schreiber. Knowledge Engineering and Management : The CommonKADS Methodology. The MIT Press. The MIT Press, 2000. ISBN 9780262193009. URL http://search.ebscohost.com.tudelft.idm.oclc.org/login.aspx?direct=true&db=nlebk&AN=21785&site=ehost-live.

[40] D.J. Singer, N. Doerry, and M.E. Buckley. What is setbased design? Naval Engineers Journal, 121(4):31–43, Oct 2009. ISSN 0028-1425. doi: doi:10.1111/j.1559-3584.2009.00226.x. URL https://www.ingentaconnect.com/content/asne/nej/2009/00000121/00000004/art00013.

[41] J.A. Sokolowski and C.M. Banks. Principles of modeling and simulation: a multidisciplinary approach. John Wiley & Sons, Inc., 2009. ISBN 9780470403556.

[42] N.P. Suh. Axiomatic design: advances and applications. New York: Oxford University Press, 2001.

[43] B.J. Van Oers. A packing approach for the early stage design of service vessels. PhD thesis, TU Delft, 2011.

[44] B.J. Van Oers, E. Takken, E. Duchateau, R. Zandstra, S. Cieraad, W. Van den Broek-de Bruin, and M. Janssen. Warship concept exploration and definition at the netherlands defence materiel organisation. Naval Engineers Journal, 130:63–84, Jun 2018.

[45] W.J.C. Verhagen, P. Bermell-Garcia, R.E.C. van Dijk, and R. Curran. A critical review of knowledge-based engineering: An identification of research challenges. Advanced Engineering Informatics, 26(1):5 – 15, 2012. ISSN 1474-0346. doi: https://doi.org/10.1016/j.aei.2011.06.004. URL http://www.sciencedirect.com/science/article/pii/S147403461100036X. Network and Supply Chain System Integration for Mass Customization and Sustainable Behavior.

[46] Wikipedia. Kortenaerklasse, 2018. URL https://nl.wikipedia.org/wiki/Kortenaerklasse.

[47] R. Young. Requirements Engineering Handbook. Artech House, 2003. URL https://ebookcentral-proquest-com.tudelft.idm.oclc.org/lib/delft/detail.action?docID=227599.

A

Decomposition of a requirement

Requirements are statements or constraints to help guide the design of a system in a way that it is useful to the stakeholders in a project; a collection of several requirements can be called a *specification* [8, p. 29]. Generally speaking a requirement is more well defined and specific than a *need* (stated requirements) [47, p. 45]. Any design of a system has numerous specifications and requirements which create documents of large proportions. It is therefore easy to lose the overview in a project. The field of Requirement Engineering focuses on streamlining the creation and management of requirements. From this research field the following definition of a requirement can be taken: a **controlled attribute** with a **relation** expressed in a **value** of a certain **unit of measure** [31, p. 94].

Here are some examples of requirements where this definition is used:

- Valve closure time is less than 0.5 seconds
- Visible length in front of the bow is less than 3 ship lengths

These examples can be seen as the bare minimum of a requirement. In the requirement documentation there is usually more text used to describe a requirement to make it more readable. Also is the *rationale* added to requirements in the documentation; this is the 'why' behind a requirement [12].

In the maritime classification sector the requirements and specifications are often referred to as 'rules'. Within the shipbuilding and shipping sector however, requirements and specifications are commonly called by the same term: requirements. This makes the definition of a *requirement* not straight forward to use. To show the working of this principle two examples from the DNV Ship rules for nautical safety (July 2011) requirements for the field-of-vision for the wheelhouse (bridge) are taken:

1 **B 202** Every effort shall be made to place the bridge above all other decked superstructures in order to obtain the best possible field of vision for safe navigation and manoeuvring of the ship.
2 **B 203** A horizontal field of vision to the horizon of 360° shall be obtained by using not more than 2 positions within the confines of the wheelhouse on either side of the workstation for navigating & manoeuvring and being not more than 15 m apart [13].

In the first rule (B 202) the requirement can be stated as: The place of the bridge shall be higher than - all other decked superstructures. In this requirement there is no value to express the relation to the unit of measure; this requirement is controlling a relation itself. The rationale for this requirement is also included in the rule.

