Integrating vulnerability analysis into the early stage distributed ship system design process

M.F. van Diessen

TU Delft

Defence Materiel Organisation
Ministry of Defence

Integrating vulnerability assessment into the early stage distributed ship system design process

by

M.F. van Diessen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday August 18, 2020 at 09:30.

Student number:	1328697	
Report number:	SDPO.20.017.m	
Project duration:	December 2, 2019 – August 18, 2020	
Thesis committee:	Prof. Ir. J. J. Hopman,	TU Delft, chairman
	Dr. A. A. Kana,	TU Delft, supervisor
	Dr. E. A. E. Duchateau,	Defence Materiel Organisation, supervisor
	Dr. P. Mohajerin Esfahani ,	TU Delft

Defence Materiel Organisation
Ministry of Defence

Disclaimer

The research as presented in this thesis was conducted at the Defence Materiel Organisation (DMO) of The Netherlands Ministry of Defence (MOD). It must be emphasized that physical input, system configurations and operational requirements as used in the testcases do not have any relation with past, current or planned procurement projects at the Defence Materiel Organisation (DMO). All presented input was chosen or altered in such a way that they are realistic, yet not representative for any of the aforementioned projects.

Abstract

Naval ships are designed to fulfil complicated missions in a hostile environment. The inability to perform these missions can have severe consequences, ultimately resulting in the loss of life and the platform. One way to reduce this possibility is to decrease the vulnerability of mission essential distributed ship systems during the design process. Vulnerability reduction is best performed early in the design process, as significant changes with regards to the design are still possible. Traditionally vulnerability reduction measures in this stage are based on best practices, rules of thumb and design rules as details are lacking for a thorough analysis. As these measures are based on past experiences and both the technology and the operational environment are continuously changing, one can debate whether these rules are always applicable. Recently developed vulnerability analysis methods are better able to deal with the available level of detail in the early stage design process. This research focusses on how these methods can be integrated in the distributed ship system design process in order to guide the early stage design effort to produce less vulnerable ship designs.

In the current design process several aspects of the distributed ship systems, such as the component positions, topology and the physical routing of connections, are sequentially determined and optimized. As these variables are dependent this inherently limits the design space after each step, possibly leading to sub-optimal solutions. Therefore a model is developed pursuing a global optimization of these design variables with the objective to maximize the operational output in damaged situations. Based on a constraining physical architecture, a system configuration consisting of networks, components and possible connections, requirements for residual capabilities and one or more threats the algorithm is able to generate designs where the component positions, topology and routing of connections of a distributed ship system are automatically determined. The model uses an implementation of a nested Genetic Algorithm (GA) to achieve this, using various implementations of existing theories from network theory such as a k-shortest path algorithm and hurt-state-percolation theory.

Five testcases are conducted using the model with varying system configurations and damage cases, in order to evaluate the contemporary vulnerability reduction measures in use at the Defence Materiel Organisation (DMO). Analyses of the testcases show that all design rules (principles) can contribute to vulnerability reduction of a design, even when the original assumptions of the design rules are violated. The testcases also show that whether the designs are actually less vulnerable is a matter of finding the right balance between the application of the principles of vulnerability reduction. Traditionally the designer is responsible for finding this balance based on his experience.

By adopting a holistic approach the developed method is able to find the right balance between the vulnerability reduction principles, for different threats and system configurations. This can assist the designer in the concept exploration phase, as survivable concepts are generated based on the resulting operational capabilities and requirement elucidation is facilitated by enabling trade-offs between these capabilities. Furthermore the method can be used to research the influence of new technological concepts and changing operational conditions, as the designer is lacking experience in these cases .

Preface

Before you lies the thesis which results from research conducted to further integrate vulnerability assessment methods in the early stage distributed system design and ship design processes. The research was conducted at the Defence Materiel Organisation (DMO) in 2019-2020 and is part of the process to obtain the Master's degree at Delft University of Technology. During the research period I received support of several people who I would like to thank.

First of all I would like to thank DMO for the research opportunity. I enjoyed working at Afdeling Maritieme Systemen for as long as this was possible, and I appreciated the interest for my research project. I especially would like to thank my daily supervisor from the DMO, Etienne Duchateau. I know the little time you had available, but you were always available for questions when I had them. I enjoyed the meetings we had when we discussed the subject, as well as the less formal talks where you provided more insight in the mind of the naval engineer.

From the TU Delft I would like to thank Austin Kana, for the critical questions in the early stage of my research and all the feedback throughout the whole period, but especially for the guidance with regard to the process. This made the research process considerably more smooth. I would also like to thank professor Hans Hopman as chairman of the thesis committee and for his feedback during the progress meetings.

Last but not least I would like to thank my family and friends for their support. A special thanks to Renate and Tijs. Renate, participating in a Master's program at the age of 36 was a challenge for me, but would not have been possible without your support at home. And Tijs, sometimes you made studying harder, but at the end of the day you put everything in perspective and made the whole effort easier.

I hope you will enjoy reading this report.

M.F. van Diessen
Delft, July 2020

Contents

List of Figures

List of Tables

1 Problem definition

The purpose of this chapter is to provide a brief background of the problem and a clear problem definition. Thereafter the research objective is formulated, followed by the outline and scope of the research.

1.1. Background

All ships are built for a purpose, which means they are created in order to fulfil a certain mission. The systems placed on board of the ship all contribute to the ability of the ship to conduct its mission. However, proper functioning of these systems can be influenced by several circumstances (with internal or external causes), essentially endangering the fulfilment of the vessels intended mission. These circumstances can have internal causes, such as system malfunctions, accidental fires, etc. or external causes due to the environment, such as adverse weather conditions.

Vulnerability[1] analyses of ship designs are conducted to gain insight in the survivability[2] of a ship in such situations. This is of particular importance for warships for two reasons. First, naval ships are operating in an environment where external actors are deliberately trying to increase the chance of mission failure through hostile actions, a so called man-made hostile environment. Second, the inability of a naval ship to fulfil its mission can have more severe effects than for other ship types, as it can lead to loss of lives and the ship itself. In other words, vulnerability analysis is essential for naval ships as they are operating in high risk environments with both an increased probability and increased negative effects of mission failure when compared to other types of vessels.

Traditionally, vulnerability analysis require a ship design with a high level of detail and is therefore conducted in a late stage of the design process [Habben Jansen et al., 2018]. These traditional methods (e.g. RESIST, SURVIVE, Measure of Total Integrated Ship Survivability (MOTISS), Survivability Management Application (SURMA), etc.) require this level of detail, as they take for instance weapon fragmentation and blast effects into account. To properly model the effects of the weapon, detailed information about the ships' structure needs to be available. However, significant modifications of the design in the late design stage to reduce vulnerability are undesirable as implementation might not be possible, or will at least result in significant delays and increased costs. These effects are amplified by the increasing use of interdependent distributed systems on board of complex vessels, meaning modification of one system will most likely trigger modifications of other systems as well. Design rules and best practices are used during the early design stages [de Vos and Stapersma, 2018] to decrease vulnerability and decrease the chance of such large modifications in a late stage of the design process.

[1]For now vulnerability is defined as "the inability of a ship to withstand damage after a sustained hit" [Said, 1995]. This definition will later be revisited.

[2]Survivability is defined as "The capability of a ship and its shipboard systems to avoid and withstand a weapons effects environment without sustaining impairment of their ability to accomplish designated missions" [Said, 1995]

In the past few years considerable research was performed on developing alternative methods and tools for vulnerability analysis and reduction early on in the ship design process, such as the max-flow-between-hubs method [de Vos, 2018], a Markov-based vulnerability assessment method [Habben Jansen et al., 2019c] and the hurt-state percolation method as described by [Duchateau et al., 2018]. These new techniques are better capable of assessing the vulnerability of designs in an earlier stage and therefore offer possibilities to consider vulnerability in relation to operational capabilities (and ship performance in general) more holistically. This means it is possible to shift from the use of design rules towards a design process in which vulnerability reduction is integrated and actively guiding the design effort.

1.2. Problem definition

The problem which will be researched in this thesis consists of two aspects: the validity of vulnerability reduction measures in the current design process (the design rules) and the utilization of new vulnerability analysis methods for vulnerability reduction. These two aspects will now be discussed more thoroughly.

1.2.1. Design rules

In the current design process designs are largely influenced by the design rules and best practices used, amongst which are the rules for vulnerability reduction. To ensure an efficient design process, the validity of the applied design rules is essential. These design rules are however mostly based on experience, meaning that the rules are not necessarily applicable when the assumptions underlying the design rules are changing. Two areas where these assumptions might not be valid anymore are the changing operational environments for ships and changing ship system architectures.

- The operational environment is constantly changing, and has always been. In the Cold War era, ships were mainly designed to operate in large task groups and counter a specific type of (conventional) threat, and the vulnerability of these ships was assessed in relation to this conventional threat. The last decades saw ships designed at the end of the Cold War era being deployed in a different operational environment where they were confronted with more asymmetrical threats. This results in different anticipated damage extents when a ship is successfully attacked. This might influence the way in which the ship should be designed: one hit resulting in one large damage is fundamentally different than 3 hits with small damages and therefore has a different impact on the ships capabilities. This is in line with the observation that vulnerability increases as ships are employed for missions for which they are not primarily designed [Brown, 1997, as cited in [Piperakis, 2013]].

- The second assumption underlying the design rules is the type of system that is to be designed using these rules. Historically, systems were naturally more independent of each other: for instance a propulsion plant was providing propulsive power, whereas generators would provide electrical power. The support systems of the propulsion plant were independent, or depending on the generators for electrical power, but in case of emergency this could be provided by back-up generators or batteries as well. The last few decades systems are becoming more dependent of each other. One example is the emergence of all-electric ships, where both the propulsion and electricity on board the vessel is provided by the generated electrical power. Besides the benefits (increased overall efficiency, reconfigurability, etc.) there is also the increased risk associated with cascading failures [Goodrum et al., 2018]. Where these failures in more conventional system designs only influenced certain parts (and therefore functions) of the complete system (for example only the electrical power on board the ship), these failures could now propagate through the entire system and therefore have more catastrophic effects. As the design rules for vulnerability reduction were initially based on ships with more independent and decoupled systems, the rules might not be fit to reduce vulnerability of interdependent distributed ship systems.

In order to make correct decisions in the early stage design process the effect of these changing assumptions on the validity of the design rules should be examined, as the purpose of the design rules (ensure vulnerability reduction in an early stage) may be nullified when the rules are not fit any more.

1.2.2. Utilization of new techniques

With the emergence of new vulnerability assessment techniques, information regarding vulnerability of designs is available to the designer faster and in an earlier stage of the design process. This vulnerability is mostly presented as post-hit availability of one, or sometimes multiple effectors[3], which is the outcome of the analysis of the logical and physical architecture of the distributed system design [Brefort et al., 2018]. Currently it is up to the designer which actions to take in order to reduce these vulnerabilities. However, in order to make full use of the new vulnerability assessment methods, these methods should be integrated into the design process. This means that the results of the analyses should not only be post-hit availabilities of systems, but rather operational capabilities guiding the design effort towards getting more effective designs from an operational point of view. This is important, as it is only possible to balance vulnerability considerations with other design factors when these considerations are incorporated early on in the design process [Said, 1995].

1.2.3. Summary

Summarizing, designs are largely influenced by design rules and best practices which at least needs to be validated. With the integration of new vulnerability assessment methods in the design process the design rules and best practices could be verified or improved.

1.3. Research objective

Based on the problem definition, the main research question for this thesis is defined as:

How can vulnerability analyses be used to guide the early stage design effort in order to increase survivability, and how does this influence the contemporary vulnerability reduction design rules?

To answer this question, the following research questions need to be answered:

1. How are designs created in early stage design and what factors can be used to influence or steer the design effort in this stage?
   This knowledge is required to indicate in which way the output of the vulnerability analyses can be used to guide the design effort, as this is an iterative process.

2. Which methods exist to assess vulnerability of distributed ship systems and what are their characteristics relevant for integration in the ship design process?
   This knowledge is first of all required to be able to assess the vulnerability of a design. Integration with early stage design methods require that the input and output of these two processes are (or can be) aligned. The limitations and possibilities of each method will be different as they are designed to meet specific demands, which needs to be considered in relation to the ability to deal with relevant operational factors [Habben Jansen et al., 2019a].

3. Which factors of the operational environment are relevant for vulnerability reduction and how are or can they be incorporated to guide the design effort?
   This knowledge is required to assess which parts of the operational domain should be considered when assessing vulnerability for improving designs. To answer this question, it is necessary to determine which interaction exists between factors making up the operational environment (for instance the threat and resulting damage) and the vulnerability of a design.

4. Which design rules and best practices for vulnerability reduction are used in the design process for distributed systems aboard warships and what are they based on?
   This knowledge is required to establish which common practices are used for vulnerability reduction. These practices will later be checked with the developed method to see whether and how they are influencing the design. Knowledge about the foundation of the design rules is necessary to design the testcase.

[3]An effector is a component in a distributed system which ultimately brings an operational effect, such as sensor and weapon systems

The expected end state for this research is a method in which (1) distributed system design is more integrated in early stage ship design, (2) where vulnerability assessment techniques are directly and automatically driving improvement of ship designs and (3) validated or refined vulnerability reduction measures.

1.4. Research design and outline

Figure 1.1 shows the coherence of the research questions and how they are contributing to the research objective. The first three questions will mainly contribute to the first part of the research objective, being the integration of vulnerability analysis in the early design stage. The operational interpretation of the results of current vulnerability assessment methods and how these can be used to steer the design effort towards better designs will be essential for successful integration.

To answer the second part of the research objective, the vulnerability of a design generated with the integrated method will be assessed in relation to compliance with the design rules/best practices. The variations of the testcase (for example damage extent) depend on the determined relevant operational factors and the found assumptions underlying the design rules.

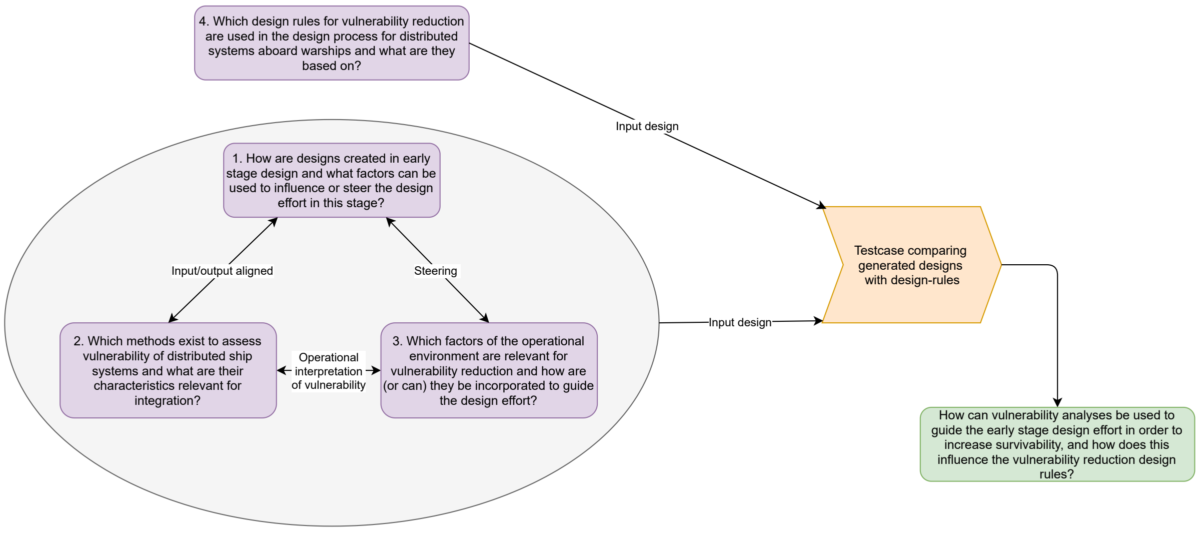

4. Which design rules for vulnerability reduction are used in the design process for distributed systems aboard warships and what are they based on?
Input design
1. How are designs created in early stage design and what factors can be used to influence or steer the design effort in this stage?
Input/output aligned
Steering
Testcase comparing generated designs with design-rules
Input design
2. Which methods exist to assess vulnerability of distributed ship systems and what are their characteristics relevant for integration?
Operational interpretation of vulnerability
3. Which factors of the operational environment are relevant for vulnerability reduction and how are (or can) they be incorporated to guide the design effort?
How can vulnerability analyses be used to guide the early stage design effort in order to increase survivability, and how does this influence the vulnerability reduction design rules?

Figure 1.1: Coherence of research questions

1.5. Scope

The literature study will be used to further scope the problem. There are however a few boundaries and terms which limit the scope of the research from the start:

- There are numerous aspects of vulnerability in ship design, such as vulnerability of system components (e.g. engines, generators, pumps, etc.), structural strength, floodable length and damage stability. Only the systems perspective of vulnerability will be considered in this research.
- Modelling of weapon damage is limited to simple damage cases which are required to achieve the objective. This means for example that damage extents from hits are predefined (e.g. compartment sized) rather than calculated taking ship structure, weapon blast, fragmentation, etc. into account. This is in line with the level of detail available in the early design stage and will be further elaborated on in Chapter 3.
- The research will focus on early stage design methods, as design rules are per definition applied in this stage of the design process. This means the level of detail of the ship and system designs is relatively low, with large degrees of uncertainty and high variability of the configurations considered.

- The allocation of resources to different effectors in case of damage requires detailed load balancing and prioritization of the effectors. This will not be part of this thesis.

2 Literature review

In this chapter the insights gained during the literature review will be discussed. Purpose of this chapter is to establish the current state of the problem environment in order to identify the gap between the current and the desired state of integration of vulnerability analyses within the design process, and to identify which theories might help bridging this gap. After the introduction regarding vulnerability, first distributed system design will be discussed, followed by a brief discussion of the overall ship design process. Thereafter theories regarding vulnerability analysis are examined. Finally factors of the operational environment influencing vulnerability are discussed. With these subjects discussed, also the four smaller research questions are answered.

2.1. Introduction

Vulnerability can have several meanings. Commonly used definitions for vulnerability with regard to naval ships are *"the inability of the ship to withstand damage mechanisms from one or more hits, to its vincibility, and to its liability to serious damage or loss when hit by threat weapons"* from [Said, 1995], or *"the extent to which the operational performance of the ship or system degrades after being hit"* [Habben Jansen et al., 2018]. As this research is specifically focussing on vulnerability of distributed systems these definitions can be combined as:

"the extent to which the operational performance degrades due to the inability of the ship or system to deal with damage after being hit."

From this definition 3 elements can be identified. As these elements together comprise vulnerability, they can also be more closely examined to see how vulnerability can be reduced. The elements identified are:

1. *The extent to which operational performance degrades...* relates to the effect of the damage on the operational performance of the ship. In the end, a ship is designed for a mission as mentioned in Chapter 1, or to generate a certain operational output. In order to assess the consequences of damage on the ships system, one should be able to determine the operational performance in that case and assess whether this is still satisfactory. This is related to vulnerability analyses as will be discussed in Section 2.4.

2. *... due to the inability of the ship or system...* relates to ship system characteristics, and as such this is the part of vulnerability on which the designer has the largest influence. Which aspects of distributed system and ship design are relevant to vulnerability is discussed in Sections 2.2 and 2.3.

3. *...to deal with damage due to a hit* relates to the extent of the damage and its source. In this research the source of the damage is assumed to be a weapon hit, but the type of weapon hit can be varied. This is relevant for vulnerability reduction, as the damage extent might ask for different measures to reduce vulnerability. Although it is not a variable which the designer can influence (a system is designed for a certain threat environment), it might be the case that a

vulnerability reduction measure for a certain damage extent (one large damage) has different effects on a different damage case (multiple smaller damages). The operational environment will be discussed in 2.5.

Summarizing, the designer can influence the operational performance by requirement elucidation during the concept exploration phase (see Section 2.3). The designer can influence the effect the operational environment has on vulnerability (i.e. damage extent) only to a limited extent. However, the way he deals with these topics can guide him in the part of vulnerability he does have an influence on, being the system and ship design, and therewith the ship system characteristics. To examine this more closely, first distributed system design will be discussed, followed by the assessment of damage and thereafter the operational environment.

2.2. Distributed systems design

This section describes the distributed systems design. First of all the description of distributed systems will be discussed, followed by the characteristics of distributed systems and how they influence vulnerability. The next step is a more thorough look into the design process of distributed ship systems, as this determines when and how these characteristics can be influenced.

2.2.1. Description of distributed systems

A common way to describe the organization of systems with a large number of connected, interdependent components is through architectural frameworks [Brown, 2018]. Such an architectural framework specifically for distributed naval systems is proposed by [Brefort et al., 2018]. In this framework a distributed ship system is described using three views of the system (architectures), all containing different information of the system. Combining these three architectures defines the system response (Figure 2.1).

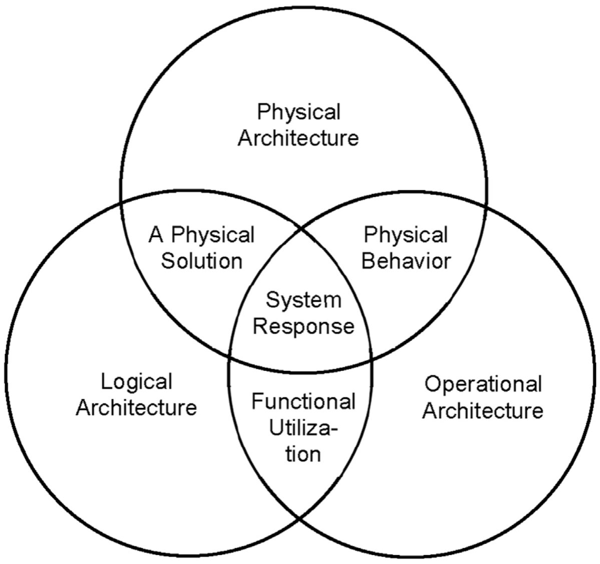


Figure 2.1: Architectural framework for distributed ship systems, from [Brefort et al., 2018]

The physical architecture describes the spatial features of the system. This encompasses the spaces onboard the ship and how they relate to each other, as well as the physical characteristics of the components themselves.

Within these spaces, the components needs to be positioned and connected. The logical architecture of the system defines how the different components are functionally related, i.e. which connections exist between the components. When the physical and logical architectures are combined, a physical solution is obtained (e.g. a general arrangement in which functions are assigned to different compartments). Several methods exist to create this physical solution in the early stage design process. A more detailed discussion of these methods will follow in Section 2.3.

Finally, the operational architecture defines how the system is used over time. As the components generating operational effects (i.e. weapons and sensors, in this context also called effectors) have different states of activity, this has an influence on the supporting networks as the resource requirements for these effectors will consequently change with the activity states of the effectors. The detailed allocation of resources to effectors over time requires detailed load balances and ways of allocating the resources to different effectors dynamically [de Vos, 2018] and will not be part of this thesis. One could argue however that the requirements of simultaneous availability of different effectors is also part of the operational architecture, when the time component is viewed as discrete states of the system, i.e. intact system and different damage states of the system. This is part of the research conducted.

2.2.2. Characteristics

The previous subsection discussed an architectural description of distributed systems. To describe the logical architecture of a distributed system, network theory is commonly used. In the most simple description of a distributed system, the system consists of just two entities: components and connections. In this system the components produce (suppliers) and/or use (users) a certain resource (such as electricity, cooling water, etc.) while the connection transports the resource between the components[4]. When depicting the distributed system using network theory, components are modelled as nodes where connections between components are edges. A special entity is a hub, which can be a component (such as a switchboard) or a common distribution line in a specific network, and which is modelled as a node. An example of such a network representation is shown in Figure 2.2.