The second rule (B 203) is to be considered as a specification; there are multiple requirements controlling several attributes. The two controlled attributes are the horizontal field of view and the viewing point(s). For an engineer this rule is comprehensible and possible to evaluate as a whole. If a computer has to validate this rule, or to get more insight in the requirements within the rule, the breakdown structure reveals six requirements:

1 The horizontal field of vision from the wheelhouse is more then 360 degrees
2 The horizontal field of vision from the wheelhouse is obtained from not more then 2 viewing positions
3 The viewing point(s) is(are) located within - the confines of the wheelhouse
4 *If there is one viewing position at the bridge*: The viewing position is located next to - the workstation for navigating & manoeuvring

Stated requirements (need)	(Real) requirements
Business User	
High-level	High-level
Functional (operational)	Functional (operational)
	Non-functional (technical specification) Performance (functional specifications for design items) Item specific

Table A.1: Requirement types in the ”stated” and ”real” requirement space

5 *If there are two viewing positions at the bridge*: The two viewing positions are on either side of - the workstation for navigating & manoeuvring
6 *If there are two viewing positions at the bridge*: The two viewing positions are no more apart then 15 m

These examples show that the requirement management for a relative simple attribute can quickly become difficult. Controlling these requirements and complying with them is a tedious task; it is easy to lose control. By managing these requirements and the attributes they control in one single database - controlled by engineers - it should be easier to keep an overview on this part of a complex system.

A.1. Types of requirements

Besides the requirements being divided in stated and real requirements there are different types of requirements. In the *Requirements Engineering Handbook* by Young [47, Ch. 4] there are several different types of requirements defined:

- Business Requirements (need)
- User requirements
- High-level Requirements
- Functional Requirements
- Non-functional Requirements (system specific properties: reliability and safety)
- Design constraints
- Performance Requirements (applied to functional requirements)
- Item specific (System, Subsystem and Component) Requirements

Next to these types there is a type called ”business rules”, which are the basis for creating the functional requirements. These requirements can be said to be the knowledge and rules present at DMO, as well as the more general rules from the classification bureaus.

How requirements types can relate to stated and real requirements is shown in table A.1. Adding more detail to a requirement makes it more ”real” while also narrowing the scope of this requirement. Following this reasoning a stated requirement can be split up in real requirements but not the other way around.

The (simplified) stated requirement from the ’Kustwacht’ were: *fast top speed* and *operate in rough seas*. Since they used the *functionally specified* requirements for the first time they did not translate their stated requirements into real requirements. This step was for the shipyard. In the real requirements as created by the shipyard there was no ’combination requirement’ covering the speed and sea keeping. Still, the created requirements were able to meet the stated requirements. The gap between the need and the requirements were not noticed as there was little communication between the shipyard and the Dutch project team for the RHIBS.

A.1.1. Design constraints

> *”A design constraint, or simply a constraint, is a boundary condition within which the designer must remain while satisfying the aggregate of the performance requirements for the item.”* O Grady [31, p. 105]

Design constraints are special types of requirements and are often in the physical aspect of a design. When a new ship has to perform in an existing environment, this environment imposes constraints on the design. Other aspects could be the physical rules as we know them, budget or schedule constraints [47]. Such constraints tend to set the design space for engineers to work with [31]. Not all these constraints - besides budget and schedule - are immediately present in higher level requirements or documents. For a problem owner stating constraints early on in the process will limit the design freedom later on.

Design constraints often require a *parameter* to be constrained. Such parameters are influenced by the design. Where the design space for these parameters are set by the design constraints it is to a designer to actually determine the value of this parameter. If the parameter is within the set boundaries the requirement is fulfilled.

This sounds straight forward, but it gets more difficult if two parameters are connected and both have contradicting design constraints. Such contradicting parameters are plentiful and are the reason for compromises in a design. Having a possibility to see these contradicting parameters means both a designer and a requirement engineer can then find a solution to this.

B

The catalogue in Shipbuilder

Creating objects in Shipbuilder starts with the initiation of an object type in the Knowledge base. Such an object has to be as general as possible. To use such an object and specify an actual entity, an entity of an object type has to be created in a catalogue. In the catalogue a template of an object (class) is used and modified to represent an entity (e.g. an engine).

Objects added from a catalogue are able to be represented in multiple complex systems. Using object classes from the KB to add an object to a complex system will result in having stand alone objects, reusing such an object requires recreating the object - losing the added value of the KB and catalogue.

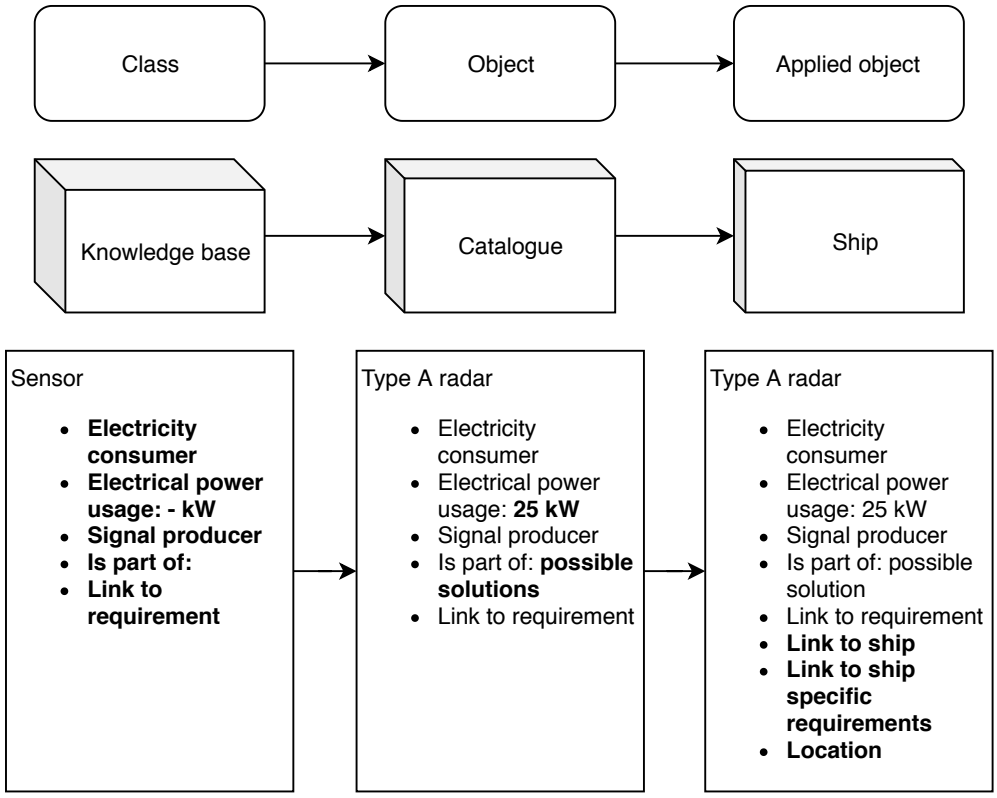


Figure B.1: Representation of the Knowledge Base, catalogue and applied versions of an ćomponentín Shipbuilder

C

Connections of the information model in Shipbuilder

The connections, with affiliation to this thesis, are shown in the table below (figure C.1). In this table there is a distinction between four separate types of connections. These types are for the purpose of providing insight in the connections for this thesis.

		1	2	3	4	5	6	7	8	9	10	11	12	13
1	Mission		1											2
2	Task	1		1			1	1	1					2
3	Function			2	1		2						2	
4	System			1		1	1	1		1	3		2	
5	Component				1		1	1		1	1	3	2	
6	Any requirement	1	1	1	1	1	1	1	1		1	1	2	
7	Parameter		1		2	2	1						2	
8	Ship type		1				2							
9	System function												2	
10	Space					1	1					1		
11	Deck						1				1			
12	Performance			1	1	1	1	1		1				2
13	Effectiveness	1	1	3			1						1	

1	Shown link/connection
2	Not shown link/connection
3	Indirect link/connection

Figure C.1: The current connections in Shipbuilder, relevant for this thesis

Showing the connections in the form of the V-model of DMO, results in the visualisation in Figure C.2. The final information model, at the time of this thesis, resulted in the possibility to create a trace as can be seen in Figure 4.4b in Section 4.3.

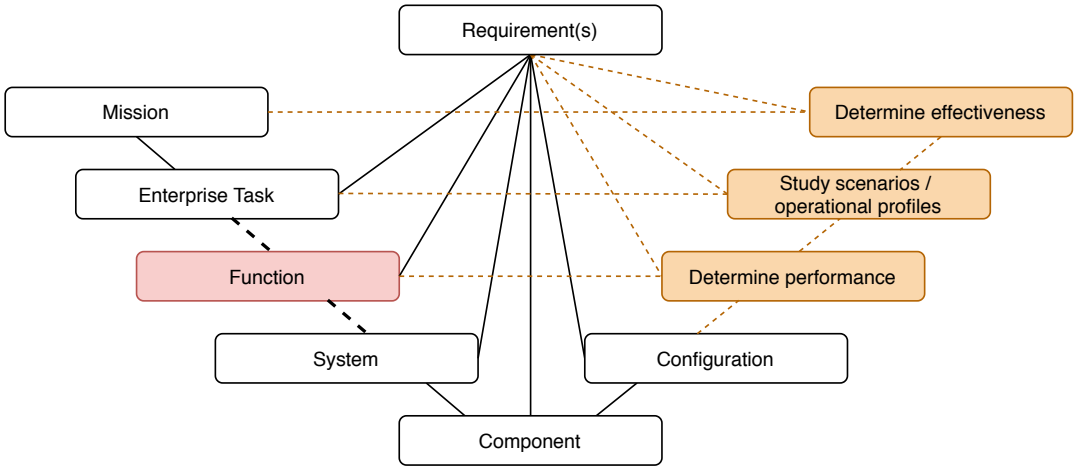


Figure C.2: A visualisation of the final version of the model at the end of this thesis. Yellow objects are activities and the red object cannot yet be placed in a ship.