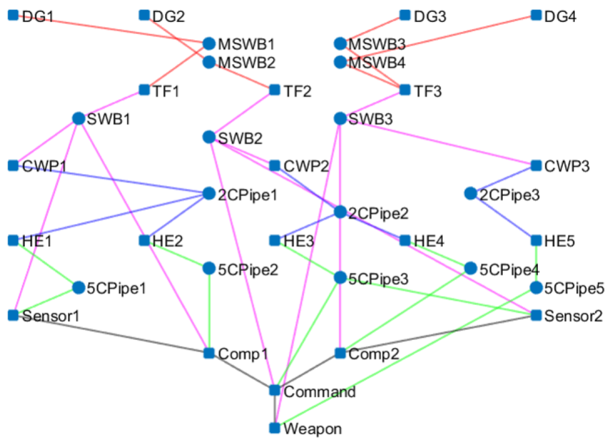


Figure 2.2: Network representation of a distributed system, from [de Vos and Stapersma, 2018]

In the design of the distributed system the components and connections can be varied, and the way in which the components and connections are combined determines in part the performance of the system, which is including the vulnerability of the system. Although not explicitly stated in the reviewed literature, the author identified 5 variables in distributed system design based on [Brefort et al., 2018], [de Vos and Stapersma, 2018] and [Brown, 2018]:

1. Number and type of components with same function. Components with the same function will (when properly implemented in the system) generally result in less vulnerability. This can be illustrated by a system consisting of two identical suppliers and a user in which one supplier is sufficient to provide a user with the required resource. In case of damage, the function of the damaged supplier can be taken over by the other supplier in this system (redundancy). This is part of the logical architecture. It should be noted that improvements in the logical architecture (i.e. increased redundant components) only decrease vulnerability when properly integrated in the physical architecture (see item 3).

2. Presence of connections between components. Increasing the number of connections between certain components will offer the possibility for reconfiguration after a damage case [de Vos,

[4]A more detailed subdivision is provided by [de Vos, 2018], but the subdivision of components in suppliers and users suffice for now

2018]. This is part of the logical architecture. The same reservation regarding integration within the physical architecture applies as for the previous variable.

3. Position of components. The position of components is affecting the vulnerability of systems in two ways. First of all, the absolute position of components is determining the probability that a component is damaged by a hit. This means that (considering a non-uniform hit probability of compartments) some components are more likely to get damaged and therefore the vulnerability of a system is influenced by the position of the component in the ship. Second, the vulnerability is influenced by positioning of similar components relative to each other. For instance redundant components will only decrease vulnerability when they are not influenced by the same damage. This is part of the physical architecture.

4. Routing of connections. Not only the number, but also the routing of the connections will influence the vulnerability of the distributed system. As for the positioning of components, routing has an absolute and relative aspect. This is part of the physical architecture.

5. Protection. Components and connections can be protected against damage effects through additional protection. Examples are blast bulkheads or (when considering secondary damage effects) fire-resistant materials. This is part of the physical architecture.

Generally protection is not named as a factor when distributed system design is discussed. The designer will try to limit the effect of a given damage on the system by altering the system through the first four variables while in case of protection the system is a given and the damage propagation will be limited by applying protection. In some cases the protection is already present as it is required for other functions of the vessel (for instance in the case of bulkheads placed to obtain the desired level of damage stability), while in other cases protection could be added when the desired level of vulnerability reduction cannot be obtained through the system design itself (for instance by the use of blast bulkheads instead of regular bulkheads).

In order to obtain the desired overall system and ship design, the right decision needs to be made regarding these variables. This means that the desired operational performance can be achieved by the system design resulting from those decisions.

2.2.3. Topology design

With the identification of the characteristics of distributed systems influencing the vulnerability, already a rough sequence of activities can be recognized. Before components can be positioned and connections can be routed, the number and type of components and how they are connected should be known. That is, the topology of the distributed system (logical architecture) should be generated first before it can be combined with the constraining physical architecture. De Vos formulated a 14-step design procedure for distributed ship systems [de Vos, 2018] in which the first 6 steps are related to obtaining and assessing a proper topology of a distributed network:

1. Determine the mission related systems the ship needs to execute its mission (typically sensor- weapon- and command systems for naval ships)

2. Determine the type of distributed networks required to support the mission related systems (different electrical networks, cooling water, etc.)

3. Determine the type of distributed networks required to support other distributed networks (e.g. electric network required to support cooling water network)

4. Determine number and type of suppliers of the networks, as users (mission related systems, and suppliers in one network requiring support of other networks) are already known

5. Determine topology of the network

6. Assess the performance of the generated topology

These first 6 steps should in the end ensure that users are connected to the required suppliers in various operational conditions to the largest extent possible. The topologies can be generated manually assisted by design rules and tools (e.g. templates), or automatically [de Vos and Stapersma, 2018]. Manual generation results in a small number of variations, meaning the explored design space is limited. For automatically generating topologies an Automatic Topology Generation (ATG) tool was developed by de Vos [de Vos and Stapersma, 2018], [de Vos, 2018]. The ATG tool and assessment of topology performance will be discussed further in Section 2.4.1. Figure 2.3 show the first 6 steps, including the required information and the relation with the earlier established variables influencing vulnerability of the system. It should be noted that from a vulnerability point-of-view step 6, assessing the performance of a topology, cannot be performed directly with regards to vulnerability. This is because not all variables influencing the vulnerability are determined yet. Instead of assessing the vulnerability of a topology, metrics are used which are linked to vulnerability, for example the number of disjoint paths, hub-hub connections, etc.

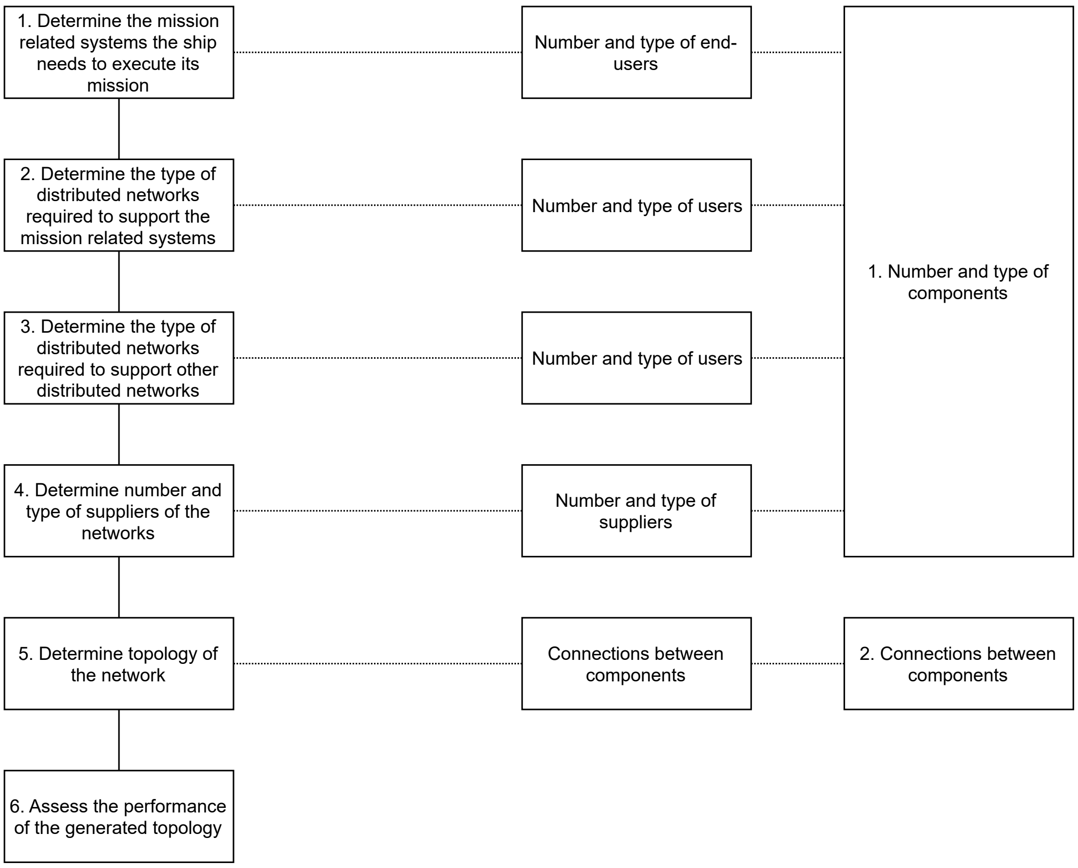
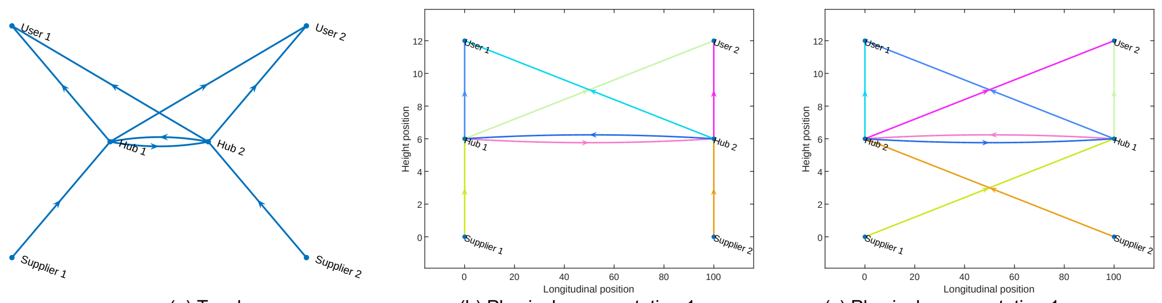

1. Determine the mission related systems the ship needs to execute its mission
2. Determine the type of distributed networks required to support the mission related systems
3. Determine the type of distributed networks required to support other distributed networks
4. Determine number and type of suppliers of the networks
5. Determine topology of the network
6. Assess the performance of the generated topology
Number and type of end-users
Number and type of users
Number and type of users
Number and type of suppliers
Connections between components
1. Number and type of components
2. Connections between components

Figure 2.3: First 6 steps in the integration process

Steps 7-9 (see Figure 2.6 in Section 2.3.4) are related to combining the obtained topology with the constraining physical architecture of the ship, that is placing the components and routing them through the ship. In [Habben Jansen et al., 2019a] this is called modelling the distributed system from a ship perspective. This will be discussed further in the section on early stage ship design processes, Section 2.3. It should however be noted that certain spatial information (approximate location of components) is already required for generating topologies. Without this information connections can be generated between components which are not practical or feasible (e.g. end-users located in the front could be connected to hubs, like Load Centres (LCs) or Switch Boards (SWBs), in the aft part of the ship). An example is shown in Figure 2.4. Figures 2.4b and 2.4c show the same topology (Figure 2.4a). The colors of the connections represent the same connection in both figures. The only differences are the positions of hub 1 and 2, meaning in the second layout the supplier in the aft part of the ship is connected to the hub in the front and vice versa.



(a) Topology (b) Physical representation 1 (c) Physical representation 1

Figure 2.4: Topology and 2 different physical layouts

In steps 10-11 (see Figure 2.6) the system design should be refined not only to ensure that a connection exists, but also that the required amount of a certain resource is available to the end users, so taking the previously discussed operational architecture of the distributed system into account. Steps 12-14 are concerning an increase in the level of detail of the design, and re-evaluating whether earlier made decisions still suffice.

2.2.4. Conclusions

Five design variables influencing the performance (including the vulnerability) of a distributed system design were identified:

1. Number and type of components
2. Presence of connections between components (topology)
3. Position of components
4. Routing of connections
5. Protection measures

The first research question, being “How are designs created in early stage design and what factors can be used to influence or steer the design effort in this stage?” can be partly answered for the distributed system point of view. Only the topology can be generated without extensive knowledge of the constraining physical architecture. Without information of the associated physical architecture, topology design can only be driven by optimizing metrics related to vulnerability, which will not necessarily decrease vulnerability. An example of such a case is a topology where the number hub-hub connections are maximized, believing this increases the reconfigurability. However with inadequate positioning or routing (e.g. all hubs or connections are damaged in one hit), maximizing the hub-hub connections is not decreasing the vulnerability at all.

The following section discusses early stage ship design processes. This will provide insight in when and how the remaining variables are determined, as well as the relation between ship design and the first 6 steps of the integrated design process as discussed above.

2.3. Early stage ship design processes

The ship design process is generally divided in 3 stages, being the early stage design, the contract design and the detail design stages. The latter two stages are concerning increasing the level of detail of a single design, but the ship design is already defined to a large extent in the early stage ship design [Droste et al., 2018] and therefore the main focus during this research. Early stage ship design itself can be subdivided in different processes. The name and exact definition of each process is different depending on the design methodology which is used, but in the end they have the same goal: generate a good understanding of the design problem and trade-offs, and generate a (limited number of) concept(s) which are feasible and can fulfil the desired mission. As this research is conducted at the Netherlands Defence Materiel Organisation (DMO), the terms and processes used at DMO will be used for this research, being concept exploration and concept definition [van Oers et al., 2018].

2.3.1. Concept exploration

During concept exploration a large number of alternative designs is evaluated on their feasibility and operational performance. Due to the large number of concepts, the level of detail of the designs is limited. The purpose of concept exploration is twofold. The first purpose of concept exploration is to refine requirements based on performance levels of the different concepts and their technical and financial feasibility. The second goal is, based on the refined requirements, to generate and select the most promising concept(s) to develop further. How vulnerability requirements are formulated and generation of designs will be further discussed.

Requirements

As stated before, the concept exploration phase contributes to the formulation of requirements. Vulnerability requirements can be formulated by showing the effects of different arrangements on the operational performance in case of damage. This will show, given the starting points for the exploration effort, what is possible to achieve. Although a thorough understanding of the possibilities and limitations due to design choices is essential to reduce vulnerability, the requirement elucidation process itself is out-of-scope for this research. This is not to say that the results of this cannot be used for requirement elucidation. The results of the process are however relevant, as the design procedure needs to be able to deal with the type of requirements that is the result of requirement elucidation. At DMO the vulnerability requirements relevant for distributed systems are captured in a Residual Capability Matrix (RCM).

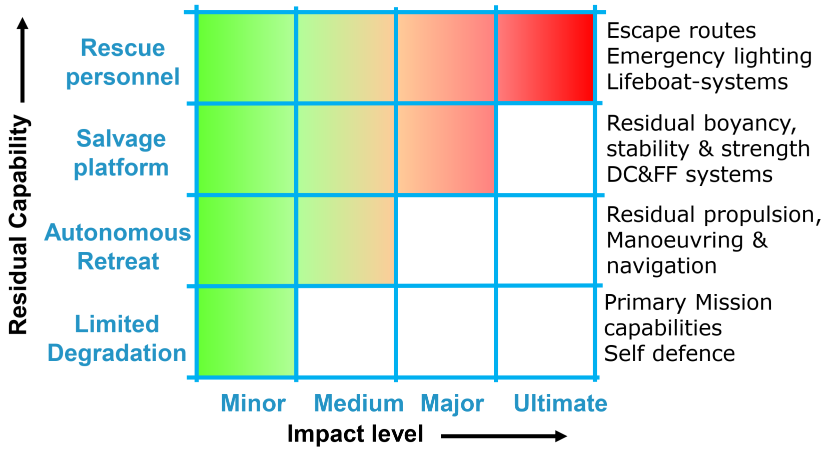


Figure 2.5: Example of a RCM for a support ship, from [van Meurs, 2019]

The RCM defines which capabilities are required after a certain damage is sustained. As indicated by the name, the requirements are formulated at the capability level, being for example mobility or the warfare areas (Anti-Air Warfare (AAW), Anti-Submarine Warfare (ASW), Anti-Surface Warfare (ASuW)). The extent to which a certain capability needs to be available depends on the damage extent. For example, after a large hit only short-range AAW might need to be available for self-defence, while for a small hit the long-range AAW capability needs to be available as well. In practice however the exact quantification of which specific capabilities should be available, to what extent after which impact is hard to determine as there are no well defined boundaries.

Synthesis models

During the concept exploration stage common variations are the hull, systems and their arrangement [Duchateau, 2016]. A large number of design solutions with these variations is obtained using a ship synthesis model. As these models define the concepts which' performance will be evaluated, it is worthwhile to see how these models generate the design solution, and how the designer can influence the development direction of solutions. There are several methods to obtain different designs during the concept exploration phase. The most frequently used synthesis models appearing in recent research are briefly discussed here.

- *Design Building Blocks* [Andrews, 1998], [Andrews et al., 2012]. This method was developed by University College London (UCL). The Design Building Block (DBB) approach uses functional building blocks which are based on a first broad intent. These building blocks are positioned, after which the arrangement is analysed for performance. Based on this outcome, the designer is responsible for manipulating the design until the desired performance is reached. Thereafter the building blocks can be further subdivided and refined until a rather detailed design is obtained. This method requires an experienced designer able to understand the consequences of the decisions he is making based on the calculated performance. The number of solutions generated by this method is limited due to the number of tasks the designer needs to perform.
- *Intelligent Ship Arrangements* [Parsons et al., 2008]. This method was developed by the University of Michigan. The Intelligent Ship Arrangements (ISA) method use a 2D description of spaces, which are allocated to zone-decks (a specific deck within a zone). The allocation and subsequent positioning within the zone-decks are optimized using a Genetic Algorithm (GA), which will try to satisfy the preferences of spaces for certain decks, adjacency / separation preferences between spaces and find an optimal deck utilization. This method only considers these geometrical objectives, without evaluating the influence on the performance of the total vessel (the influence of the positioning on performance can be limited by defining the boundaries of the earlier mentioned geometric objectives).
- *Packing* [Van Oers, 2011], [van Oers et al., 2018] [Duchateau, 2016]. This method was researched by Delft University of Technology (DUT). The approach generates designs by packing 3D objects (including the hull form) in a larger space. The systems, hull form and arrangements can be varied. Designs are varied and optimized by the use of a GA, which can be steered through the definition of (opposing) objective functions by the designer.
- *Subdivision Block Compartment Allocations* [Brown and Waltham-Sajdak, 2015]. This method was developed by the Virginia Polytechnic Institute and State University (VT). In this method a preliminary hull form and deckhouse is defined first, including a subdivision with decks and bulkheads. The spaces created are the Subdivision Block (SDB). The required equipment is divided in clusters based on logical relationships and assigned to the various SDB, thus obtaining a physical arrangement. The design process is guided by Multi-Objective Genetic Optimization (MOGO), meaning the designs are generated and optimize towards multiple objectives. The large extent of automation of the design generation process facilitates the desired large number of concepts to consider.

Although all methods differ in detail, two common features can be identified. First of all, most models use 3D building blocks with assigned functions for which arrangements are generated automatically. Second, the arrangements are optimized by a GA towards objectives which are defined by the designer. Also, design of distributed systems is not explicitly integrated in the synthesis models, but the output of the models (system configuration and positions) has a significant influence of the design of the distributed systems.

2.3.2. Concept definition

Once the relevant designs are considered, one (or at least a limited number of) concept(s) is selected to be developed in further detail. While the concept exploration phase was aimed at generating several solutions to the design problem with sufficient variety in order to refine the requirements, the concept definition is aimed at de-risking the design [van Oers et al., 2018]. By adding more detail, for example routing of connections, the performance and feasibility can be assessed with a higher fidelity.

This does however also mean that it becomes harder to make large modifications, as this means that time consuming calculations and design efforts needs to be repeated. Preferably the decisions made in the exploration phase which have a large impact on other aspects of the design should therefore be the correct ones. This can be illustrated when considering the powering concept. During concept exploration a certain type and number of engines is chosen (e.g. Combined Diesel-Electric and Diesel Engine (CODLAD) or Combined Diesel-Electric or Gas Turbine (CODLOG)). Modifications during the concept definition phase should be prevented or at least be limited to a certain bracket, so that there is no influence on the overall ship design This means for example that the chosen powering concept (i.e. the type of engines like diesel, gas turbine, etc.) remains the same, but the choice of the actual engines could still change. That is, the changes should still fit the building blocks used in the concept exploration phase. These components have a large influence on the other parameters of the design, as for example their possible position in the ship is limited due to the size and weight of the components. The latter is also the case for most of the effectors. Most of the effectors are on the open decks, and although weight and size limitations might be less in absolute terms (with exception of the large sensor masts and large gun and missile systems), their position is still limited by the arcs they can cover (i.e. all effectors prefer a coverage of 360°).

During concept definition the ship design is getting more detailed, as are the performance prediction tools. At DMO the ship synthesis tool during concept definition is called Functional Integrated Design Exploration of Ships (FIDES) [van Oers et al., 2018]. This tool delivers a product data model which can be used by different performance tools.

Although concept exploration and definition are represented here as two distinct and purely sequential processes, in reality this is not the case. More detailed analysis of concepts may lead to new questions requiring additional exploration efforts.

2.3.3. Relation with distributed ship design

Relating concept exploration to the earlier section of distributed system design, it is clear that the choice of which components to use for the distributed systems is largely determined during the concept exploration phase. This is at least true for the effectors and large suppliers of resources (e.g. diesel engines) which are assigned a designated own space. The number and type of other (smaller) components such as the number and position of chilled water plants might be more flexible, and could be delayed until the concept definition stage (but still within the constraints of the chosen concept resulting from the concept exploration).

As mentioned in Section 2.2.3, steps 7-9 of de Vos of the integrated design process relate to the ship perspective, while in parallel, the system should be further developed from the systems perspective (steps 10-14) as argued in [Habben Jansen et al., 2019a]. Habben Jansen incorporated two additional steps. This is shown in Figure 2.6 (additional steps are 15 and 16), again with the relation with the earlier distributed system variables influencing vulnerability of a system.

2.3.4. Conclusions

The previous section discussed how topologies are generated and can be influenced. The way in which the number and type of components, their positions and to a certain extent how the physical implementation of connections are determined was discussed in this section:

- During concept exploration ship synthesis models are used which are capable of automatically generating arrangements using 3D building blocks. This means a large part of the variables influencing distributed system design (major components and positions) are chosen during this stage by using the building blocks, as these leave limited room for changes in later stages of the design where distributed system design is explicitly considered.

- During concept exploration, ship arrangements are created by the use of optimization algorithms provided by user-defined objective functions. The number, type and position of the large components is more or less fixed after concept exploration. When the influence of these factors on vulnerability should be taken into account, the GA should be fed with feedback on how certain design choices impact vulnerability.

- Decisions during the design process ”lock” certain aspects of the design. As wrong decisions likely requires rework in later stages of the design process, the decisions should be taken at the latest time possible (to maintain flexibility as long as possible), and verified to the extent possible

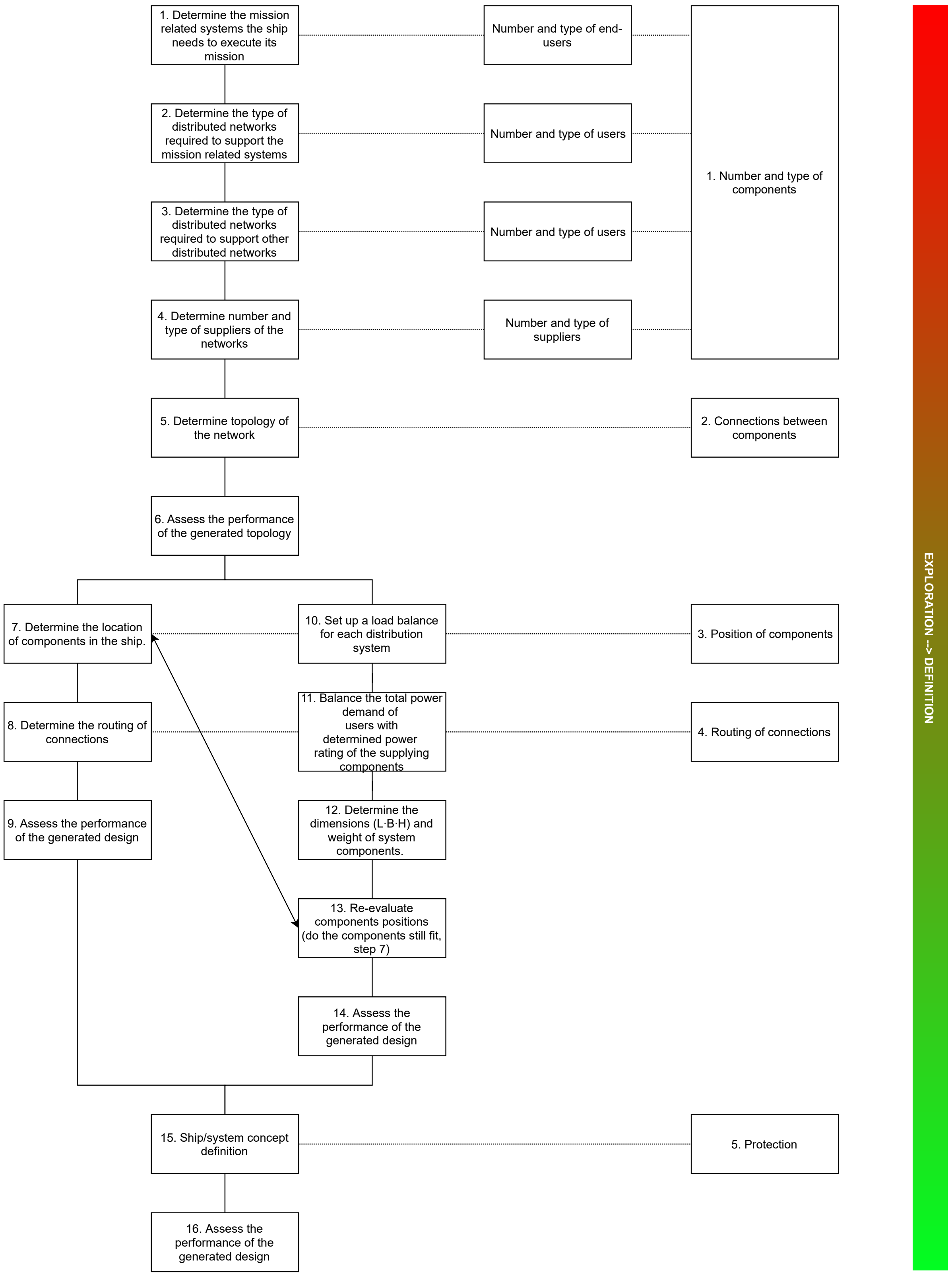

1. Determine the mission related systems the ship needs to execute its mission
2. Determine the type of distributed networks required to support the mission related systems
3. Determine the type of distributed networks required to support other distributed networks
4. Determine number and type of suppliers of the networks
5. Determine topology of the network
6. Assess the performance of the generated topology
7. Determine the location of components in the ship.
8. Determine the routing of connections
9. Assess the performance of the generated design
10. Set up a load balance for each distribution system
11. Balance the total power demand of users with determined power rating of the supplying components
12. Determine the dimensions (L·B·H) and weight of system components.
13. Re-evaluate components positions (do the components still fit, step 7)
14. Assess the performance of the generated design
15. Ship/system concept definition
16. Assess the performance of the generated design
Number and type of end-users
Number and type of users
Number and type of users
Number and type of suppliers
1. Number and type of components
2. Connections between components
3. Position of components
4. Routing of connections
5. Protection
EXPLORATION --> DEFINITION

Figure 2.6: Integrated distributed system design process

to decrease the chance of rework (i.e. they should be informed decisions). These principles are also known as set-based design [Singer et al., 2009], [van Oers et al., 2018].

This means that the designs in an early stage can be influenced by using objectives to which a GA can optimize. In order to be able to feed the GA with feedback concerning the impact of choices on the vulnerability of a design, first the vulnerability of the designs should be determined. In the integrated approach of Figure 2.6 this assessment takes place at the steps 6, 9, 14 and 16. The methods available to assess vulnerability will be discussed in the next section.

2.4. Vulnerability assessment methods

In order to be able to integrate the information gained from vulnerability assessment methods into the early stage ship design process, it is necessary to be aware what the output of the different methods is, and what additional information they can provide which might be relevant to improve the design. An overview of different methods, developed by DUT and DMO, and their possible place in the integrated early stage design process was provided by [Habben Jansen et al., 2019a]. These methods will be discussed in the following subsections. Several methods in other engineering branches are shortly discussed in the last subsection.

2.4.1. Max-flow-between-hubs

The first method considered is the max-flow-between-hubs method. This is actually not a method to assess the vulnerability after a system design is created, but rather a method to limit vulnerability during distributed system design based on network metrics (i.e. an a-priori method as described by [de Vos, 2018]). It is one of the metrics associated with vulnerability as discussed in Section 2.2.3. The method focusses on limiting vulnerability by increasing the reconfigurability of a system. This means that after sustaining damage alternative paths are available to supply users with the required resources. Increasing reconfigurability is achieved by increasing the number of connections between hubs, for example by applying crossovers in a chilled water network or connecting switchboards in the electrical power network.

The max-flow-between-hubs function is an objective function which is designed to be used in the ATG tool by [de Vos and Stapersma, 2018]. The ATG tool generates topologies using the Non-dominated Sorting Genetic Algorithm II (NSGA-II), with the opposing objective functions of system claim (i.e. minimisation of the total number of connections to minimize costs) and a robustness functions, which is the max-flow-between-hubs function. This function tries to maximize the amount of independent connections between hubs. The function favours the addition of connections to hub layers with a small amount of hubs first, as these layers are supposed to be more vulnerable. Simply put, when the max-flow-between-hubs function would be used as a stand-alone metric to assess the vulnerability of a distributed system design, systems with a lot of hub-hub connections would be deemed less vulnerable than other system designs[5].

The max-flow-between-hubs method can be used to create topologies, assess the potential vulnerability of these topologies and optimize them through the ATG tool. This is a method to execute the first 6 steps of the integrated distributed design process as shown in Figures 2.3 and 2.6: applying the max-flow-between-hubs function to a topology is step 6. The required input is limited, being the number of components, which part of the network these components belong to and the type of node they present (supplier, hub or user). With this information the topology can be generated, thereby determining the second design variable of distributed system design (the presence of connections between components). The limited amount of required information makes it a suitable method to apply in the earliest stages of distributed system design.

The choice of which components to choose and their number however is still defined by the designer, and it is left to the designer to vary this during concept exploration and determine the preferred choice. Another limitation and difference with the other methods is that, due to the chosen perspective (reconfigurability) and the stage for which the method is designed (concept exploration of distributed system topologies), different damage extents cannot be considered as spatial information is not (or only very limited) available and considered. The method doesn't consider damage at all, but instead optimizes

[5]As de Vos states in [de Vos, 2018], system designs cannot be simply compared when the number of hubs of the design differ due to normalisation of the total number of hub-hub connections within a hub layer

the chance of a path being available between supplier and user in case of an arbitrary damage. Although this is inherent to the goal and chosen perspective and not a problem for design of different topologies, the method is probably less suitable for the purpose of this research in which the design rules need to be tested for different damage extents.

2.4.2. Markov-based vulnerability assessment

The Markov-based vulnerability assessment method determines the probability that systems are available after one or more hits using a discrete time Markov chain [Habben Jansen et al., 2019c] and [Habben Jansen et al., 2019b]. In this context a system is actually a chain of systems, resulting in the availability of a weapon system, propulsion system, etc. The systems are modelled using network theory, with the systems consisting of components modelled as nodes and connections between components as edges. Each distinct path between a supplier and an end-user is regarded as a system: that means that an end-user which can be fed with a resource via two (redundant) paths is considered as two systems. In this method, the ship is considered to be in a certain state, which is defined as the combined status of all systems. Systems are either available or unavailable, there is no status considered where systems are partially available (degraded)[6]. In case of a ship consisting of two systems this means that the ship has 4 states: either both systems are available, both systems are unavailable or one of the systems is available. In the intact state, all systems are supposed to be available.

When the ship is sustaining damage, the state of the ship might change depending on the location of the hit. Whether the change will occur is defined by a transition matrix T. This matrix contains the probability that the ship will transition from one state (e.g. all systems are available) to another state (e.g. only one certain system is available, or all systems are unavailable). The probability of this transition is determined based on two parameters. The first parameter is the presence of components or paths of a system in compartments. If a path is hit, the system becomes unavailable. When a node is hit, all systems containing that node will become unavailable. The second parameter is the hit probability of the compartments.

The Markov method requires information on both the distributed system design and the ship design. First of all, the topology of the distributed system needs to be known. This topology needs to be routed through the ship. The resulting paths serve as input for the transition matrix. Secondly, the hit probability of the different compartments needs to be known, also as input for the transition matrix. Thirdly, from the ship design the location of the nodes and the routing of the connections through the spaces needs to be known.

The initial output of the method is the probability that a system is available after a certain number of hits. Habben Jansen combines these probabilities and translates them to different level of residual capabilities, thereby quantifying the effect of the damage on the capabilities of the ship. These residual levels of capabilities need to be defined by the designer with the required systems to achieve each level. Besides this vulnerability metric, the Markov method can provide valuable insights for the designer by determining the eigenvalues and eigenvectors of the transition matrix as introduced in [Habben Jansen et al., 2019c]. In this paper explicit formulations are obtained for the different states which are based on the eigenvalues and -vectors of the transition matrix. These explicit formulations can be used to see what effect changes in the transition matrix (i.e. changing the presence of components and paths in compartments) have on the probability to reach (or remain) in a certain state. This can for example be used to maximize the probability to reach a certain state (i.e. maximize the availability of a system), and subsequently examine the effects this change has on the probability of reaching the other states using the other explicit formulations. This could be used to examine trade-offs between design choices affecting the vulnerability of different systems.

The required input of the Markov method makes it suitable to assess the vulnerability from the ships' perspective, being step 9 in the integrated distributed system design process. One of the strengths of the method is the ability to consider a large number of hits on a infrastructure without a dramatic increase in computational effort. This is due to the fact that for one hit, all possible hit locations are considered at

[6]It is possible to consider these in-between-status, but this will make the calculations computationally more demanding meaning the size of the networks to be considered become smaller

once through the use of the transition matrix. The extent of the damage due to a hit is depending on the detail of spatial arrangement of the ship: when the spaces are large, only large hits can be considered and the probabilities of connections/components being hit in the transition matrix will increase. The computational effort required to perform the calculations is mainly dictated by the number of systems (connections) c considered, as this determines the number of states the ship can be in. The number of states is 2^c, and the matrix T has a size of $2^c \times 2^c$. This means that in the current implementation by Habben Jansen the number of systems is limited to 16 [Habben Jansen et al., 2019a]. This makes the method suitable for implementation in an early stage of the design process, where the design contains a low level of detail. Still, the number of systems is probably be too low when a complete distributed ship system network consisting of multiple networks needs to be considered, as every disjoint path is seen as a single system.

2.4.3. Hurt-state-percolation

The hurt-state-percolation method is part of a tool developed by Duchateau [Duchateau et al., 2018] and based on a vulnerability assessment approach developed by van Leeuwen [van Leeuwen, 2017]. The tool is used to route a distributed system topology through a spatial layout. The tool uses different network representations of the spatial layout. For routing, a network representation is used presenting different spaces which can be connected to each other (the routing adjacency network). This might differ from the physical spatial layout, as spaces through which routing is not possible/desirable (e.g. tanks, or spaces separated by blast bulkheads) are not connected in this network. With a distributed system topology known, the tool generates a number (k) of physical paths between components which needs to be connected through a k-shortest path algorithm. So for each connection, several possible paths are generated. Subsequently a number of combinations of paths is chosen and fed to a GA. This algorithm will search for paths with the smallest total length and vulnerability.

The vulnerability is determined through the hurt-state-percolation method. This method uses a routed topology in combination with a damage adjacency network. This is another representation of the physical layout, showing whether damage could propagate to adjacent spaces. A hit to a compartment is simulated, and the impact of the damage (depending on the damage extent and damage adjacency network) on the compartments is determined. The affected nodes and edges of the routed topology are determined, and from this a new, damaged topology is created. The vulnerability is then assessed by determining whether user-defined critical user(s) are still connected to the required resources (this is where the name of the method, hurt-state-percolation is based on) through the use of a search tree [Duchateau et al., 2018], [van Leeuwen, 2017]. This is repeated for several damages, in the end providing a probability that in case of a hit the defined critical user(s) are still available. In the tool, the GA will minimize the probability of the defined critical users being unavailable, thereby minimizing the vulnerability of the chosen paths.

The hurt-state-percolation method requires both a topology and spatial layout of the ship, with fixed locations of the nodes. For the vulnerability assessment a routing through the ship needs to be known as well, but this can be generated automatically by the described tool. The computational limitation of the hurt-state-percolation method is not the number of nodes and connections, making it suitable to analyze both low- and high-level of detail distributed systems. The largest computational limitation at this moment is the number of hits h and (to a much smaller degree) the number of spaces in the design n, as the number of hit possibilities are $C_h = \binom{n+h-1}{h}$. This limits the current use of the method to simulate one or two hits. The testcase in [Duchateau et al., 2018] used a vessel consisting of 45 compartments resulting in evaluation of respectively 45 (1 hit) and $\binom{46}{2} = 1.035$ (2 hit) hit cases. Expanding this to 3 hits would increase the number of hit cases to 16.215. There are however relative simple measures which could be taken to ease the computational load (e.g. only consider the occupied compartments).

The primary output of the hurt-state-percolation method is the availability of user defined systems (critical users) rather than residual capabilities of a vessel in damage cases. Besides this, the nature of the method (using the search trees) provides the ability to determine why systems are unavailable, i.e.

which nodes or paths are failing causing the unavailability. The ability to handle large networks makes the method suitable to analyse designs with a higher level of detail, corresponding to step 16 in Figure 2.6.

2.4.4. Vulnerability assessment in other fields

Vulnerability analysis is not only relevant for distributed ship systems, but can be used for increasing the reliability of any distributed system or network. Several examples are discussed in this subsection.

One of the applications of vulnerability assessment is the evaluation of road networks. The sensitivity of the network to unavailability of a road or a traffic junction (e.g. due to traffic incidents, natural disasters or terrorist attacks) is important as interruption of the traffic flow can significantly impact daily operations. Several techniques exist to measure this robustness. A comprehensive overview of recent research is presented in [Mattsson and Jenelius, 2015], where the methods are divided in two trends, being the analysis of topologies or flows. Applicability to the ship problem is limited, as vulnerability in transportation cases is often quantified by metrics which are less relevant in distributed ship systems (e.g. waiting times, length of detour, efficiencies, etc.). More importantly transport systems are considered separately (e.g. either the road network or the public transportation network are considered) while the distributed systems onboard are interdependent.

Although still a single network, a land-based power network is the same type of network as found onboard. In [Koç, 2015] the robustness of a power distribution network is determined analysing several network metrics, by removing single nodes and edges. This neglects the physical dimension of the network, i.e. the possibility that multiple nodes are damaged at the same time is discarded. This is a viable assumption when the nodes in the network are far apart and the damage is caused unintentionally (e.g. due to adverse weather conditions). This assumption does not hold for ships though, where simultaneous damage to multiple nodes and edges is more likely.

Networks more resembling the ones onboard are in civil engineering often referred to as critical infrastructure systems as described in [Zimmerman, 2004]. The vulnerability of these systems are however often analysed using network flow models ([Ouyang, 2017], [Ouyang et al., 2018]), while this level of detail is not yet available in the early design stage for damaged cases. Moreover the proposed metrics are often either valid for a single network, a single architectural layer (i.e. only the logical layer) or for a single damage.

Due to the ship-specific properties of the problem (multiple dependent design variables, multiple networks, variation of operational environment, etc.) and the limited available information in the early design stage, the vulnerability assessment methods will be limited to the 3 earlier discussed methods for the remainder of this research.

2.4.5. Conclusions

Each of the 3 discussed vulnerability methods require different inputs and provide different types of output. This makes them suitable for different stages in the design process as indicated above. The max-flow-between-hubs method is unique in the sense that it requires a low level of detail as input and evaluates vulnerability from a reconfigurability point of view. Both the Markov method and hurt-state-percolation require the same type of input (both methods require the components, topology, location and paths) but differ in the level of detail the method can handle to the computational possibilities in the current application. The fact that hurt-state-percolation is used in a later stage of the design process as suggested in Figure 2.6 and [Habben Jansen et al., 2019a], doesn't necessarily mean that it is inadequate to be used during an earlier stage of the design process. Whether to employ hurt-state-percolation or the Markov method is then more a choice of which factors to consider and which output is preferred. The Markov-method provides residual capabilities and generic design insights following a large number of hits for smaller networks, while the hurt-state-percolation method can evaluate larger networks for a limited number of hits resulting in system availabilities with specific failure causes.

2.5. Integrating operational context

As described in the introduction of this thesis a ship is build for a certain purpose. The extent to which a ship or design is able to fulfil this mission is the effectiveness of the ship. As shown in Figure 2.7 the effectiveness of a design can be seen as a composite of the means (system capabilities), the environment and the ways (operations) [Brown, 2013]. The system's capability is largely defined by the design. How a design is conceived was described in the previous sections. This section will start with a discussion about which factors of the operational environment (Section 2.5.1) and operations (Section 2.5.2) are relevant in the context of vulnerability and how these factors can be taken into account in the design process (Section 2.5.3). The section will end with methods which can be used to quantify effectiveness.

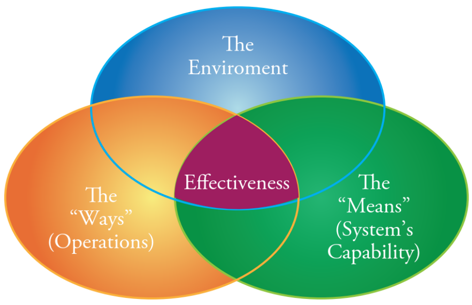


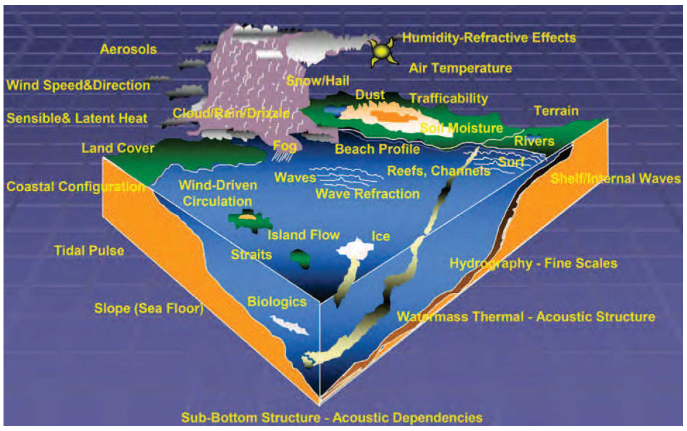
Figure 2.7: Operational output, from [Brown, 2013]

2.5.1. Environment

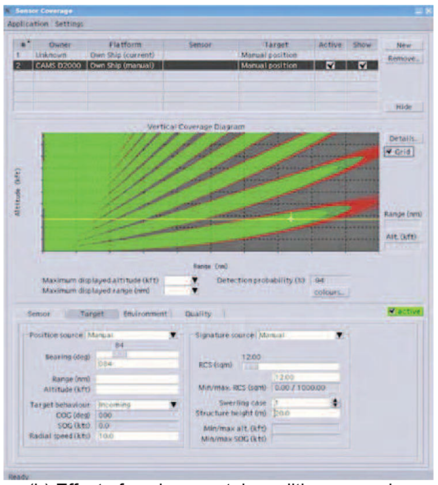
The operational environment can be defined as *"A composite of the* conditions, circumstances*, and* influences *that affect the employment of capabilities and bear on the decisions of the commander"* [US DoD, 2019]. It is one of the parts of the design problem on which the designer has limited or no influence. The operational environment will however influence the distributed systems and ship design, as the design should be able to deal with the environment in a satisfactory matter.

The first two factors, being the conditions and circumstances affecting employment of capabilities, are for example meteorological and oceanographic conditions in which the ship will operate. This will impact the effectiveness of a design, as performance of effectors is influenced by these factors, and this performance is normative for the extent to which a design can avert damage (i.e. the susceptibility[7]). As such, conditions and circumstances are not relevant from a vulnerability point of view, as this is related to the ability of the ship to deal with damage *given a hit*. The third factor, being the influence of the operational environment, is relevant: this is the existing threat which is causing damage.

[7]Susceptibility is defined as "The inability of a ship to avoid being damaged in the pursuit of its mission and to its probability of being hit" [Said, 1995]



(a) Conditions and circumstances, from [NL MoD, 2014]



(b) Effect of environmental conditions on radar performance, from [NL MoD, 2014]

Figure 2.8: Operational environment

In the context of this thesis a threat is considered to be a weapon which can impact the ship. This means that for example weapon carriers (submarines, ships, aircraft, etc.) are excluded unless used as a weapon. A weapon has a few characteristics as listed below which are relevant for determining the vulnerability. These characteristics have an influence on how the weapon affects the ship. High-fidelity weapon modelling can be performed to examine these effects, taking for example fragmentation and ballistics after explosion into account. For the early design stage a more basic weapon modelling would suffice, as this is more in line with the low level of detail a design contains in this stage. The modelling possibilities below are (a) in line with the level of detail for early stage ship design, (b) the parameters can be used in the vulnerability methods discussed in Section 2.4 and (c) are deemed to be sufficient to reach the objectives for this thesis. The latter means that with this modelling different types of impacts can be simulated (i.e. small vs. large hits from different domains) without going into detailed weapon modelling, as this is not the objective of this research. The list is deduced from [Goodfriend, 2015] with the author's own interpretation.

- Domain of the weapon and trajectory. The domain of the weapon can either be sub-surface, surface or air, where the trajectory describes the path a weapon follows (e.g. sea-skimming, ballistic, high-diving, etc. for air launched targets). This is relevant as it will influence the hit probability of compartments and systems onboard the ship[8]. Assessing the vulnerability of a design against the threat of an anchored mine will mean the hit probability of the compartments below the waterline will be larger than for compartments well above the waterline, while this is opposite for high-diving missiles.
- Speed of the weapon. In general, weapons with a higher speed will have a larger penetrating ability than weapons with a lower speed. This can be reflected by a higher hit probability of the inner compartments (near the ships' centreline) vs. the outer compartments.
- Fuze setting of the warhead (proximity, impact, delay, etc.). The fuze setting of a warhead is relevant for the same reasons as the speed of the weapon: it influences the hit probability of certain compartments.
- Warhead size. Depending on the payload of the weapon, weapon impact will have varying effects on the design: a small payload will only affect the hit compartment, where a large payload will also affect the surrounding compartments. This can be modelled with several levels of detail suitable for early stage design. The first method is to look at the space or compartment level[9].

[8]Hit probability in this thesis will refer to the probability that a compartment or system will get hit, given the fact that the ship will be hit. It does not refer to the probability that the ship will get hit, as this is part of susceptibility whereas vulnerability deals with the effects of a hit.

[9]In this thesis a compartment is defined as a zone-deck. A compartment can consist of several spaces.

Based on the warhead size the hit space (smallest damage possible) or the hit compartment and directly adjacent spaces and compartments are considered as damaged, and all components and connections in these spaces and compartments are affected. The second and more detailed approach is using a damage volume. The damage volume shows, given a point of impact, what spaces are affected. The damage volume can be determined using a generic volume given a warhead size and structural ship characteristics [Gates, 1987] called a damage ellipsoid. Another method is to use a more physics-based approach to establish the volume as the method developed in [Stark, 2016].

Combining the weapon characteristics defines a damage extent and location, and the total damage to the ships system can be determined given the number of hits.

2.5.2. Operations

The type of operations conducted is depending on the environment (which threat exists) and therefore the tactics to be employed. During literature study, the author noticed employment of naval tactics was never mentioned as a factor which should be considered[10]. It can however be debated that the employment of naval tactics in case of an attack might influence the vulnerability of a ship. This is best illustrated for the case of an air attack. In this case one of the actions taken on board of the ship is to start manoeuvring for several reasons, e.g. for minimizing ship signature and maximizing the employment of weapons and decoys (i.e. minimizing the susceptibility of the ship). This is where susceptibility and vulnerability overlap. Depending on the tactic being employed, this could mean that certain compartments are more susceptible to hits than others, which once again should be reflected by the hit probabilities of the compartments. When for example the tactic is that the visual aspect needs to be minimized, the ship will manoeuvre in such a way that the threat will approach the ship from the bow or astern. This means compartments abeam should be assigned a smaller hit probability. It is still possible that these compartments get hit though, as the tactics might fail (e.g. when the ship is unable to manoeuvre according to tactics in time due to manoeuvring restrictions, or unawareness of the threat). An example of the resulting hit probability distribution for this tactic is shown in Figure 2.9.

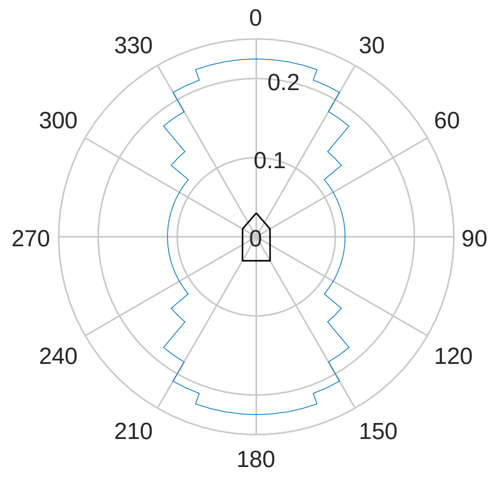


Figure 2.9: Possible hit probability distribution resulting from use of tactics

However, the influence of naval tactics on vulnerability will not be taken into account for several reasons. First of all naval tactics have a confidential nature, meaning the statements and calculations being made would not be publicly verifiable. Second, it would require operational modelling to accurately quantify the influence of the tactics on vulnerability which is not the objective of this research. Third, employment of tactics would commence only once the ship is aware of the threat. This would also be an argument for not taking naval tactics into account, as once again operational modelling is required

[10]This doesn't mean that tactics are not taken into account at all, especially during high-fidelity simulations in the detailed design stage; it simply means the author isn't aware of any consideration of naval tactics with regard to vulnerability in early stage ship design

to take the performance of sensors and human behaviour into account. Finally, and maybe the most important reason, one could argue whether the designer should actually take tactics into account at all, or whether tactics should be more tailor-made for certain ship designs, i.e. take a given ship design with its vulnerabilities and evaluate the consequences for naval tactics.

2.5.3. Effectiveness

This subsection will discuss the operational output of a ship design. The initial operational output of a design is dictated by the design itself: the combination of effectors and supporting systems provide capabilities. These capabilities are necessary to perform tasks, which are in turn needed to be able to perform a certain mission as shown in [Brown, 2013], Figure 2.10.

The reason that the operational output is discussed in this thesis is that, in the end, the operational output should be optimized with regards to other properties of a design and the given boundaries (e.g. costs). This makes it a relevant steering factor when it comes to assessing and optimizing designs. Operational output is discussed in this section regarding the operational context because the output is a combination of the system capabilities, the environment and the operations as shown in Figure 2.7 [Brown, 2013]. When it comes to vulnerability, the output of a design is increasingly influenced by the environment. The overlap between the ship design and the operational environment increases when the ship is damaged (environmental aspect), as this will influence the initial system capabilities (ship design aspect).

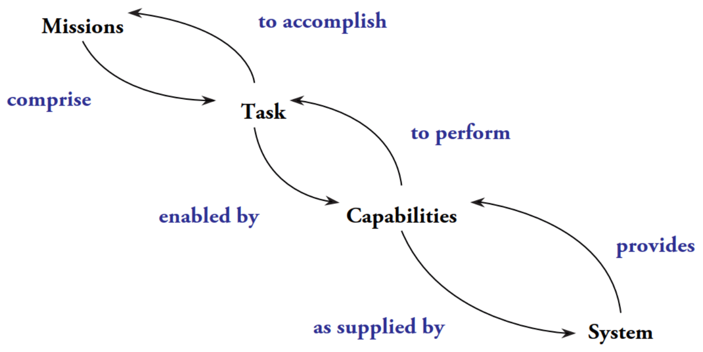


Figure 2.10: Relating mission to systems, from [Brown, 2013]

There are several ways and levels in which the output of a ship can be quantified. The system engineering V-model as shown in Figure 2.11 can be used to illustrate the different levels of assessing output as defined by [Brown, 2018]. Given a physical solution (i.e. a design) the performance of certain systems can be quantified, independent of the environment, by *Measure of Performances (MOPs)*. Currently this system level is used at DMO when assessing the vulnerability of designs (i.e. which systems are available given a hit). These MOP can be grouped together and with the appropriate weighing factors form *Measure of Effectiveness (MOE)* for each mission. Similarly all MOE can be combined into an *Overall Measure of Effectiveness (OMOE)* to quantify the fitness of a design to accomplish all set requirements. The OMOE might include the vulnerability of a design as a separate factor as illustrated in [Goodfriend, 2015], where the effectiveness in a damaged state is quantified by an *Overall Measure of Vulnerability (OMOV)*. The following subsections describe methods to (1) obtain values for these metrics and (2) weighting factors.

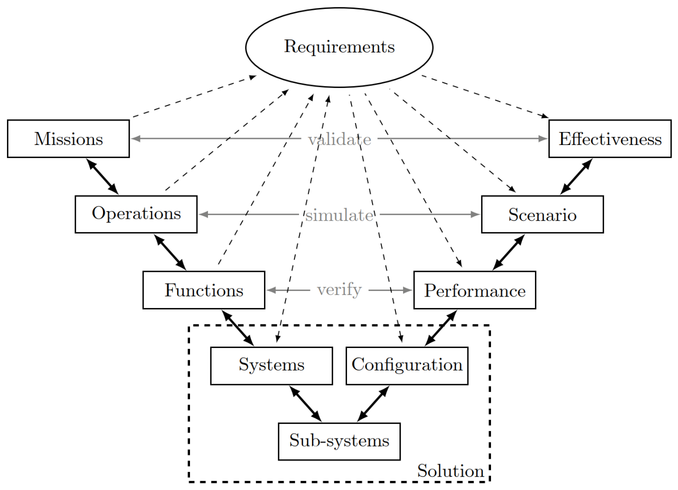


Figure 2.11: Systems engineering V-model, from [Duchateau, 2016]

Expert-opinion based

Traditionally values for weighing factors and to a certain extent scores for the performance metrics are obtained by expert-opinion [Brown and Kerns, 2010], [Brown, 2013]. Using pairwise comparison and the Analytical Hierarchy Process (AHP), values for a range of variables can be obtained. This method uses experts who manually assign a score to 2 competing variables. This can for example be scoring the relative importance of the different warfare areas of Anti-Air Warfare (AAW), Anti-Submarine Warfare (ASW) and Anti-Surface Warfare (ASuW). With scoring the relative importance of AAW vs. ASW, AAW vs. ASuW and ASW vs. ASuW, the relative weight of the areas (summing to 1) can be calculated using AHP. This can be done for a complete hierarchy. The actual performance score for each area can be expert based as well.

An advantage of this method is that it is simple and as such easy to understand. A disadvantage is that the awarded scores are subjective, as they are based on human judgment. The subjectivity can be diminished by having multiple experts assessing the performance and averaging those scores. A consequence of the subjectivity and the experience that experts use to base their assessment on, is that performance scores for novel concepts are harder to determine as experience with these concepts is lacking.

Design reference mission

In order to make the assessment of performance more objective, an alternative, more physics-based way of determining the OMOE was developed in [Brown, 2013]. This approach uses a Design Reference Mission (DRM) which is a detailed description of the intended mission for the design. These DRMs are build from Operational Situations (OpSits), which can be seen as missions the vessel might perform. In [Kerns et al., 2011] an example of an Ocean Going Patrol vessel (OPV) is given, which for example define drugs interdiction as an OpSit. These OpSits are made up of a sequence of smaller tasks called Naval Mission Essential Tasks (NMETs), which in turn are drawn from an Universal Naval Task List (UNTL) (Figure 2.12).

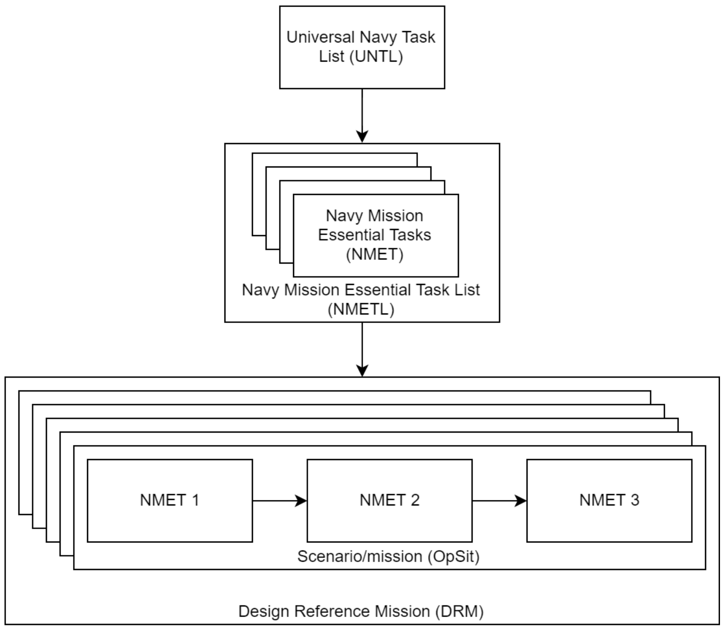
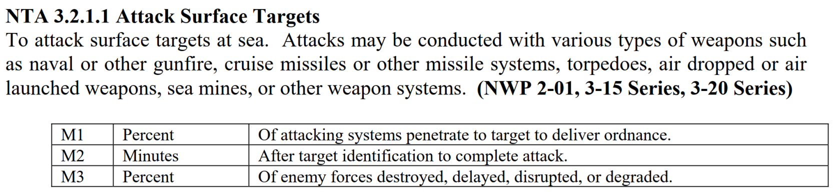

Universal Navy Task List (UNTL)
Navy Mission Essential Tasks (NMET)
Navy Mission Essential Task List (NMETL)
NMET 1
NMET 2
NMET 3
Scenario/mission (OpSit)
Design Reference Mission (DRM)

Figure 2.12: Composition of a DRM

The UNTL contain all naval tasks with a detailed description and variables on which the performance of these tasks can be based (Figure 2.13).



NTA 3.2.1.1 Attack Surface Targets
To attack surface targets at sea. Attacks may be conducted with various types of weapons such as naval or other gunfire, cruise missiles or other missile systems, torpedoes, air dropped or air launched weapons, sea mines, or other weapon systems. **(NWP 2-01, 3-15 Series, 3-20 Series)**

M1	Percent	Of attacking systems penetrate to target to deliver ordnance.
M2	Minutes	After target identification to complete attack.
M3	Percent	Of enemy forces destroyed, delayed, disrupted, or degraded.

Figure 2.13: A task from the UNTL with the indicated MOPs, from [US DoD, 2007]

The OpSit, combined with *"detailed characterizations of the threat, background traffic, weather and other factors required to assess system and overall platform effectiveness"* [Brown, 2013] are used to determine Operational Effectiveness Models (OEMs) for each OpSit using simulation software and wargaming. Combining the OEMs will provide an OMOE for a specific design.

This method has some advantages. First, it provides a detailed, more physics-based approach to quantifying the operational output of a design. It should be noted that expert-opinion is often still used to assign weights to reflect the importance of the different missions. Second, the method provides the ability to track how scores are established, meaning that, besides the scores, insight can be generated as well in what is needed to improve the scores. A disadvantage is the level of detail that is required in defining the DRM. Not all required level of detail in the tasks to be achieved may be available in the early design stage, as part of this stage is the establishment of detailed requirements. The same applies to the ability to conduct detailed simulations and wargaming.

Intermediate forms

The third method to define operational output is an intermediate form in between the previously discussed methods. In this form the relative importance of the different outputs (for example warfare areas) is determined using expert-opinion and the AHP. The scores are then determined using relative simple performance prediction tools, meaning the modelling effort and level of detail is less than required for the full simulations and wargaming. These values are then combined into a single OMOE. This intermediate form is described in [Brown, 2018] and was used in the context of assessing vulnerability in [Goodfriend, 2015].

Using this method it is possible to evaluate performance on a higher level than only on the system level, but without the need for more detailed requirements and a more detailed design. Instead of looking at system availability, the effect on operational capabilities can be evaluated. This enables the designer to get a more holistic view of the effect of design decisions in an early design stage.

2.5.4. Conclusions

The purpose of this section was to establish the influence of the operational context in relation to vulnerability of a ship design. The following relations were identified:

- The main factor of the operational environment with an influence on vulnerability is the threat. The interaction of the threat with the design can be modelled through the establishment of hit probabilities, number of hits and the use of a damage volume with different levels of detail.
- Tactics used during operations may influence the vulnerability characteristics of a design. This interaction is not considered in this thesis, as thorough research would be required to establish the exact influence on vulnerability, which lies outside the scope of this thesis.

The combination of the design, environment and operations are determining the effectiveness of a design. This operational metric can be used to drive the design process. Several methods were identified and described to quantify the effectiveness. The methods range from expert-opinion to assess the effectiveness of a design without a clearly required minimum level of detail to a method determining effectiveness through detailed simulation requiring a higher level of detail. An intermediate form using expert-opinion to establish relative importance of capabilities and simple performance prediction tools was described as well. The latter seems to have the best connection with early stage design and the current vulnerability assessment methods. This is due to the capability to deal with a low level of detail, while providing additional insight in the impact of design decisions on the effectiveness of a design.

2.6. Vulnerability reduction measures

The previous sections were related to the first part of the research objective, examining the relations between distributed system and ship design, vulnerability assessment methods and the operational environment. This section is related to the second part of the research objective, being the validation or refinement of the design rules, best practices, rules of thumb etc. which are used in early stage ship design to reduce vulnerability and the basis/assumptions of these rules. For readability this collection of measures will be referred to by the term design rules in the remainder of this thesis.

2.6.1. Design rules

There are numerous design rules for vulnerability reduction as illustrated by [Said, 1995]. He states that at time of publication Naval Sea Systems Command (NAVSEA) identified over 100 design rules based on (amongst others) analysis of existing design, accidents and actual sustained battle damage. Examples of design rules taken from [Said, 1995, Table 2] are:

1. *Redundancy and Separation*
   - *Separate vital redundant elements by at least _ feet longitudinally.*
   - *Require separated and redundant cable paths (armour where needed).*
   - *Provide separated, redundant sources, and distributive systems for vital combat system support.*
2. *Enclaving*
   - *Incorporate enclaving of capabilities in each warfare area, as feasible.*
3. *Location*
   - *Locate sensor/weapon control and equipment rooms near associated sensor/weapon, and in protected area.*

- *Locate fire pumps toward forward and after ends of ship at least one pump in each fire zone.*

4. *Back-up and Local Control*

   - *Utilize distributed processing, distributed systems and local control to improve ship survivability.*

In [Piperakis, 2013], Piperakis conducted a literature survey with numerous sources and found several vulnerability reduction measures, amongst which the principles of redundancy, separation and enclaving for systems. Combining all these principles and rules would generate a long list which would be hard to assess on validity in the testcase, as time for this research is limited. Therefore the design rules will be limited to the most important rules in use at DMO. As these design rules leave room for interpretation, they will be supplemented with generally accepted practices from other sources when necessary.

The design rules in use by DMO have two important sources. In [Streppel, 2004], explicit design rules were formulated for distributed system design in an early stage of the design process. In the research a quantification model for residual capacities of a system after damage was developed and used for redesign of existing ship systems. The assumptions during redesign were rephrased as design rules. The model was based on a damage extent of a missile hit, with one hit only. One of the remarks in the conclusion of the research was that the scalability to more hits was yet to be proven. Also the model from which the rules were derived was aimed at the design of the distributed systems, apart from the ship design process. Therefore the rules were later refined for use within DMO, which resulted in 10 rules as published in [Spruit et al., 2009]. These rules with the rationale as presented in the paper are as follows:

1. *Avoid central distribution systems.* This is a result of the dependency of effectors on supporting networks: even if the effector is not damaged by a hit, the unavailability of a supporting network can result in unavailability of the effector (see also [Goodrum et al., 2018]). Therefore central distribution systems should be avoided when possible. This means that the necessary resources for a user are best generated locally instead of centrally and then distributed to the users.

2. *Separate redundant sources (pumps, generators, chilled water plants, etc.).* Purpose of this design rule is to avoid unavailability of redundant sources as the consequence of a single hit. The sources should be separated preferably in all dimensions (in height, longitudinally and transversally).

3. *Apply protection if sources must be in each other's vicinity.* When physical separation as dictated by the previous design rule is not possible, the sources should at least be separated in the sense of damage propagation.

4. *Separate redundant paths.* The same rationale as for the separation of redundant sources apply.

5. *Apply protection where redundant paths have to be close together.* The same rationale as for the protection of redundant sources apply.

6. *Arrange feed and return lines for closed loop systems next to each other.* As both lines are needed to operate a closed loop system, separating them would only increase the probability that the system is affected in case of a hit.

7. *Avoid Single Point of Failures (SPFs).* This is can be seen as a generalisation of the first design rule and has the same rationale: avoid that a single hit can affect the availability of the complete system by limiting the damage envelope.

8. *Implement a cross-over near the system's essential users.* This will provide the largest flexibility in dealing with damage, as it will create disjoint paths with the largest degree of separation. An exception to this rule can be the case when the hit probability is non-uniform. In that case the position for the cross-over should be determined keeping the threat against which the system needs to be protected in mind.

9. *Implement a cross-over near the system's sources.* According to [Spruit et al., 2009] this is the second best place for a cross-over. It can however be debated whether this is the second best place for a cross-over of a redundant path, or the best place for a second cross-over of a redundant path as stated by [Streppel, 2004]. The latter statement makes sense, as this increases the possibilities for reconfiguration and thereby reduces the vulnerability. It is however not clear why this would be the second best place if only one cross-over is present. After all, in this case a hit in the supply line to a user will always be fatal for the user.

10. *Combine paths of distribution systems for essential capabilities.* If a system require several types of resources to function (for example electricity and chilled water), the paths of these resources towards the end-user should be the same. As for rule 6, the separation of these lines would only increase the probability that the end-user is unavailable in case of one hit.

These rules are also stated in [van Meurs, 2019], and an additional rule is introduced which states that in general ring-shaped systems are a preferred solution. Although not specifically mentioned, the separation distance is usually dictated by the weapon effect radius [Belcher, 2008, as cited in [Piperakis, 2013]] .

2.6.2. Conclusions

There are 10 golden rules / best practices for distributed system design in ship design in use at DMO, which can be supplemented with an additional rule regarding ring-topologies. These rules are covering all aspects of distributed system design for ships as discussed in Section 2.2.2. These best practices are applied by other navies as well [Piperakis, 2013].

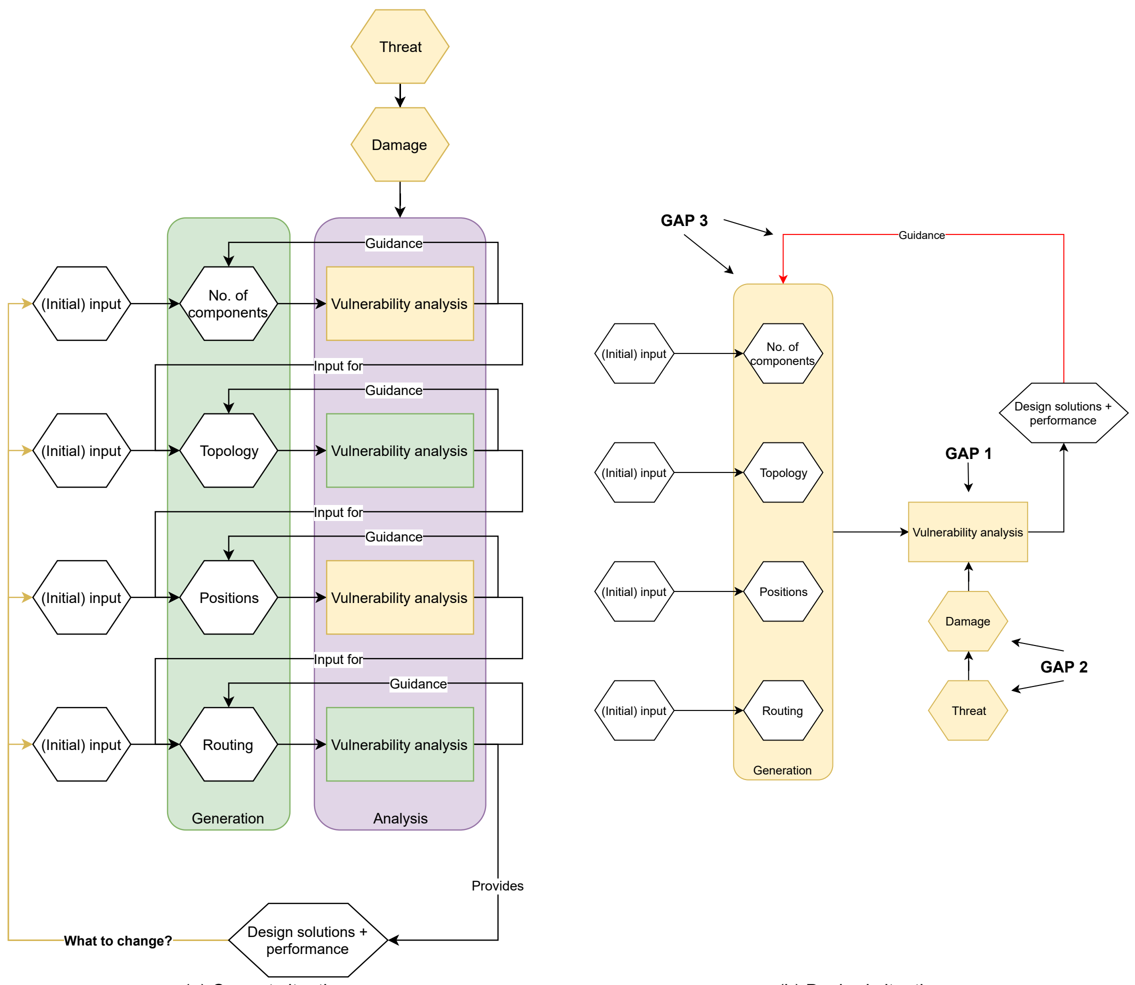
The main assumption for these rules is that they are based on a single hit with a relative large damage extent (as the damage is caused by a missile). This is particularly interesting in light of recent operations of naval ships (which are in the lower end of the spectrum of force, such as anti-piracy) and the survivability requirements set by the residual capability matrix (which requires less capabilities for larger damage extents).

2.7. Gap analysis

Based on the research objective and the literature research, a gap can be identified. The first part of the research objective was to see how vulnerability analyses can be used to guide the early stage design effort in order to increase survivability, which was further scoped to vulnerability reduction for distributed systems. In other words, how can vulnerability analysis be integrated in the early stage design process so that less vulnerable design solutions are produced. Using this, the second part of the research question (determining the influence on design rules) can be answered.

The literature study examined the factors influencing the vulnerability of a design solution (Section 2.2), when and how these factors are determined (Sections 2.2 and 2.3), how the vulnerability of a design can be assessed (Section 2.4) and the interaction with the operational environment (Section 2.5).

Figure 2.14 schematically depicts the current situation and the desired situation.



(a) Current situation (b) Desired situation

Figure 2.14: Gap analysis

The color of the shapes in Figure 2.14 is a result of the literature review in relation to the research objective. The shapes in green indicate subjects for which theory and implementation at DMO is available, yellow indicate available theories which need adaptation or implementation at DMO and red indicated subjects for which no theory is available or insufficient at this stage. The choice for which theories or methods will be used in this research will be based on the requirements for the integrated model, which will be described in the next chapter. Figure 2.14b shows two areas in which a gap exists between the current and desired situation: the vulnerability analysis, and the guidance and generation of designs.

The first part of the gap (GAP 1) is related to the vulnerability analysis. The research objective contains the phrase *"...is to decrease the vulnerability..."*. From the definition in Section 2.1, this means that the degradation of operational performance following damage needs to be minimized, i.e. operational capabilities needs to be maximized. From the literature study, there is a range of analysis methods available to conduct the analysis. The results of the considered vulnerability assessment methods should be a measure of operational performance or capabilities, in such a way that it can be used to guide the generation of designs. This means that the design process is being fed with an input which is able to steer the process in the right direction. As indicated in the schematic, theory for determining operational performance at different levels exists as discussed in Section 2.5.3 and is deemed sufficient.

The second part of the gap (GAP 2) is the way in which damage is simulated. Varying theories to simulate different types of damage and extents as discussed in 2.5.1. At DMO only single or multi compartment damages are considered in the early design stage. As indicated in the problem description, smaller damages might have different effects on the overall vulnerability of a design. Although detailed weapon modelling is not the purpose of this thesis, more differentiation in the damage types considered is necessary for evaluation of the design rules.

The third part of the gap (GAP 3) is the generation of better (less vulnerable) designs using the output of the vulnerability assessment. Bridging this gap will be the emphasis for the research, as little research has been found on this part of the problem. This gap consists of various aspects:

- *Which design variables to vary in order to decrease vulnerability*. Five design variables were identified earlier in Section 2.2.2 which determine the vulnerability of a distributed system. In the current situation, these design variables are more or less determined and optimized separately, with input of the previous processes as also shown in the integration tree in Figure 2.6. Recent research give examples where vulnerability is taken into account as (one of the) considered variables for optimization. Goodfriend uses a GA where vulnerability is part of the objective function for finding optimal system configurations with a fixed topology in [Goodfriend, 2015], de Vos uses a GA with a reconfigurability objective in his ATG to find optimal topologies given a system configuration in [de Vos, 2018] and Duchateau uses a GA with a post-hit availability of systems objective to find optimal paths given a system configuration, positions and topology. This means that local optimization occurs, where the optimal solutions are bounded by the choice of the design variables which are taken as input. The goal however should be to find global optimal solutions, meaning least vulnerable ship designs.

  In order to find the best overall design, the combination of design variables should be optimized using the vulnerability assessment methods. This is gap, as previous research indicated above only take one variable into account.

- *Up to when to conduct/allow the change of certain design variables*. Even if it is known which design variables should be changed, the question is whether this is known in time. In this context this refers to the practical computational possibilities that exist. It is for example undesirable to make changes in the number of components in a late stage as this requires modifications of the topology and paths as well. Some variations might not even be possible without changing ship design variables: if for example the additional components are big (e.g. additional generators), the available space might be insufficient and the hull form needs to be changed. So far this problem has not been specifically addressed in research, as this is only a problem when variations are conducted using multiple design variables which are determined in different stages of the design process.

- *To what extent are variations allowed*. As said in the previous paragraph, a distinction might be possible within design variables. A change in components not always has the same effect. The previous item described the effect of adding big components. The addition of small components however (e.g. a LC) might be possible within the existing ship design, or only have limited effects.

Summarizing, the main gap exists of 3 parts: inclusion of operational capabilities as an output for vulnerability assessment methods, the way damage is modelled and a methodology to create less vulnerable ship designs using vulnerability methods in which variations of all distributed ship system design variables are considered holistically. The next chapter will discuss what requirements can be deduced for the new model to generate better designs, which discussed existing theories can help bridging the gap and what methodology will be used in the model.

3 Solution generation

Purpose of this chapter is to outline the direction for the solution of the problem. First the requirements of the desired model will be discussed, which is a result of the research objective and the identified gap. Thereafter the contribution of the different theories found in the literature review will be considered with regards to the requirements. Finally the outline of the model which will be developed is presented, including the necessary boundaries and simplifications.

3.1. Requirements

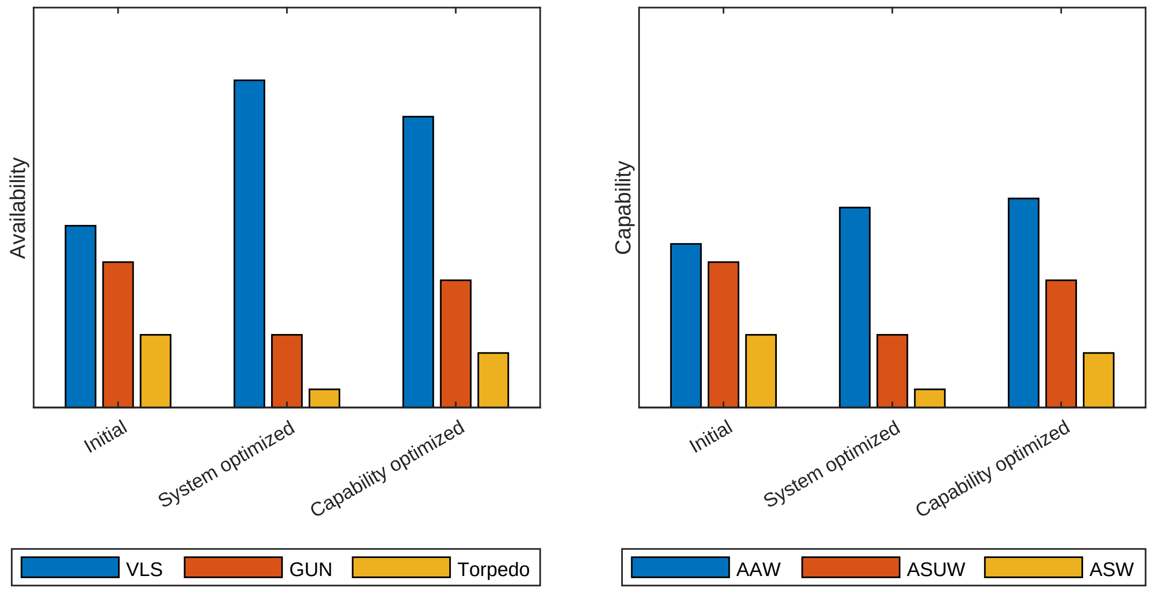
From the research objective and gap analysis, several requirements for the model can be identified (part of which has already been discussed in Chapter 2, Section 2.7). The objective is (1) to improve designs through the use of vulnerability analysis, resulting in an increase of operational capabilities in damage situations and (2) to be able to evaluate the design rules for different types of damages. The following requirements are formulated based on this objective and earlier formulated desired situation.

RM01 *Operationally driven (GAP 2 & 3)*. The design driver used to steer the design process shall be operationally oriented. This is required to evaluate the influence of design decisions on the performance of the complete design instead of only one system. Figure 3.1 shows the difference. The figure shows an example with arbitrary chosen values for system availability, with calculated resulting operational capabilities. In this example the AAW capability is provided by both the VLS and gun, whereas ASuW is only depending on gun availability and ASW on torpedo availability. When one is optimizing system availability and a better AAW capability is preferred, the availability of the Vertical Launch System (VLS) would be the logical system to optimize as shown in Figure 3.1a. The example shows however that to increase AAW capability as depicted in Figure 3.1b, a slightly less VLS availability but higher gun availability would be preferred. This shows that optimization of capabilities is not the same as maximizing the availability of separate systems.

RM02 *Requirements (GAP 2)*. The model shall be able to incorporate the operational requirements as set by the designer. This means that the model is able to steer the process based on (1) different operational capabilities (e.g. AAW, ASW, mobility, etc.) and (2) different levels of this capabilities (e.g. for AAW short-range AAW, long-range AAW) as defined by a RCM for different types of impact.

RM03 *Integration*. The model should be able to integrate with existing relevant tools used for distributed system design and ship design. This makes it easier to implement the method in existing workflows and makes maximum use of proven tools and research.

RM04 *Varying multiple Design Variables (DVs) (GAP 3)*. The model shall be able to influence the design by altering multiple DV. The reason for this requirement is the dependency between DVs as discussed in Section 2.7.



(a) Resulting system availability. (b) Resulting operational capabilities.

Figure 3.1: Example of the difference between optimizing a configuration for maximum system availability or maximum operational capabilities.

RM05 *Different damage cases (GAP 2)*. The model shall be able to handle different damage cases (number of hits and extent of damage). The reason for this requirement is twofold. First, it was noticed in the problem definition that warships were employed the last years with a more diffuse threat. This means that during the design process not only the conventional missile threat should be considered, but also threats with a smaller damage extent. Related to this is the second reason, which is the desire to evaluate the design rules for these smaller damage cases.

RM06 *Early design stage*. The model should be suitable for the early design stage. This means the method should be sufficiently fast and able to deal with the level of detail used in the early design stage. The speed requirement is hard to quantify. In this research the requirement is that the speed should not limit the designer in using the method. For the level of detail this means that several interdependent networks can be considered, while the level of spatial detail (e.g. detailed arrangements within compartments) is limited.

3.2. Theories

In the literature review three parts of the model were identified where theory already exists, being vulnerability analysis, the quantification of operational capabilities contributing to an operationally oriented metric for guiding the design, and threat modelling to simulate different types of damage. This section will examine which theories will be used in the model, based on the requirements formulated above (Figure 3.2). Using these theories will help bridging the first and second part of the identified gap (inclusion of operational capabilities as a output for vulnerability assessment methods and damage simulation).

3.2.1. Vulnerability analysis

As dictated by RM03, existing tools or methods should be integrated when possible in the model. Vulnerability analysis is required for assessment of the performance of a ship in case of damage. From the model requirements several requirements specific for the vulnerability analysis method can be derived. Besides these specific requirements, the selected analysis method should not violate the previously formulated overarching model requirements (e.g. calculation time, level of detail, etc.). The specific requirements are:

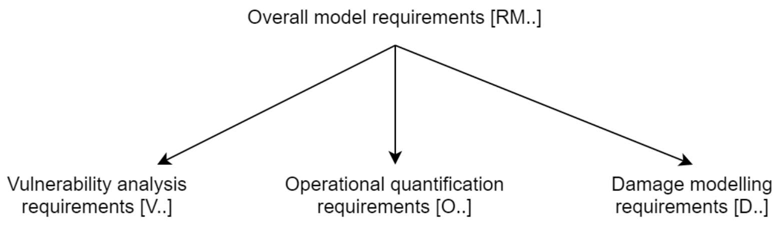

Overall model requirements [RM..]
Vulnerability analysis requirements [V..]
Operational quantification requirements [O..]
Damage modelling requirements [D..]

Figure 3.2: Relation of model requirements

V01 *Assessment level*. The method should be able to assess the vulnerability of a complete design, taking into account all DVs.

V02 *Guidance*. The vulnerability assessment method should be able to help guiding the design effort as much as possible. This might help determining which DV should be altered or how the DVs needs to be altered. This requirement is related to the model requirements RM04 and RM06.

V03 *Complexity of networks*. The vulnerability assessment method should be able to process networks typical for the early design stage. In [Duchateau et al., 2018] a distributed systems network consisting of 23 nodes was considered. In [de Vos and Stapersma, 2018], de Vos introduces a realistic benchmark system for his ATG research which contains 36 nodes, of which 2 sensors and 1 weapon. The supporting networks (2 times chilled water, 2 times electricity and data) are at a level of detail one would expect in the early design stage, while the number of effectors is typically larger. This requirement is related to RM06.

V04 *Damage cases*. The method shall be able to deal with damages of different extent and multiple hits. This requirement is related to RM05.

V05 *Flexibility*. The vulnerability assessment method should be flexible. This means that the method is not only capable of assessing the vulnerability of a complete design (V01), but preferably also able to conduct assessment in different stages of the design process (that is, with different input).

The vulnerability assessment methods should be able to fulfil these requirements directly or, if this is not possible, with limited modifications. Table 3.1 shows to what extent the vulnerability methods comply with the requirements.

	Max-flow-between-hubs	*Markov*	*Hurt-state-percolation*
Assessment level	Topology	Design	Design
Guidance available	No	Based on eigenvector analysis	Identification of weakest link possible
Complexity of networks[12]	100s nodes/edges	10s nodes/edges	100s nodes/edges
Damage cases - number of hits	Yes	Large	Limited
Damage cases - extent	No	Yes (depending on spatial resolution of the input)	Yes
Flexibility	No	No	Yes

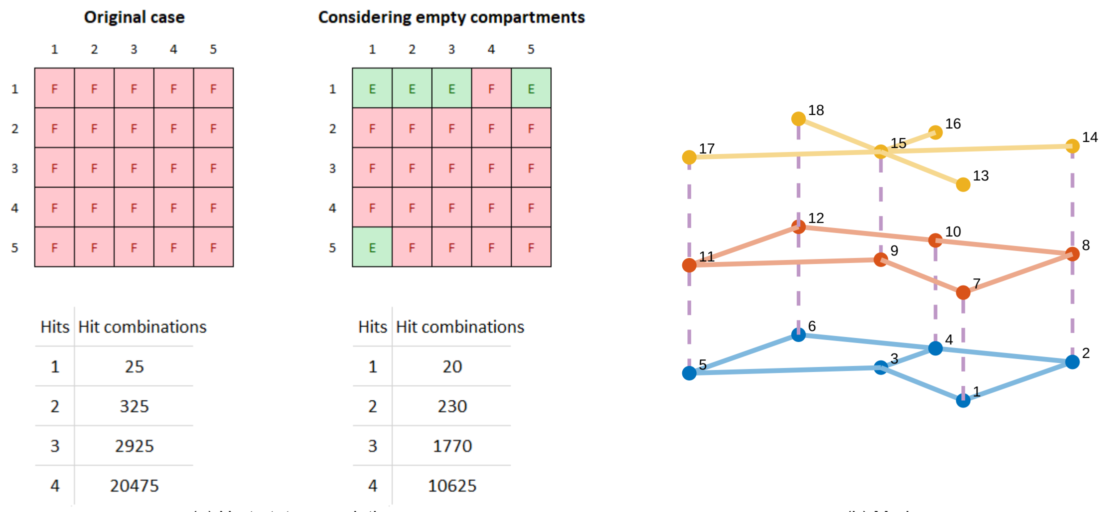
Table 3.1: Vulnerability assessment methods

[12]This is a generalization of the numbers as presented in [Habben Jansen et al., 2019a], stating max-flow-between hubs need at least 36 nodes/edges, the Markov-method a maximum of 16 nodes/edges and hurt-state-percolation 20-100 nodes/edges. Networks with up to 300 nodes have already been analysed using hurt-state-percolation.

Based on the requirements and literature survey a vulnerability assessment method can be selected. First of all, the max-flow-between-hubs method is not suitable for vulnerability assessment in the integrated method as it is unable to take the physical architecture into account. This is easy to understand, as the method was developed for design of topologies of distributed systems and not for the assessment of ships.

The compliance of the Markov and hurt-state-percolation methods with the requirements are much closer together. In the end the hurt-state-percolation method is deemed better suitable to conduct the vulnerability assessment for a variety of reasons.

The first and most important consideration is based on the computational possibilities and limitations. Hurt-state-percolation is able to handle complex networks with numerous nodes, whereas the Markov method is limited in that respect at this stage. On the other hand, the method is limited with regard to handling multiple simultaneous damages, whereas this is no problem for the Markov method. It is however deemed more important that the method is able to handle complex network than damage cases with more than 2 hits in the early design stage. Besides this, computational improvement of the methods within this research to diminish the limitations are assessed to be better possible for hurt-state-percolation than for the Markov-method. The limitation for hurt-state-percolation is based on the number of hit possibilities which needs to be evaluated. This could for example be improved by limiting the number of hit combinations that needs to be evaluated (i.e. when a compartment is hit and empty, this hit doesn't need to be evaluated, saving computing time, Figure 3.3a)[13]. The limitation for the Markov-method is based on the maximum size of the transition matrix in Matlab®. To improve this and get the same network complexity as possible with hurt-state-percolation, networks could be evaluated separately, after which the results should be combined to evaluate the total vulnerability of a design (Figure 3.3b). Without looking further in detail, this adjustment is assessed much more complex than the adjustment for the hurt-state-percolation method as this would require multiple iterations per design to incorporate the effect of cascading failures in all networks.



Original case

	1	2	3	4	5
1	F	F	F	F	F
2	F	F	F	F	F
3	F	F	F	F	F
4	F	F	F	F	F
5	F	F	F	F	F

Hits	Hit combinations
1	25
2	325
3	2925
4	20475

Considering empty compartments

	1	2	3	4	5
1	E	E	E	F	E
2	F	F	F	F	F
3	F	F	F	F	F
4	F	F	F	F	F
5	E	F	F	F	F

Hits	Hit combinations
1	20
2	230
3	1770
4	10625

(a) Hurt-state-percolation (b) Markov

Figure 3.3: Visualisation of suggested computational improvements

A second possible advantage of hurt-state-percolation over Markov is that it is able to give rather specific direction what factor is causing the unavailability of a system. The Markov method can provide guidance on how certain vulnerabilities can be decreased as well, but this requires analysis of the explicit formulations of each state. The ability to automatically use these insights to create better designs

[13]Looking for example at the damage case presented in [Duchateau et al., 2018] with 2 simultaneous hits, 18 of the 45 compartments were 'empty' (40%). Therefore the number of hit combinations to be evaluated reduce by 60% for 2 hits, 75% for 3 hits and 84% for 4 hits just by optimizing the calculation method. With more realistic configurations the percentage 'empty' compartments will decrease

might therefore be more complex than for hurt-state-percolation.

The last advantage of the hurt-state-percolation over the Markov method is that the method can be used to examine topologies without spatial information as well. This can provide information regarding the sensitivity of a topology to loss of a particular component or connection. This is not possible with the Markov method, as spatial information is required to set-up the transition matrix. The possible advantage of having one method suitable for various stages in the design process is that it might be able to re-use certain parts of the code. This lightens the programming load and enables the research to focus on correct integration of the tools in the design process instead of a focus on programming.

3.2.2. Quantifying operational output

As dictated by the first model requirement RM01, the output of the vulnerability assessment method should be translated to a metric quantifying operational capabilities or effectiveness. Given the model requirements, the following requirements regarding the translation can be formulated:

O01 *Vulnerability input*. The operational output shall be determined based on the output of the vulnerability assessment method in order to determine the operational output in a damage case as accurate and objective as possible.

O02 *Compatibility requirements*. The converted output shall be compatible with the formulated requirements (i.e. RCM) to allow determination of compliance with the minimum requirements. This requirement is related to RM02.

O03 *Level of detail*. The method to determine the output shall be able to deal with the relative low level of detail available in the early design stage. This requirement is related to RM02.

O04 *Complexity of calculations*. The computational load should be as low as possible. The method will be used to convert the output of the vulnerability assessment method into an operationally relevant score in order to be able to drive the design process. As such, the conversion is not a separate objective of the design method. This requirement is related to RM06.

In the literature review 3 possible methods to quantify the operational output were identified: the expert-opinion based method, DRM-method and intermediate form. Table 3.2 show to what extent the different methods comply with the requirements.

	Expert-opinion	*Intermediate*	*Design Reference Mission*
Vulnerability input	No	Yes	Yes
Compatibility requirements	Yes	Yes	Yes
Level of detail	Low	Low	High level of detail required
Complexity of calculations	Low	Low	High level of detail required

Table 3.2: Quantification method operational output

From the first requirement it is immediately clear that the method using exclusively expert-opinion to determine both the importance of different aspects and their score is not sufficient for determining the operational output, as this method doesn’t take the output of the vulnerability analysis into account. Therefore only the intermediate form and DRM-method will be discussed further.

As discussed in Chapter 2 Section 2.3.1, the vulnerability requirements are formulated as minimum required operational capabilities (for example mobility, warfare areas, etc.) given a certain damage in the RCM. Both remaining methods are able to provide feedback at this level. Where both methods differ the most is in the area of the required level of detail and the complexity of calculations. The DRM-method requires a relative high level of detail to conduct the rather detailed simulations. Due to the high level of detail in the input, the simulations consist of complex calculations. The intermediate method requires a low level of detail as input: it is solely based on the relative importance of systems when contributing to a certain capability and the system availability after sustaining damage which results from the vulnerability assessment tool. The level of detail required as such is not depending on the quantification method for operational output, but on the required input for vulnerability assessment.

Based on these considerations the intermediate form will be used, where the performance is predicted by the vulnerability assessment method and the relative importance by the designer using the AHP with expert opinion of the operational users of the vessels in the requirement elucidation phase.

3.2.3. Damage simulation

The first part of the problem definition questions whether the design rules are still applicable in conditions where the assumptions of the original rules are violated.. A difference in the anticipated damage was mentioned as an example, as traditionally the vulnerability is based on one large hit. The effect of smaller hits is not evaluated in the early design stage at all. This is reflected by the current method of simulating damage, where complete compartments (zone-decks) are damaged by a hit. A more refined method of simulating damage enables a better evaluation of the design rules. The following requirements are formulated for the refined damage modelling:

D01 *Damage extent*. The method shall be able to simulate different damage extents, including smaller damages. This enables a more thorough evaluation of the vulnerability aspects of a design.

D02 *Level of detail*. The method shall be in line with the level of detail of the early design stage. This means the method shall be relatively simple, as the detail level of the design is not able to correctly reflect the effects of detailed weapon modelling (e.g. fragmentation).

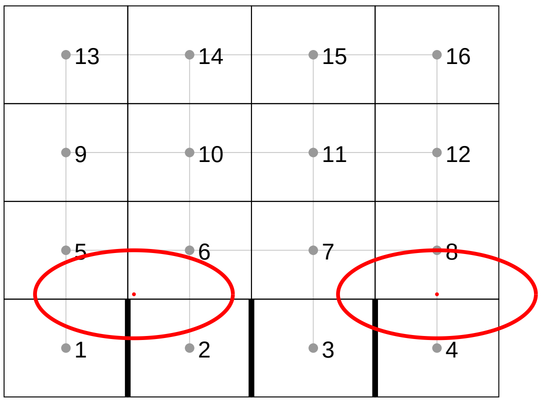
D03 *Domain*. The method shall be able to simulate threats from different domains. This means that the damage extent and locations should be a logical result from the threat.

D04 *Simplicity*. The method shall be simple to implement. As weapon modelling is not a main objective of this research, the method should be 'good enough'. That means that damage modelling should be simple, but able to achieve the objective which is evaluation of the design rules.

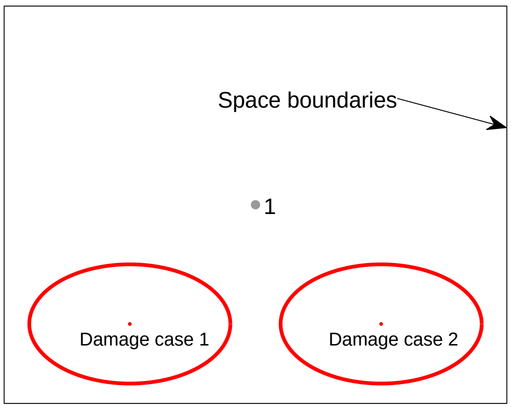
The model requirement D03 can partly be achieved by implementation of hit probabilities. As argued in Chapter 2, Section 2.5.2, the damage extent can be simulated by examining effects of discrete damage volumes (like compartments) or using a more detailed determination of the damage volume like damage ellipsoids based on a continuous hit distribution. In this research damage will be modelled using discrete damage volumes, where the resolution of the discrete volumes, normally compartments, will be increased to spaces inside the compartment. This method is chosen above the use of damage ellipsoids for several reasons.

First, implementation of a damage ellipsoid is more complex than implementation of discrete volumes. This is especially the case for modelling the protection offered by blast bulkheads. This protection would mean that the ellipsoid is unable to affect certain spaces, but this depends amongst others on the hit location and geometry of the hit location versus the space with protection. An example of this geometry is shown in Figure 3.4a. In this example spaces 1 through 4 are separated by blast bulkheads. The hit shown in space 6 has a different effect on space 1 than the hit in space 8 has on space 3. For discrete damage volumes it is possible to model the effect of blast bulkheads in a simple way, using damage adjacency matrices [Duchateau et al., 2018].

Second, the smallest damage which can be modelled is depending on the level of detail of the design. This means that when damages smaller than the resolution of the design are considered (or when spaces are partially affected as shown in Figure 3.4b), assumptions need to be made regarding the effect within the space (e.g. all components within the space are inoperable, or a certain percentage). An example where the damage extent is smaller than the space dimensions is shown in Figure 3.4b: the damage is smaller than the affected space. This means that both shown damage cases could have different effects, depending on the exact (unknown) positions of components or paths within a space. Since spaces are the lowest level of detail that is considered for the design (e.g. it is known which components and paths are in which space, but not exactly where they are within this space), the effect of the damage is unknown. Therefore the use of a damage ellipsoid does not have added value over discrete damage volumes in the context of this research, as the increased complexity requires the introduction of additional assumptions.



(a) Effect of blast bulkheads using damage ellipsoids



(b) Damage extent vs. design detail level

Figure 3.4: Lack of added value using damage ellipsoid over discrete damage volumes

3.3. Scope and design generation methodology

Figure 3.5 shows the earlier defined desired situation. In the previous subsection inclusion of operational oriented output for vulnerability assessment was addressed as well as definition of the damage extent. This section will address a solution direction bridging the third part of the gap: given the operational capabilities in a damage situation, how can this be used to generate better ship designs? Two possible methodologies to bridge this gap will be discussed.

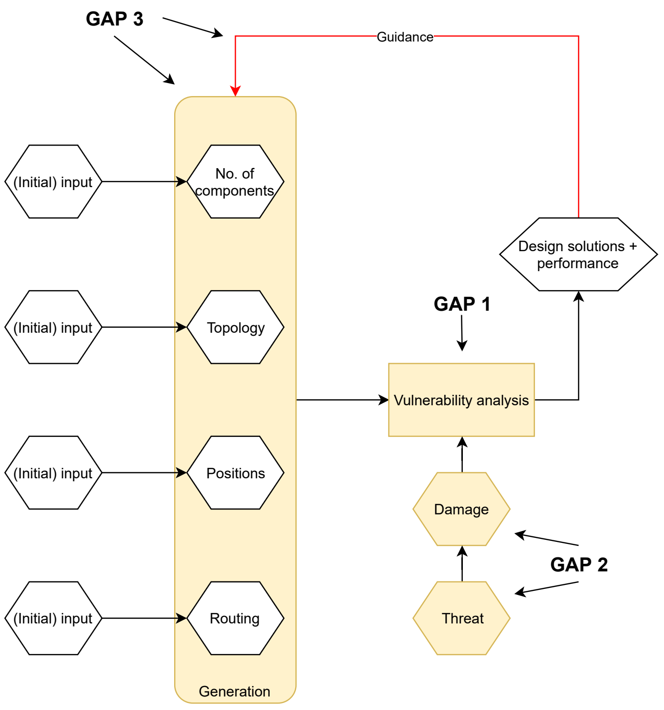


Figure 3.5: Desired situation

3.3.1. Method 1: Directed variations

The first method is based on the hypothesis that the vulnerability assessment method can not only provide a measure of the vulnerability but also identify the contributing factors to the vulnerability. This can then be used to improve those factors. If the result of the vulnerability assessment of the ship design is that a certain network (for example a Chilled Water (CW)-circuit) is the main cause for a decrease in operational capabilities, this part of the network should be improved. With hurt-state percolation it can be established why this particular circuit is a weak spot: either by failing components or failing connections. This weak spot could then be improved using existing tools to generate solutions for DVs, thereby reducing vulnerability.

The most important advantage of this method is that it directs the design process where to look for improvements. This means time and resources can be used efficiently to improve designs. The existing methods to generate solutions defining the DV can be used, as only a feedback mechanism needs to be implemented.

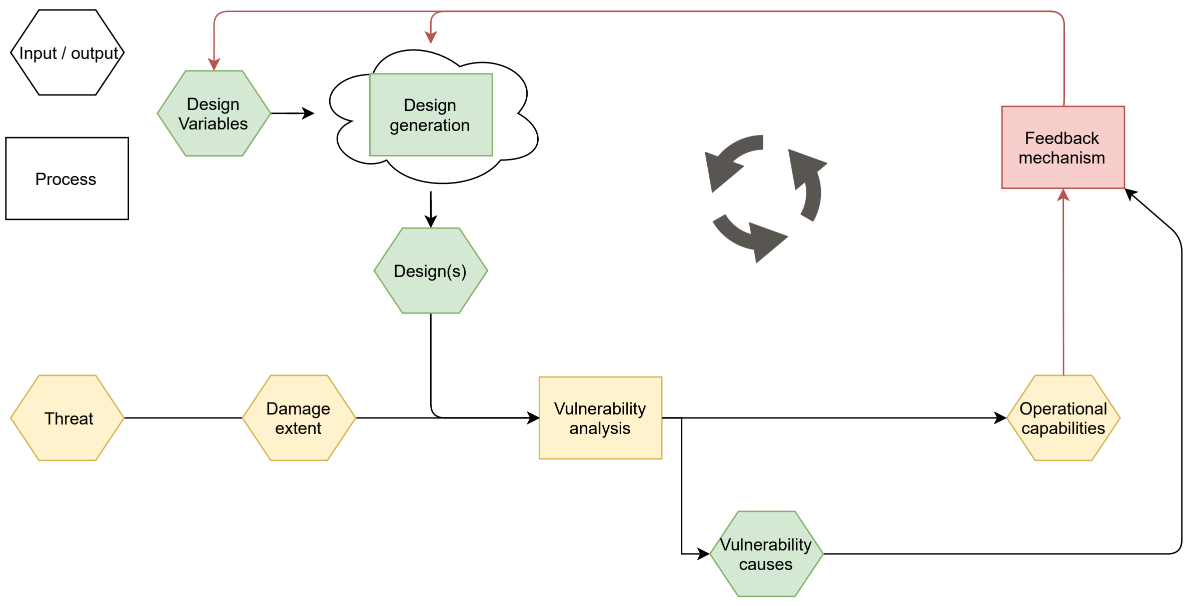


Figure 3.6: Workflow of method 1

A few difficulties for this method exists. The first difficulty is related to the specificity in which the contributing factor can be identified. The method is based on the hypothesis that the vulnerability analysis can identify the contributing factors to the vulnerability. To a certain extent this is the case: as said, the failing components, connections and as a result networks can be identified. Actually determining which DV is causing this, and as such how to solve it is more difficult to establish as shown in Figure 3.7. When for instance components are failing, should their position change (e.g. because redundant components are too close to each other), their number (e.g. because a component is a single point of failure), or could it be solved by changing the topology (e.g. create additional paths from a redundant component)? If it is not possible to identify a contributing factor, it is not possible to direct the design process in the right direction by targeted changes of variables. A possible solution for this problem is application of the design rules to determine which factor should be changed (e.g. when two redundant components are close to each other separation should be increased), but one of the purposes of this research is to evaluate the design rules.

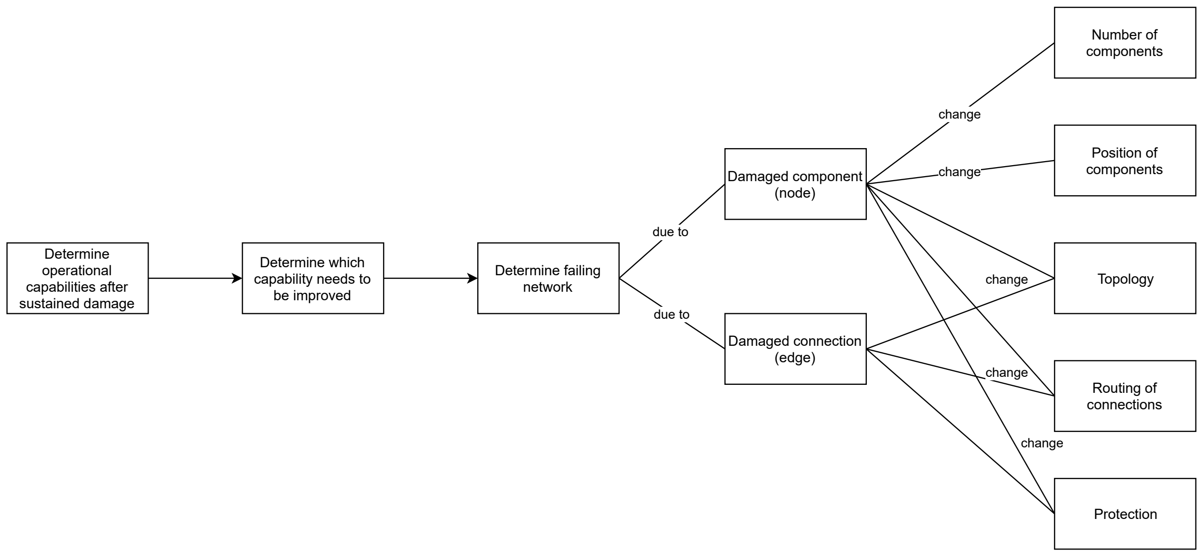

Determine operational capabilities after sustained damage
Determine which capability needs to be improved
Determine failing network
due to
due to
Damaged component (node)
Damaged connection (edge)
change
change
change
change
change
Number of components
Position of components
Topology
Routing of connections
Protection

Figure 3.7: Fault tree

The second difficulty is related to second order effects of the changes. A change in one variable likely necessitates other DVs to change as well due to the earlier discussed dependencies between the variables. The effect on the overall vulnerability is not known until after evaluation.

The third difficulty is related to complexity of the networks considered. As shown in the fault tree in Figure 3.7, a problem (for example a failing component) can have different solutions (change of topology, different routing, etc.). In that case, the best solution should be picked (e.g. the biggest improvement in vulnerability, or in case of similar performance the most cost-effective, etc.). However, the best solution can only be identified when a trade-off can be made, i.e. when the effect of different solutions is evaluated. This is negating the advantage of this method, being an efficient use of time and resources to improve the design solutions.

3.3.2. Method 2: Holistic variations

The starting point for the second method is a variation of all DVs, where the effect of changes of the variables on the overall vulnerability is evaluated. The variables are changed and optimized towards a global optimum instead of the sum of local optima. GAs are widely used for optimization of a single DV as discussed in the gap analysis. This technique could also be used in a more holistic approach as shown in Figure 3.8. In this example 3 DVs are varied, where DV 1 and DV 2 are independent of each other. Where the previously discussed method only implements a feedback mechanism and uses existing methods to create solutions for the DVs, this method will use the GA to generate solutions taking multiple DVs into account.

In the current design practice, DV 1, DV 2 and DV 3 are generally optimized sequentially and more or less in isolation. Feedback from a subsequent process (e.g. the outcome of the optimization of DV 3) is only applied at an earlier stage manually at the discretion of the designer. Only one recent research paper was found where Ant Colony Optimization (ACO) is used to generate and optimize topologies and paths [Shields et al., 2018]. One of the recommendations for future work was to increase the size of the distributed system network and research the effectiveness of other optimization techniques, including GAs.

The proposed method will search for a set of global optimal solutions. DV 1 and 2 are optimized, but taking the overall effect of these DVs in the total solution into account. This is illustrated with the DNA-string (chromosome) for an individual. Equation 3.1 shows the complete chromosome for a certain

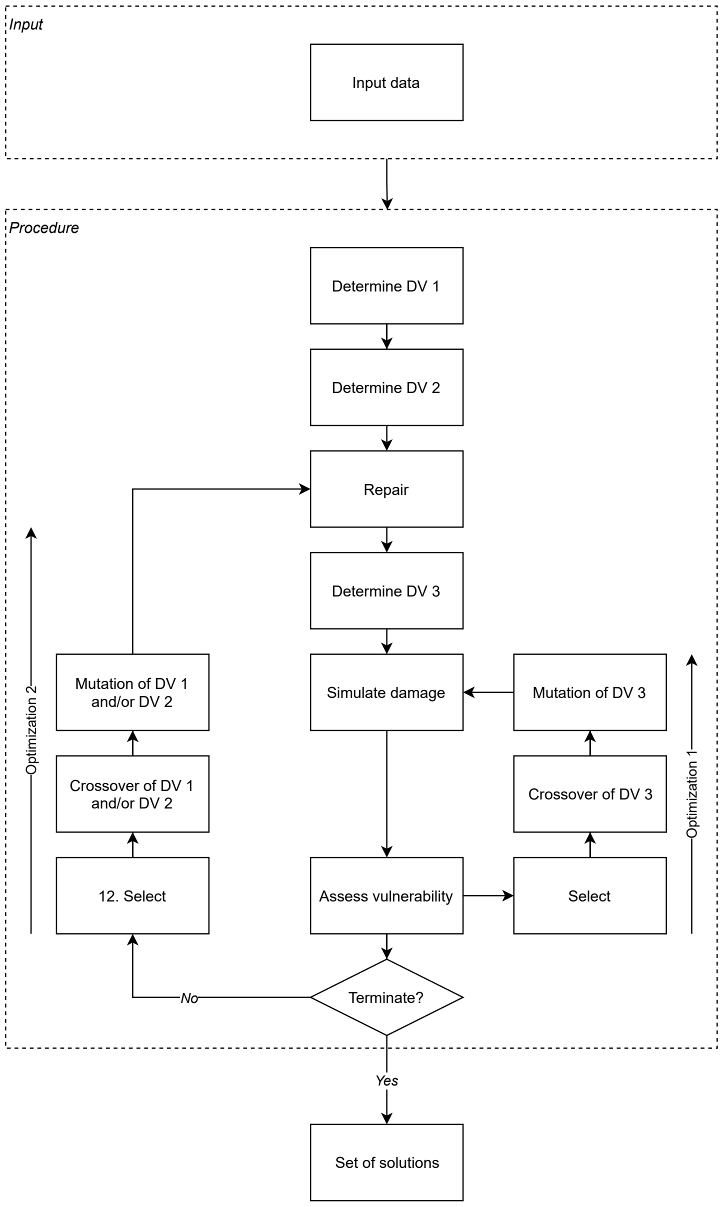


Figure 3.8: Method 2

individual[14].

$$\text{Individual} = \overbrace{\text{x }|...|\text{ x}}^{\text{DV 1}} \mid \overbrace{\text{x }|...|\text{ x}}^{\text{DV 2}} \mid \overbrace{\text{x }|...|\text{ x}}^{\text{DV 3}} \mid \overbrace{\text{x }|...|\text{ x}}^{\substack{\text{Performance}\\ \text{indicators}}} \tag{3.1}$$

To produce such a solution, DV 1 and DV 2 are determined initially, followed by DV 3. Thereafter damage is simulated and the vulnerability is assessed. This produces a solution represented by the chromosome in equation 3.1. To produce better solutions, DV 3 will be optimized locally. This will change only the parts of the chromosome indicated with *DV 3* and *Performance indicators*. The optimization will give optimal solutions given the input of DV 1 and DV 2. To improve the solutions towards a global optimum, DV 1 and DV 2 will also be altered. This will be done after optimization of DV 3 (i.e. a nested GA will be used to conduct the optimization). For this change the parts of the chromosome indicated by *DV 1* and *DV 2* will be altered, using the determined performance indicators. This will

[14]For modelling convenience the chromosome can be split in 2 separate chromosomes, one per optimization. This does however require some additional bookkeeping, to be able to regenerate a design based on the correct combination of chromosomes.

form the starting point for the next iteration. The difference with the current situation is that the overall performance of all DVs is used to find better solutions by altering DV 1 and 2, whereas in the current situations only the effect of separate DVs are taken into account in the optimization.

This is also the main advantage of this method: the DVs are changed, using the overall vulnerability. This means the effect of a change in multiple DVs can be analysed to improve the design, instead of the effect of optimization of one DV. Another advantage is that it eliminates the need for various intermediate vulnerability analyses in the design process as is the case in the current situation.

This method has also a few disadvantages. The first disadvantage of this method is that the use of GAs in general can be computationally expensive, as finding optimal solutions requires the generation and assessment of numerous individuals. The second disadvantage is that the method itself does not provide direct insight in why certain design solutions are less vulnerable, which could be used in evaluation of the design rules.

3.3.3. Comparison of methods

Table 3.3 contains a qualitative comparison of the characteristics as discussed above.

	Method 1. Directed variations	*Method 2. Holistic variations*
Computational effort	Potentially faster due to directed variations	Potentially slower due to use of GA
Improvement mechanism used	Known	Initially not known
Successful implementation	Depending on how specific causes can be identified	Depending on effective implementation of nested GA

Table 3.3: Method comparison

When the difficulties mentioned in the previous subsections can be overcome, the methodology using directed variations could provide a method which is faster and provides more knowledge initially than the method using the GA. The successful implementation however is essential, and for the first method this depends on the ability of the method to determine which variable needs to be changed, which' determination is not trivial. The successful implementation of the second method is more certain within the available time-frame, due to experience with the use of GA for ship design applications within the maritime community. Based on the probability of successful implementation the second method is preferred, if the disadvantages are not violating the requirements set in Section 3.1. This raises the question whether the disadvantages of the second method could be diminished.

Although the method using the GA is slower compared to directed variations, this is not necessarily limiting its use for the early design stage: several early stage design methods use GA. Also, the speed might be improved by a more directed search towards possible optimal solutions. The problem with directing GA is that this might lead to a unwanted limitation of optimal solutions, that is the Pareto front (see also Chapter 4, Section 4.2) might not be fully explored. Nevertheless it should be investigated whether the additional insights after vulnerability analysis could be used.

The second disadvantage is that the method provides optimal solutions, but it is not known how these solutions are created, i.e. what mechanisms cause these solutions to be less vulnerable. For the first method this is explicit due to the method: e.g. when the number of sources needs to be increased to decrease vulnerability, the decreased vulnerability is caused by the increase in the number of sources. For the second method, an additional analysis should be performed, analysing the optimal solutions and their characteristics. This could provide the same insights, but with an additional step.

3.3.4. Decision and scope

Based on the comparison of methods the methodology using a nested GA will be pursued. However, with 5 DV and multiple networks, the design problem is still too large to solve all at once. This requires further simplification of the problem in order to reach the research objective. The following simplifications and assumptions are made:

1. *End-users*. The metric driving the design process will be the operational capabilities after a damage case. In order to make the comparison of design solutions based on vulnerability aspects, the intact operational capabilities should be the same for all design solutions. This means that the number and type of end-users (i.e. effectors) will be fixed. Also the position is fixed, as this is largely dictated by other considerations than vulnerability (e.g. weapon and sensor arcs, interference, etc.).

2. *Number of DVs to vary*. Immediately varying all DVs is not preferred, as it is harder to assess the proper working of the model due to the interdependencies of the DVs. A maximum of two depending variables will be varied. Below the DVs are listed, and with which other DV variations can be combined:

   (a) *Protection*. First of all protection will not be varied. Application of protection in the form of blast bulkheads is closely related to the zonal subdivision of the ship, while other forms of protection are used to repair the effect of other chosen DV (see design rule 3 and 5). Therefore protection in the form of blast bulkheads will be only be considered as input.

   (b) *Number and type of components*. This is closely related to topology. This DV (number of suppliers and hubs, as end-users are already fixed) could be varied in combination with either one of them.

   (c) *Topology*. Topology can be varied with either the number of components as mentioned, or routing as the routing is depending on topology.

   (d) *Routing*. Routing of connections can be varied with topology.

   (e) *Position of components*. This has a close relation with routing, and a more loose relation with topology generation. The latter was mentioned earlier when the ATG tool was discussed. The relation was illustrated by a case in which a component near the bow would be connected to a component near the stern. These designs would however prove to be vulnerable and will therefore be filtered out, as the probability that this connections is damaged is high due to the length of the path. Therefore the relation between component position and topology is not a limiting factor. The relation with routing is that the component position is needed for routing. As routing of connections is always the last DV to be determined, this relation is not a limiting factor either for variation. This means the position of components can always be varied.

This leaves the following possible DV variations:

1. Number and type of components
2. Topology and positions
3. Routing

With the limitation of varying a maximum of two dependent DVs, either 1 and 2 or 2 and 3 can be varied. For this research the second option is preferred for several reasons:

- The number of design rules associated with the second option is larger, meaning the second option provides more possibilities for design rule evaluation.

- Manually adding a variation of the DV that is not automatically varied is easier for the number and type of components then for routing. Manually varying routing would mean determining for each connection in each design alternative routings and evaluating their performance. The component variations are more limited, and only requires a new definition of the input.

The choice to vary the position leads to one additional assumption/simplification. It is assumed that the constraining physical architecture is known and fixed. This means that the ship's hull and the subdivision are known to a certain level. When determining positions, the weight and space requirements of the distributed systems versus the constraining architecture will only be taken into as defined by the designer (see Chapter 4, Subsection 4.3.1)

3.4. Proposed method

Combining the choices made in the previous sections, a mathematical model is formulated as shown in Figure 3.9. This model is able to bridge the formulated gap. The first part of the gap was the translation of the vulnerability assessment output into an operational metric. This is incorporated in step 8 in the model. The second part of the gap was a refined method to simulate different damage extents. This is incorporated in step 7 of the model. The third gap existed of 3 aspects: which DV to vary, when and to what extent. The first two aspects are addressed in the way the model is set-up. The last aspect is further determined during implementation of the model (e.g. what are the bounds for position variation, and are they the same for all components?).

The model shows potential feedback from the vulnerability assessment method to the second optimization process. Initially the model will be implemented without this feedback mechanism. Thereafter the effect of a feedback loop will be investigated.

As the problem was analysed in the previous chapters and a mathematical model is now formulated, the next steps will be to implement the model [Shiflet and Shiflet, 2006] and verify the working. Following implementation, the model will be used to generate solutions with the simplifications discussed in Section 3.3.4. The resulting optimal solutions will (1) be analysed to identify common features and (2) compared to design rules, to determine whether these design rules are reflected in the generated solutions.

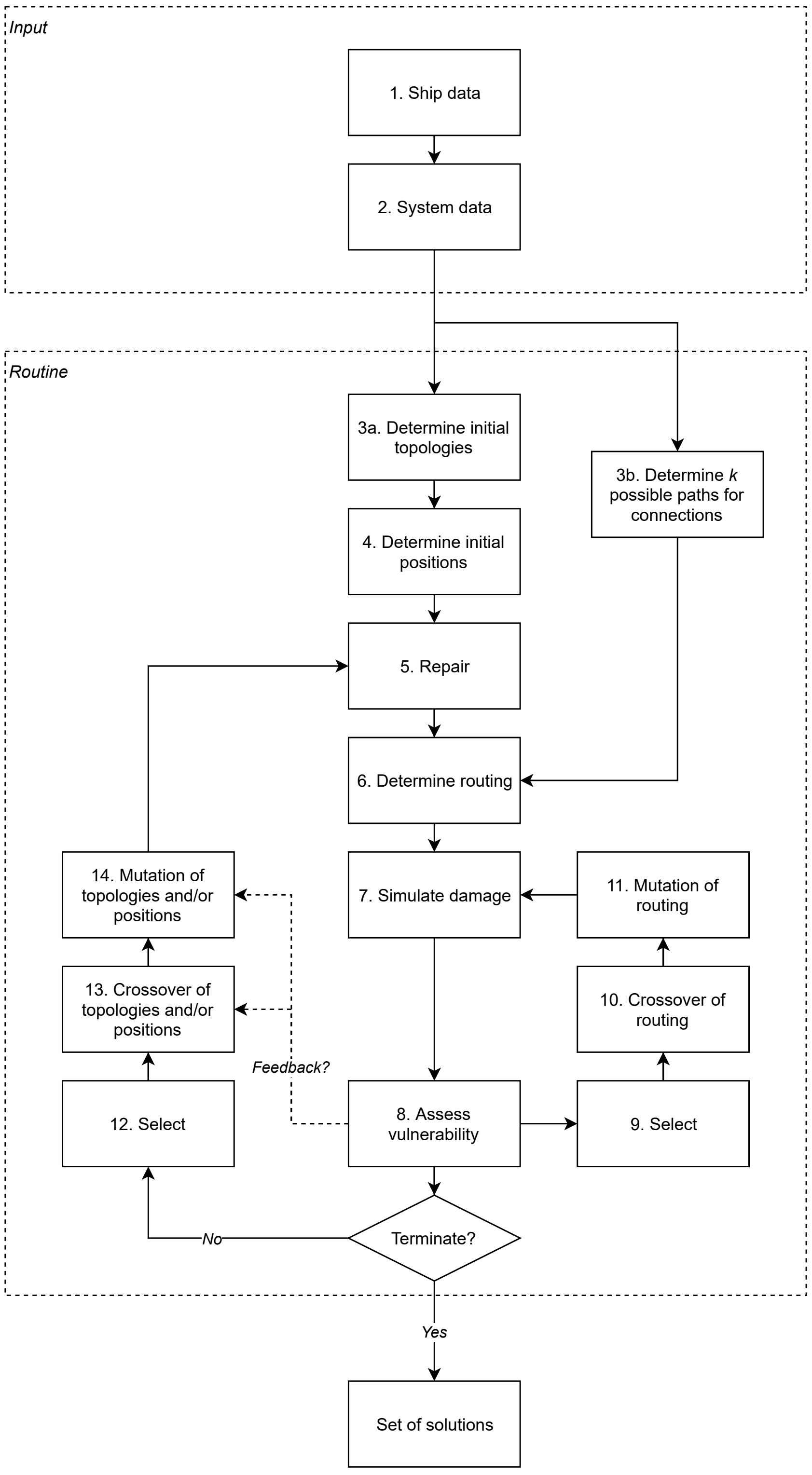


Figure 3.9: Mathematical model

4 Implementation of the mathematical model and verification

This chapter presents the way in which the mathematical model as described in the previous chapter is implemented in Matlab®. The decisions made during implementation will be discussed, along with the inherent limitations and consequences accompanying them. This is followed by verification of the separate functions and validation of the model. The overall purpose of this chapter is to provide the reader with confidence in the produced model and its limitations, and therewith the solutions the model will produce.

4.1. Choosing an optimization algorithm

In the previous chapter a model was proposed using a nested GA. The reason to opt for a GA as optimization method was not discussed yet, as well as which GA will be used. This will be shortly discussed in this section.

The following factors were considered for selecting the optimization method:

1. *Ability to deal with the type of problem*. The nature of the problem excludes some optimization methods. An example of such an inadequate method for this problem is optimization through linear programming, where the optimization problem is defined and bounded by linear mathematical functions [Hillier and Lieberman, 2015].
2. *Ability to be used both optimizations in a nested configuration*. Due to the limited time available and to ease implementation, the optimization method should be able to handle both optimizations (topology/position optimization and routing optimization). This means that certain optimization techniques are dismissed, as they are only able to solve part of the optimization problem. An example is ACO which can be used to optimize routings as this optimization technique is used to find optimal paths through a network [Blum, 2005]. However ACO can not easily be used op optimize the topology and positions of the components, as describing this problem as a network with paths is harder to achieve, if possible at all[15].
3. *Ability to optimize multiple objectives*. As there are many relevant aspects of a design besides vulnerability (e.g. costs or optimal use of volume available), the optimization method should be able to optimize towards more than one objective. This will enable the designer to influence the design effort by defining different objective functions, and allow the designer to improve the trade-offs required to make a decision.
4. *Availability of the method*. As mentioned in Chapter 1, the focus of this research is on implementing new vulnerability assessment techniques to guide the design process rather than implementation of the best optimization technique that exists. Therefore the method should be readily

[15]As mentioned in Chapter 3, Section 3.3.2, ACO was used in [Shields et al., 2018] to optimize topology and routing, but it was assumed that the positions are known. The ACO-algorithm was significantly expanded and tailor-made for the problem.

available if possible, as implementation of the optimization method itself is not the main focus of the research.

5. *Familiarity and past performance*. Although the optimization method itself is not the main focus of the research, it will still influence the ability of the model to produce a set of solutions. Therefore methods which are regularly used in maritime design applications and which proved to be able to produce such a set of solutions are preferred.

One of the methods satisfying all the requirements is optimization using GAs. As observed during the literature review, GAs are used extensively for optimizing general arrangements in ship synthesis models and topology design as demonstrated in the ATG. More specifically, the ship synthesis model in use at DMO (Packing) and the ATG use the Non-dominated Sorting Genetic Algorithm II (NSGA-II) developed by [Deb et al., 2002] making it a prime candidate for implementation. To reduce the chance that the decision to use a nested implementation of NSGA-II would have a dramatic impact on the performance of the resulting model (i.e. the optimization is too slow or unable to create diverse solutions), a quick scan of existing research was conducted. The main purpose was to identify whether other common GAs, like the Strength Pareto Evolutionary Algorithm-2 (SPEA-2) [Zitzler et al., 2001] are clearly outperforming NSGA-II. Although the performance depends on the problem definition, no such indication was found in the reviewed research of nested setups (e.g. [Wagner, 2016]). Review and implementation of possibly better suited optimization methods, possibly combining different methods for both optimizations, should however be a topic of future research.

4.2. Implementation of the nested GA

The GA is the framework of the model, and the other functions as described in the mathematical model will support the functioning of the GA. This section will discuss the working of the NSGA-II, more specifically how a nested implementation of the GA is achieved and which evaluation criteria will be used.

The standard NSGA-II consists of 5 distinct steps as shown in Figure 4.1. An initial population is created with a certain number of individuals (chromosomes) (n_{pop}), and the fitness of each individual is evaluated for certain objective functions. If the termination criteria are met (this can be an achieved score for the objective functions, but also a number of generations or time passed since the algorithm started), the generation of new designs will stop. Otherwise, 2 individuals from the parent population (the population which was evaluated) will be selected. Parts of the individuals will be crossed-over, and thereafter each gene of the chromosome might be mutated. This will be repeated until a child population is created with the original n_{pop}, which is then again evaluated. This repeats until the termination criteria are met. A more detailed explanation, including the working of the different GA functions and how diversity is maintained can be found in [Deb et al., 2002].

As discussed in the previous chapter, the reason to use a nested optimization is twofold. First, the actual impact of certain decisions on the vulnerability will be used to improve designs, instead of derived metrics. A second advantage is that only one vulnerability analysis needs to be conducted. This does however require an adaptation of the GA. This adaptation is shown in Figure 4.2. Instead of directly evaluating a certain topology and position combination, first a routing optimization will be conducted. This optimization will be conducted for every topology/position individual and will result in a set of routing solutions for that specific topology/position combination. From all these routing options, the best solutions will be selected and fed back to the topology/position optimization (in the remainder of this thesis this optimization will be referred to as the outer GA or outer optimization). For multi-objective optimizations it is common that not one particular solution scores best on all objectives. Instead the best solutions are Pareto-optimal, or positioned on the Pareto-front. This means that for this set of solutions, if the score of one objective increases, the score for the other objective will decrease. Solutions on the Pareto-front are outperforming other solutions on one objective when the other objective scores are the same. In the nested optimization, all the scores of the solutions on the Pareto-front of the routing optimization are fed back to the topology/position optimization loop. This means that if there are 3 Pareto-optimal solutions found, the topology/position chromosome will be copied 3 times in the outer GA population. Each topology/position individual will be assigned the scores of a particular solution, and the unique ID of the routing individual (age) will be stored in the chromosome as well for reconstructability.

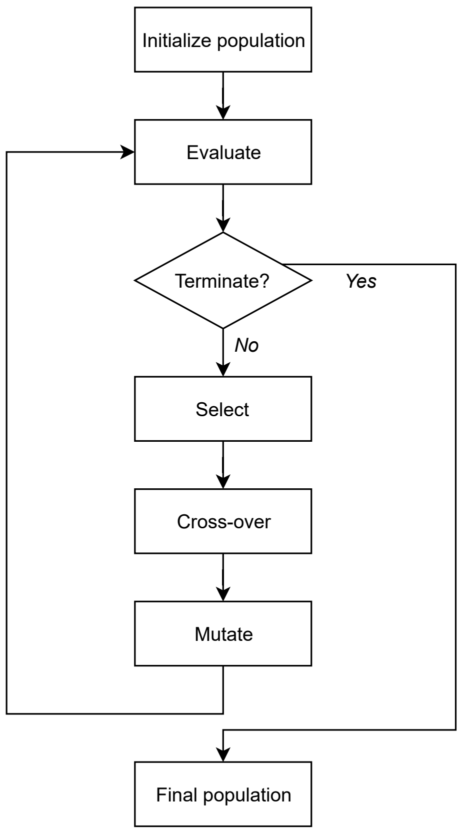


Figure 4.1: Standard process of the NSGA-II

This has two implications for the working of the outer GA. First of all, the population of the outer GA will grow. This means that after the evaluation of each population, the population size n_{pop} will need to be reduced to its original proportion during selection, mutation and cross-over processes. Otherwise the population would grow uncontrollably, resulting in an infeasible computational effort required to generate and evaluate all designs. The second implication is related to the fact that the size of the Pareto-front of the routing optimization is variable. This means that the number of occurrences of a topology/position individual in the population of the outer GA after evaluation depend on the size of the Pareto-front of the inner GA. This could instigate a bias in the algorithm towards certain topology/position individuals when there is a large difference in size of the Pareto-fronts of the individuals, as there is an increased probability that a topology/position combination is selected for mutation and cross-over. However, during experiments and the conducted testcases it was observed that the size of the Pareto-fronts are the same order of magnitude for different individuals, and therefore no bias was observed.

Finally the evaluation criteria are determined. One of the evaluation criteria was already discussed at length in the previous chapters, being the operationally driven vulnerability score. How this score is determined exactly will be further discussed in Section 4.3.7. As mentioned earlier, there are other factors relevant for a design, of which cost is a common and important one. This objective can also be related easily to the DVs of the optimization, as more connections and longer paths resulting from varying routing or component positions will increase costs. Therefore the other selected objective function is related to the cost of a design. This objective function is a basic one. As the components of each system configuration are fixed, the component costs does not need to be calculated in the objective function as it simply raises the value with a fixed amount. If cases with a manual variation of the system components are compared, one should correct for the component costs if a specific cost comparison between designs is conducted. Disregarding the component costs means only the path

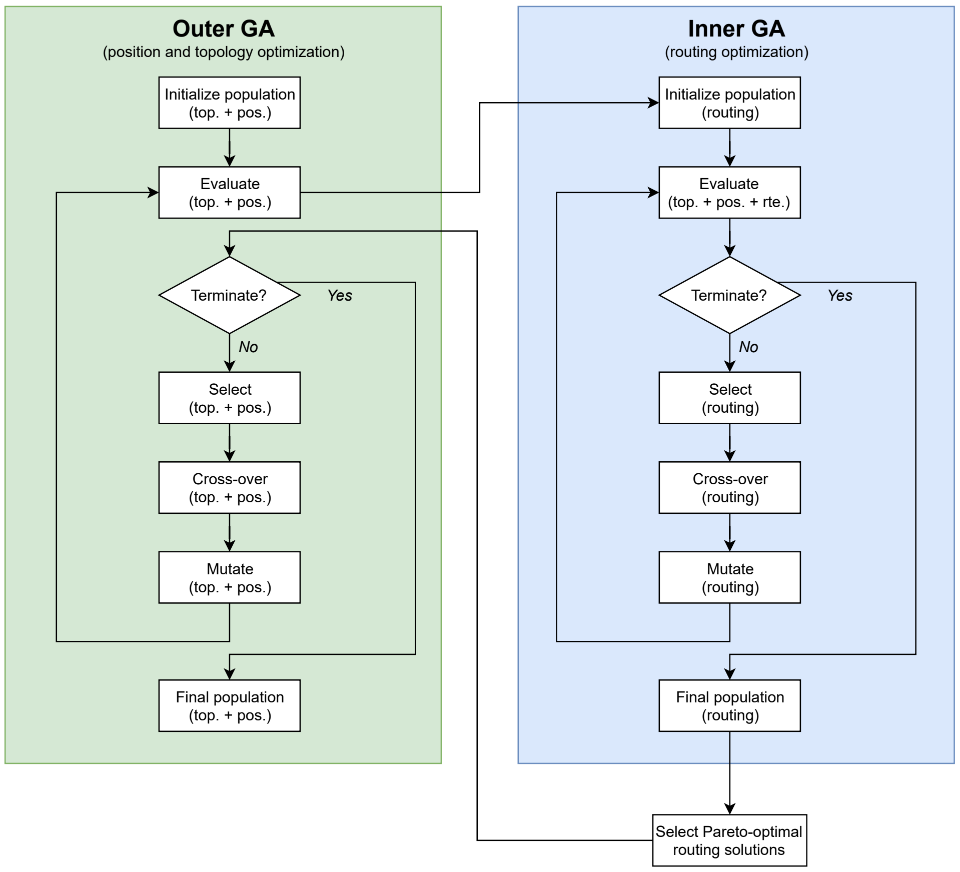

Outer GA
(position and topology optimization)
Initialize population (top. + pos.)
Evaluate (top. + pos.)
Terminate?
Yes
No
Select (top. + pos.)
Cross-over (top. + pos.)
Mutate (top. + pos.)
Final population (top. + pos.)
Inner GA
(routing optimization)
Initialize population (routing)
Evaluate (top. + pos. + rte.)
Terminate?
Yes
No
Select (routing)
Cross-over (routing)
Mutate (routing)
Final population (routing)
Select Pareto-optimal routing solutions

Figure 4.2: Nested implementation of NSGA-II

costs of the different designs are relevant. In the implementation the paths costs will be approximated by the sum of the length of all the chosen paths. This is a very crude approximation of the path costs, as it neglects all common paths of the same type of resource and assumes every type of connection (e.g. power cables or CW-pipes) costs the same. The main purpose of the cost objective however is to act as counterweight for the vulnerability score; it should prevent generating designs in which every possible connection is made just to decrease vulnerability, simply because there is no penalty. The cost objective will be calculated as follows, with P the number of paths and l the length of a chosen path i:

$$f_1 = \sum_{i=1}^{P} l_i \tag{4.1}$$

For this model a Matlab® implementation of NSGA-II which is in use at DMO will be used with some minor adjustments. The minor adjustments concern the selection function, which is adapted to be able to deal with a growing population size, and ranking function of individuals, which is adapted to also sort on the actual value of constraint violation. With implementation of the nested NSGA-II, the framework of the model is ready. The next section will discuss the functions required to provide this framework with the correct input and ensures proper processing of this input.

4.3. Functions

This section will discuss the implementation of the different aspects of the mathematical model, decisions or assumptions made, and the consequences (e.g. possible bias resulting from the implementation) for (results generated by) the algorithm.

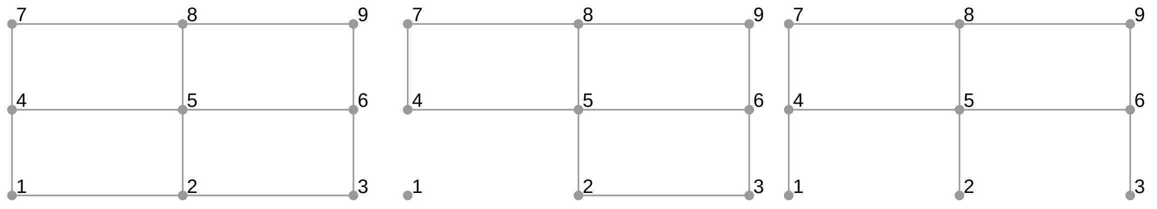
4.3.1. Input

The model needs to be fed with input in order to be able to generate designs. The input required consists of five aspects: the physical constraining architecture (ship hull and spaces), system components and networks, contribution to capabilities, the threat and variable position ranges.

Physical input

The physical input consists of network diagrams, representing the constraining physical architecture for the system networks as used by [Duchateau et al., 2018]. These networks are defined by adjacency matrices. The following networks are required:

- The network diagram representing all the spaces is the subdivision adjacency network. The spaces are represented by nodes positioned at the Centre of Gravity (COG) of the space, and edges showing the adjacency to other spaces (Figure 4.3a).
- Not all spaces (e.g. exhausts and tanks) can be used for routing of connections. The spaces that can be connected are defined in the routing adjacency network. A path between two nodes is only possible when an edge exists between them. Figure 4.3b shows the routing adjacency network. While space 1 is spatially adjacent to space 2 and 4, there can be no connection routed through space 1 (e.g. because this space is a tank).
- Damage propagation is defined by the damage adjacency network. This will connect all spaces which can transfer damage to the next space. The damage adjacency network will consist of all connections between spaces as defined in the subdivision network, with the exception of the spaces which are separated with a blast bulkhead. Figure 4.3c shows an example. While space 2 is spatially adjacent to spaces 1 and 3, there can be no damage propagation between these spaces as there are blast bulkheads between space 1 and 2 and space 2 and 3.



(a) Subdivision network (b) Routing network (c) Damage network

Figure 4.3: Networks describing the constraining physical architecture

Besides these network diagrams general information about the physical arrangement is used for calculating hit probabilities. This information consists of the limits of a space in 3 dimensions, the vulnerable areas and the relative position (port, starboard, at the bow or aft and at the top or bottom) of a space on board the ship when in the outer layer (i.e. when the space is susceptible to a direct hit).

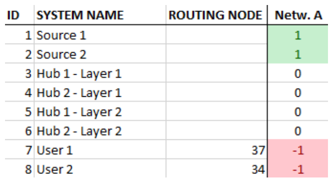
It should be noted that the definition of these network diagrams largely impacts the quality of the solutions generated, but also the computational effort required to generate the solutions, especially when multiple hits are evaluated. A constraining physical architecture for example where the ship is divided only in port and starboard spaces will not lead to meaningful results when evaluating small damages in which only one space is affected. In this case, routing connections via port or starboard will not make a difference (assuming there is no difference in hit probability based on which side the space is). For this a third layer should be added at the centreline of the ship. This does not necessarily increase the computational load, as long as these additional spaces are not susceptible to direct hits as will be discussed in Section 4.3.6.

Another remark is that the spaces not necessarily need to reflect actual spaces onboard the ship. It might as well be a zone-deck compartment, which is subdivided in 6 'spaces' (port/centreline/starboard and fore/aft parts). When defining the adjacency networks, the designer should keep in mind the level of detail of the constraining architecture should be in line with the design stage for which the model is intended.

System input

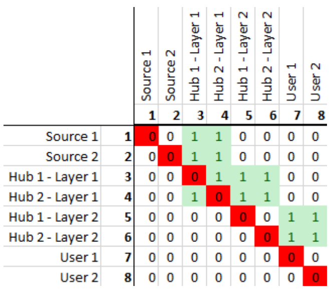
The system input consists of the different components of the distributed networks. For each component several aspects are defined per distribution network in an Excel-sheet (Figure 4.4), which is processed by a Matlab-script. The aspects which can or need to be defined are:

- *Role (required)*. A component within a network can be a supplier, hub, user or have no role at all in a specific network. A component can have different roles in different networks, i.e. a Chilled Water Plant (CWP) can be a user in the electrical power network, whereas it is a supplier in the CW-network and has no role in the data-network.
- *Position (optional)*. If a component's position is fixed, the node ID in the subdivision network corresponding with the position is defined.
- *Possible connections (optional)*. Based on the defined role, the model has a possibility to automatically generate a system adjacency matrix containing the possible connections A_{pos}. The system adjacency matrix is used by the algorithm to define the elements of the chromosome in the outer GA and to repair infeasible topologies (Section 4.3.4). For more complex networks automatic generation will be insufficient as it can only distinguish between the 3 earlier defined component roles. In more complex networks however a further subdivision might be required. An example is a 440V network, in which the hub layer is further subdivided in 2 hub layers: a layer consisting of Main Switch Boards (MSWBs) and a layer consisting of LCs. When A_{pos} is automatically generated, both the MSWBs and LCs can be directly fed by the suppliers, and all hubs can be bidirectionally connected to each other. This would mean a LC would be able to supply a MSWB which is in reality not the case, although the other way around is possible, meaning the possible connection is unidirectional. To deal with these kind of networks the option exists to manually define A_{pos} for each network (Figure 4.4b) and use this as input. This also allows for the possibility to create spatially redundant paths[16]: a possible connection is defined by a 1, if there should be a possibility for 1 spatially redundant path this is defined by a 2, etc.



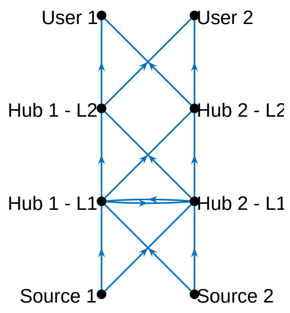
ID	SYSTEM NAME	ROUTING NODE	Netw. A
1	Source 1		1
2	Source 2		1
3	Hub 1 - Layer 1		0
4	Hub 2 - Layer 1		0
5	Hub 1 - Layer 2		0
6	Hub 2 - Layer 2		0
7	User 1	37	-1
8	User 2	34	-1

(a) Definition of components, positions and role



		Source 1	Source 2	Hub 1 - Layer 1	Hub 2 - Layer 1	Hub 1 - Layer 2	Hub 2 - Layer 2	User 1	User 2
		1	2	3	4	5	6	7	8
Source 1	1	0	0	1	1	0	0	0	0
Source 2	2	0	0	1	1	0	0	0	0
Hub 1 - Layer 1	3	0	0	0	1	1	1	0	0
Hub 2 - Layer 1	4	0	0	1	0	1	1	0	0
Hub 1 - Layer 2	5	0	0	0	0	0	0	1	1
Hub 2 - Layer 2	6	0	0	0	0	0	0	1	1
User 1	7	0	0	0	0	0	0	0	0
User 2	8	0	0	0	0	0	0	0	0

(b) Adjacency matrix (A_{pos})



(c) Network representation

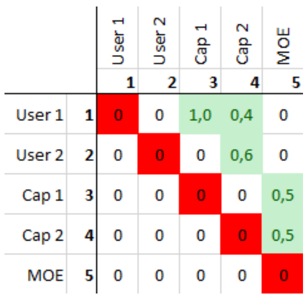
Figure 4.4: Example of the definition of system input

Capabilities and Residual Capability Matrix (RCM) input

The input for calculation of capability/vulnerability scores and the RCM is defined in the same Excel-sheet as the system input. The input defines how effectors are contributing to capabilities and what requirements are set on the level these capabilities need to have:

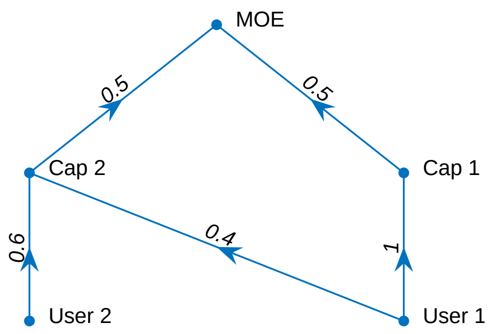
- *Capabilities (required)*. The contribution of effectors to capabilities needs to be defined. The capabilities are defined by matrices as well, containing the relation and weight of the contribution. A component can contribute to multiple (sub-)capabilities, for example a gun can contribute to both the AAW- and ASuW-capability., which in the end will contribute to the MOE as shown in Figure 4.5. The actual values will be determined using expert-opinion and the AHP as described in Chapter 3, Section 3.2.2

[16]Spatially redundant paths in this context are 2 paths between one source and one sink. Another type of redundant paths is when the redundancy is achieved functionally. In that case, there are 2 paths between 3 nodes: one node is a sink, the other 2 are (functionally redundant) sources.



		User 1	User 2	Cap 1	Cap 2	MOE
		1	2	3	4	5
User 1	1	0	0	1,0	0,4	0
User 2	2	0	0	0	0,6	0
Cap 1	3	0	0	0	0	0,5
Cap 2	4	0	0	0	0	0,5
MOE	5	0	0	0	0	0

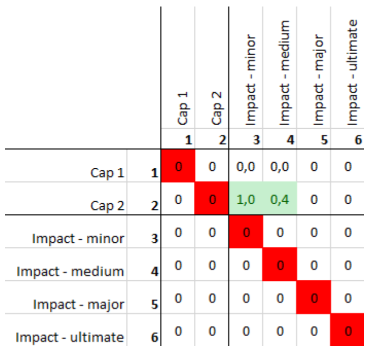
(a) Definition capability relations



(b) Resulting capability tree

Figure 4.5: Example the contribution of effectors to capabilities and MOE

- *Residual Capability Matrix (RCM) (required)*. For several magnitudes of impact the minimum required level of a certain capability can be defined. This will be used to determine whether a generated design is feasible. Also the designer will need to define a percentage, C_{RCM}, defining in how many damage cases the requirements need to be adhered to. The need for this percentage will be further discussed in Section 4.3.7. An example of the input is shown in Figure 4.6, where capability 2 needs to be fully available after a hit with minor impact, and 40% for a hit with medium impact, in the number of cases as defined by C_{RCM} (not shown).



		Cap 1	Cap 2	Impact - minor	Impact - medium	Impact - major	Impact - ultimate
		1	2	3	4	5	6
Cap 1	1	0	0	0,0	0,0	0	0
Cap 2	2	0	0	1,0	0,4	0	0
Impact - minor	3	0	0	0	0	0	0
Impact - medium	4	0	0	0	0	0	0
Impact - major	5	0	0	0	0	0	0
Impact - ultimate	6	0	0	0	0	0	0

Figure 4.6: Example of the definition of RCM

Threat input

The threat input is required to define a list with damage cases for which the vulnerability of the designs will be evaluated. The following aspects of the threat need to be defined:

- *Hit probabilities*. This defines which Probability Density Function (PDF) will be used to determine the hit probability for different spaces. A sea-skimming missile approaching from abeam for instance will have a higher probability to hit amidships then to hit the bow or mast, while a high-diving missile will be more likely to hit the mast than the bow. The probability distribution will be combined with a definition of vulnerable spaces, as the PDF is a continuous distribution, but the vulnerable ship area is bounded. This will be further discussed in Section 4.3.6.

- *Damage size*. This defines the extent of the damage as a result of a hit. For a missile this might be the complete zone-deck compartment to which the space belongs, while a Rocket Propelled Grenade (RPG) hit might only result in a singe damaged space.

- *Impact category*. The impact category is linked to the RCM. A missile threat will for example be categorized as a major impact, while a RPG impact will be a minor impact and as such have different residual capability requirements. Two RPG hits might however be classified as medium impact.

These parameters will be used to compile a damage case list which will be discussed in Section 4.3.6.

Variable position ranges input

As discussed earlier the system input defines whether the position of a component is fixed or variable. For each component the designer needs to define the range of spaces in which the component can be positioned. If there are no restrictions, the range consists of all nodes in the routing network. The designer can limit this range based on the type of components or for practical considerations. If for example the component is a CWP, it is probably not possible to position it in the top of the superstructure of the ship due to pump capacity restrictions. The range can also be limited based on the available volume in a space and the volume of a component. Another consideration of the designer can be to already assign certain components to certain zones.

Limitation can be beneficial as it ensures the computational effort is focussed on only examining solutions which are practically feasible. The designer might however also introduce bias in the model based on his preferences when the limitations are too severe and based on his preferences (such as the limitations of component positions to certain zones), as the design space is manually limited and therewith the ability of the GA to improve a design. Therefore caution should be used when applying these kind of limitations.

4.3.2. Determine k-possible paths

Once the routing network is defined, all the possible combinations of routing nodes which might need to be connected are known. As the component positions are one of the DVs which are varied, it is unknown at the moment of initialization which specific routing node pairs need to be connected. During each generation, different routing nodes might need to be connected as component positions change. Instead of determining the k-shortest paths during generation for each connection present in an individual design, the k-shortest paths for each possible combination is determined when the algorithm is initialized. For determining the k-shortest paths initially the algorithm developed by [Yen, 1971] was used. This algorithm was also used in [Duchateau et al., 2018] for routing optimization.

As the number of possible connections to be routed are significantly larger due to the variable positions of components, the choice for k is also more limited due to the computational effort required. As an example, the testcase as presented in [Duchateau et al., 2018] required 43 connections to be routed, while all possible connections in the testcase which will be used later is $\binom{98}{2} = 4753$. The model should however be provided with enough path variations to choose from, in order to decrease vulnerability by finding better suited paths. The fact that k is limited decreases the number of options, and this effect of a limited k is further aggravated by 3 factors.

The first factor is the way in which Yen's algorithm finds alternatives from the shortest path, only looking at costs for selection of alternatives. The second factor is the fact that in the generated designs, paths are possibly further apart than in a 'regular' design. The model might choose to connect a component in the aft part of the ship to a component at the bow, which is unusual in a regular design. The last factor is that in 3D designs, alternatives may not make any difference at all from a vulnerability point of view. This is the case when variations in paths are made over the width of the ship, while only zone-deck compartment size damages are considered. This is illustrated in Figure 4.7. When in this figure node 39 is hit, a compartment size damage will also affect nodes 79 and 119. Therefore the shown path variations are indifferent for this type of damages.

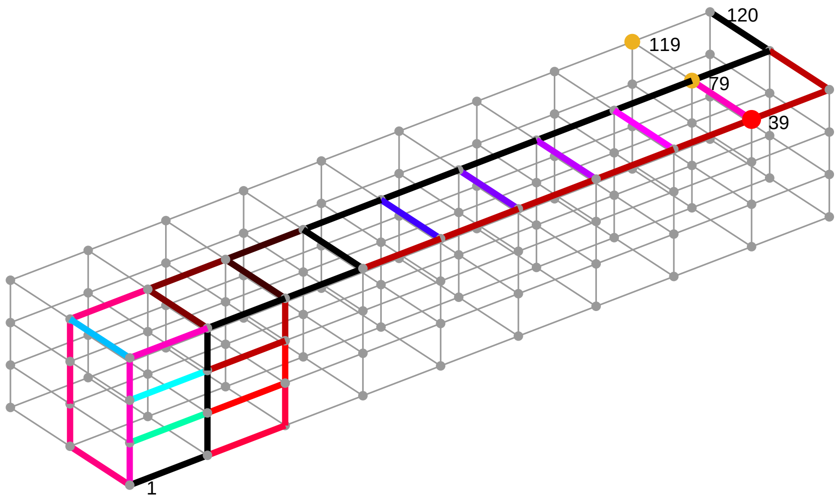


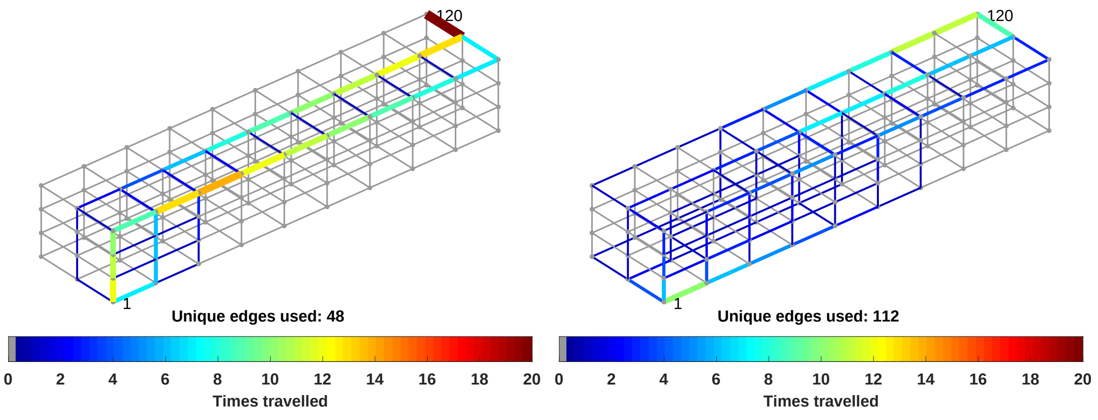
Figure 4.7: k-Shortest paths between node 1 and 120 using Yen's algorithm ($k = 20$). Hit node is indicated in red, other affected nodes for compartment size damage in orange.

Figure 4.7 shows the effect of these factors for the 20 shortest paths between 2 distant routing nodes, where each color represents a unique path. As can be seen, the different paths share a significant amount of edges. This means that the alternatives to the shortest paths will have a very limited effect on the vulnerability of the design. Instead of the 20 shortest paths, one should generate the 20 shortest paths, which are also sufficiently different. This can be achieved by algorithms which find the k-Shortest Path with Limited Overlap (kSPwLO), as discussed in [Chondrogiannis et al., 2015] and [Chondrogiannis et al., 2020]. The possibility to implement these algorithms has been investigated, but due to time constraints an alternative solution is pursued. A straightforward form of Chondrogiannis' baseline solution is implemented where sufficiently different paths are discovered using brute force. Figure 4.8 shows the pseudo-code of the algorithm for this approach. The sizing of the initial k as described in line 4 depends on the calculation time available for determination of the k-possible paths.

Algorithm 1: Determine k_{dis}-shortest paths

```
1  Choose the number of dissimilar paths required $k_{dis}$;
2  Find all possible combinations of routing nodes;
3  foreach combination do find the shortest path;
4  Determine initial $k$ for each routing node combination based on
     nr. of nodes in shortest path, where $k >> k_{dis}$;
5  foreach routing node combination do
6      Determine the $k$-shortest paths, using Yen's algorithm;
7      foreach found path do
8          Determine the similarity $\theta$ with other paths, by
             determining how many common routing nodes are
             present in any 2 path combinations, excluding the start
             and end node
9      Find the $k_{dis}$-paths by selecting the paths for which $\theta < \theta_{req}$;
10     while nr. of found dissimilar paths < $k_{dis}$ do
11         Increase $\theta_{req}$
```

Figure 4.8: Pseudo-code for finding k_{dis}-shortest paths



(a) Heatmap conventional method

(b) Heatmap dissimilar path method

Figure 4.9: Comparison of paths found by conventional and dissimilar method. $k_{conv/dis} = 20$

The results of this approach are shown in Figure 4.9. This shows a heatmap of the edges used for $k = 20$ and $k_{dis} = 20$. As can be seen, the approach described above actually finds alternative paths, as opposed to the conventional method. Of course this could also be achieved by stopping after line 6, as the described method simply select k_{dis}-paths from a larger set. This would however mean that a large number of paths could be selected by the routing optimization algorithm, while these are actually not really alternative paths. This should be avoided as computational time is wasted evaluating these undesirable alternatives. Further limitation of undesirable alternative paths is also achieved by automatically limiting k_{dis} for adjacent nodes. For nodes further apart, a limitation of k_{dis} is for this model not beneficial, as k_{dis} already has a relatively low order.

Off course this method requires computational effort as well[17], but the impact is limited as the results can be re-used as long as the constraining physical architecture is not altered. Implementation of the earlier mentioned (more efficient) kSPwLO-algorithms would further reduce the required effort.

4.3.3. Determine initial topologies and positions

The purpose of determining the initial topologies and positions is to initiate the structure of the outer GA chromosome and generate an initial design consisting of a topology and component positions. The first part of the the chromosome are genes that represent the variable component positions, x_{pos}. This part will be followed by the genes representing the possible connections, x_{con}. Each gene of x_{con} is connected to a non-zero value in the A_{pos} to determine the number of genes required. The third part of the chromosome will contain values which are required by the GA to function, such as objective scores, constraint violations, ranking, age of the topology/position individual and age of routing individual. The latter 2 values are unique identifiers for individuals in the optimizations. The structure of the chromosome in the outer GA is shown in Figure 4.10.

$$\text{Chromosome outer GA} = \underbrace{x\,|...|\,x}_{x_{pos}}\;|\;\underbrace{x\,|...|\,x}_{x_{con}}\;|\;\underbrace{x\,|\,x}_{\text{Obj. scores}}\;|\;\underbrace{x}_{\text{Viol.}}\;|\;\underbrace{x}_{\text{Rank}}\;|\;\underbrace{x}_{\text{Crowd}}\;|\;\underbrace{x}_{\text{Age top.}}\;|\;\underbrace{x}_{\text{Age rte}}$$

Figure 4.10: Structure of chromosome of outer GA

The values elements in x_{pos} can attain are depending on the range of positions which is defined in the input. The lower bound is 1, while the upper bound is the number of elements in the position ranges for that component. There are several ways to determine the initial variable positions. First, random positions can be assigned. This could however mean that similar components are positioned in the

[17]For the 4753 paths of the testcase, with $k_{dis} = 20$ and an initial k-size of $k = 15n_s^2$ (n_s is number of nodes in shortest path), the determination of all paths took approximately 3.5 hours with parallel computing on a Quad-core laptop

same space or closely together, which is unlikely to generate a robust design. Another way is to use elite solutions, manually assigning specific positions. A method in between these 2 extremes is to subdivide the complete range in sub-ranges. If there are for example 2 MSWBs, which can both be positioned in the same range of 40 spaces, the first MSWB will be randomly assigned a value in the first half of the range, while the second MSWB will be randomly assigned a value in the second part of the range. This generates a certain degree of separation initially, but without the need to generate numerous designs manually, and maintaining the flexibility for the algorithm to position components closer to each other later on.

The lower and upper bound for values in the part representing the different connections x_{con} are 0 and 1, where 1 represents a connection and 0 the absence of a connection. The initial values for the connections are randomly determined. If this generates an infeasible topology, than this will be repaired in the next step.

4.3.4. Repair function

Following the generation of a population, the chromosome likely needs to be repaired. The purpose of this repair function is to only pass position and topology combinations to the routing optimization which are functioning in intact condition.

During implementation of the algorithm the decision was made to focus less on considerations associated with the positioning of components as this is a rather complex topic itself. Normally the ship synthesis models discussed in Chapter 2 will deal with this topic as it encompasses for example weight and space considerations. Therefore the positions generated by the GA will only be repaired, when they are out of the predefined lower and upper bounds. In that case the position will be set to the lower or upper bound (whichever boundary is violated). The model should and can however be further improved with more attention for the variable positioning, where for example a basic check of available volume in a space and volume required by a component is conducted, or where a large component (e.g. a Diesel Generator (DG)) can be assigned to multiple spaces, making the model more flexible.

The topologies will most likely need to be repaired to become feasible. A topology is feasible when, in intact condition, all suppliers and users are connected in the network and the hubs have a meaningful function. In this context hubs have a meaningful function when they are connecting a hub or supplier to another hub or user. This means no floating hubs (i.e. a hub is only connected to one other hub) should exist, as this is essentially a configuration with less hubs and therefore not comparable to the other topologies (Figure 4.11).

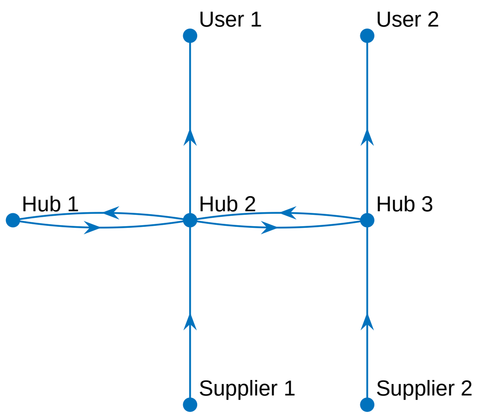


Figure 4.11: Topology in which hub 1 has no meaningful function

Figure 4.12 show the pseudo-code for the topology repair function. As shown in the pseudo-code, the repair functions are defined based on the type of component. The basic principle of each repair function is the same. First, determine for a certain node which connections are planned based on the chromosome. If this is insufficient, remove or add additional connections based on the A_{pos} and the distance to other components (i.e. connect or stay connected to the nearest possible node).

```
Algorithm 2: Repair topologies
1  Determine component positions based on $x_{con}$;
2  Determine Euclidean distance between components;
3  foreach distribution network do
4      Get $A_{pos}$ as defined in the input;
5      Generate the intact system adjacency network matrix
         $A_{net,intact}$ based on $x_{con}$;
6      Make all hub-hub connections bidirectional;
7      foreach node in the network do
8          if node = supplier then
9              Repair using supplier repair fcn
10         else if node = hub then
11             Repair using hub repair fcn
12         else if node = user then
13             Repair using user repair fcn
14     if floating hubs are present then
15         Connect hub to nearest possible hub or user
16 Update $x_{con}$
```

Figure 4.12: Pseudo-code for repairing intact topologies

The reason that the repair function is different for every node is that the connections to be added are different for each type of component. When the lowest layer consist of suppliers, and the highest layer of users, a user will need to be connected to the layer below it, while a supplier will need to be connected to the layer above it. This requires a different way of determining which connections should be made. Figures 4.13 through 4.15 show the pseudo-code for the respective repair functions.

```
Algorithm 3: Repair supplier nodes
1  Get possible destinations for outgoing connections from $A_{pos}$;
2  Get the current connections for the supplier from $A_{net,intact}$;
3  if 1 connection then
4      continue
5  else if no connections then
6      Connect to nearest destination node
7  else if >1 connection then
8      Keep connection to nearest destination node, remove all
         others
```

Figure 4.13: Repair function for supplier-nodes

Algorithm 4: Repair hub nodes

```
1  Get possible origins for incoming connections from $A_{pos}$;
2  Get the current connections for the hub from $A_{net,intact}$;
3  Keep all planned connections to hubs;
4  Keep earlier established supplier-hub connections;
5  if hub is being supplied by supplier or other hub which is being
     supplied then
6      continue
7  else
8      Connect to a connected hub in same layer
9  if no connected hubs in same layer (only possible for multiple
     hub layers) then
10     Connect to nearest hub in lower layer
```

Figure 4.14: Repair function for hub-nodes

Algorithm 5: Repair user nodes

```
1  Get possible origins for incoming connections from $A_{pos}$;
2  Get the current connections for the hub from $A_{net,intact}$;
3  if 1 < $n_{con}$ < 2 then
4      continue
5  else if no connections then
6      Connect to nearest origin node
7  else if >2 connection then
8      Keep connections to nearest origin nodes, remove all others
```

Figure 4.15: Repair function for user-nodes

These repair functions ensure all suppliers and users are connected in the network, and all hubs are being fed. This still leaves the possibility for floating hubs. If a hub is floating it will be connected to the nearest node in the layer above it (being a hub in a higher layer or a user).

There are still ways in which the repair function can be improved. Currently, the minimum and maximum number of connections is hard-coded in the repair function. The minimum number of connections for each type of node is 1, while the maximum number of connections for suppliers is 1, for hubs unlimited and for users 2, although this can be exceeded due to the floating hub repair function[18]. It can be desirable to make the minimum and maximum number of connections variable for each node and provide this as input, e.g. set the minimum number of connections to 2 for vital users. This should however also follow from the vulnerability evaluation when the minimum number of connections is not defined, as the algorithm will search for the least vulnerable and least expensive design. Another remark is that the feasibility definition is solely based on connectivity. A further improvement is to incorporate more elements in the feasibility check of a topology, i.e. checking whether the suppliers actually have the required capacity to sufficiently supply the connected users in an intact condition.

4.3.5. Determine routing

Based on the positions of and connections between components as captured in the chromosome for the outer GA, the problem for the routing optimization can be defined. This is the first step of the inner GA. This step also incorporates the possibility to create spatially redundant paths if this possibility is defined in the system input.

[18]The possibility of exceedance can be eradicated during the optimization process by declaring a constraint violation if the maximum numbers of connections is exceeded and skip routing optimization. In the current implementation it was decided to accept an exceedance of the maximum number of connections as it generates insight in possible improvements of the design which can locally be solved (i.e. when an effector often requires more than the maximum amount of connections, it can be a consideration to add a local distribution panel to accommodate this).

The routing setup starts with compilation of a connection list with all the connected system node pairs, as defined by x_{con} in the outer GA chromosome. As the location of each system node is defined by x_{pos} in the same chromosome, the start and end location of each connection is known. The connection list will also contain the k_{dis} possible path variations.

If a connection can be spatially redundant, the connection will be duplicated. The designer can define in the input the probability that a spatially redundant connection will be formed (P_{red}). This is achieved by adding dummy paths to the list of k_{dis} possible paths. For example, if $k_{dis} = 10$, and $P_{red} = 0.5$, 10 empty dummy paths will be added, each with a cost of 0. If these paths are chosen, it means the redundant connection is absent.

It can be debated whether spatially redundant connections are part of the topology, when it is seen as 2 separate connections between 2 nodes, or part of a routing solution, when one connection between 2 nodes is implemented by defining 2 separate paths. The decision to incorporate it in the routing optimization was mainly motivated by the ease of implementation: implementation in the topology would require a redefinition of the outer GA chromosome and would complicate the topology repair function. An implication of this decision will be discussed in Chapter 5, Section 5.1.1.

The result of the routing setup is the size of the variable part of the inner GA chromosome, x_{route}. This part contains the indices of the paths chosen for each connection. The initial population is determined randomly, with the lower and upper bounds defined by the number of possible paths each connection can have. The structure of the chromosome of the inner GA is shown in Figure 4.16.

$$\text{Chromosome inner GA} = \underbrace{\text{x}\,|\ldots|\,\text{x}}_{x_{route}}\;|\;\underbrace{\text{x}\,|\,\text{x}}_{\substack{\text{Obj.}\\\text{scores}}}\;|\;\underbrace{\text{x}}_{\text{Viol.}}\;|\;\underbrace{\text{x}}_{\text{Rank}}\;|\;\underbrace{\text{x}}_{\text{Crowd}}\;|\;\underbrace{\text{x}}_{\substack{\text{Age}\\\text{rte}}}$$

Figure 4.16: Structure of chromosome of inner GA

4.3.6. Simulate damage

With a combination of the chromosome of the outer and inner GA, a design is generated with system component positions, connections between system components and physical paths implementing these connections. The next step is to inflict damage and assess the effects. This subsection will discuss how damage is generated, and how the effects of damage are materialized on the design. The next Section, 4.3.7, will discuss how this is translated into a vulnerability score.

Damage case definition

The first part of damage simulation is defining the damage cases. The damages in the model are defined using a damage case list. The use of this list ensures that all designs are evaluated for the same damage cases, and that the calculations required to determine the damage cases only need to be conducted once. The pseudo-code to compile this list is shown in Figure 4.17, which will be discussed below in more detail.

```
Algorithm 6: Generate damage case list
1 Determine the hit probability for each space;
2 Determine the vulnerable spaces;
3 Determine the hit probability;
4 Compile the damage case list;
```

Figure 4.17: Generate damage case list

1. *Determine the hit probability for each space*. Based on the threat the correct predefined PDF will be selected. Initially 2 PDFs are implemented: one using a uniform hit probability and one using

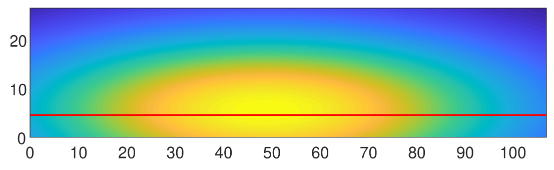
a joint PDF for the hit probability of each side. The hit probability for a space will therefore consist of 3 elements: the hit probability when the space is approached by the threat from starboard or port side (XZ-plane), bow or aft (YZ-plane) or the top or (theoretically) bottom (XY-plane). The hit probability will be determined by integrating the joint PDF over the geometric boundaries of the space, thereby taking the size of the space into account. This means each space is modelled as a cube. The hit probability distribution in each plane is assumed to follow a bivariate normal distribution based on the coordinates in that plane, meaning the PDF can be defined as [Dekking et al., 2005]:

$$\frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}}\exp\left(-\frac{1}{2}\frac{1}{(1-\rho^2)}Q(x,y)\right) \tag{4.2}$$

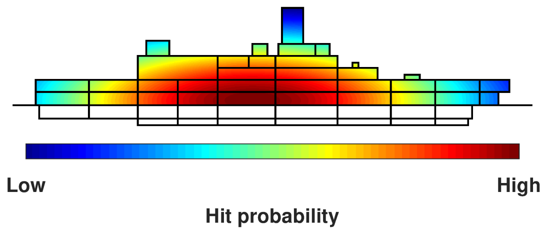
where

$$Q(x,y)=\left\{\left(\frac{x-\mu_X}{\sigma_X}\right)^2-2\rho\left(\frac{x-\mu_X}{\sigma_X}\right)\left(\frac{y-\mu_Y}{\sigma_Y}\right)+\left(\frac{y-\mu_Y}{\sigma_Y}\right)^2\right\} \tag{4.3}$$

and X and Y are the 2 dimensions of the plane (not necessarily the same X and Y as the axis in the coordinate system). The boundaries of the coordinates are defined by the ship coordinates. Values of σ and μ should normally follow from (classified) weapon data, including seeker logic. Figure 4.18 shows a PDF without and with the overlay of the ship, before integration over the space boundaries.



(a) PDF bounded by ship dimensions with waterline marked in red.



(b) PDF with ship overlay

Figure 4.18: Example Probability Density Function (PDF) for XZ-plane

2. *Determine the vulnerable spaces*. Not all spaces are susceptible to a hit. For most weapons only spaces at the outer hull or superstructure can be hit directly. This can be further reduced by taking the domain and trajectory of the threat into account: in general, air- or surface-fired weapons can only hit spaces above the waterline. For a sea-skimming missile, only the spaces located at 4 of the 6 threat directions are eligible to be hit: these are the spaces located at starboard, port, bow and aft of the ship. This helps in limiting the number of damage cases which needs to be evaluated and means the definition of additional spaces does not necessarily result in a significantly increased computational effort.

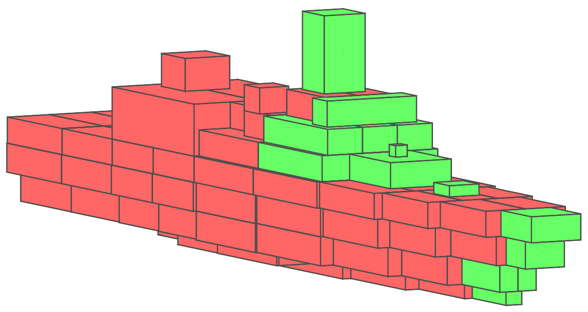
3. *Combining the first two items, the total hit probability can be determined for each of the 6 threat directions*. As the joint pdf is defined in a rectangular plane, and the ship silhouette is not rectangular, the sum of the hit probabilities of the spaces will not equal 1. Therefore the hit probabilities will be scaled to equal 1 for each threat direction. For the above mentioned sea-skimming missile, for each of the 4 threat directions the hit probabilities are known. To combine all these probabilities, the found hit probabilities will be scaled with the total vulnerable area of the spaces for that particular threat direction. This leads to the weight of the hit of a particular space, where the last scaling factor ensures that the sum of the weight of all hit cases equals 1 for all threat directions.

4. *Compile the damage case list*. The next step is to compile the actual damage case list. This list consist of all the damage cases, where a damage case is defined as the space which is hit, all spaces which are affected (this might include for example neighbouring spaces which cannot be hit directly) and the weight which is assigned to the damage case for determining the vulnerability score later on. To assess which spaces are affected, the damage adjacency matrix will be used, enabling the designer to incorporate protection as input for the model.

The process above describes how a damage case list for one hit with a specific threat is compiled. Predefining the damage case list which will be used for damage simulation has however another advantage: it offers the possibility to compile a list consisting of multiple threats or hits for varying damage extents. This means that the vulnerability can be assessed for a variety of threats and, when this score is used for optimization, the design is optimized for these threats instead of for one particular threat. This does however come with a cost: the evaluation for more hits and threats means that the number of cases will explode, and thereby less designs can be evaluated and generated in the same amount of time. This is especially the case when a damage case consists of multiple simultaneous damages, as all combinations of vulnerable compartments resulting in the number of damages should be evaluated.

There are some remarks which can be made about this method. Not all spaces are per definition affected and therefore evaluated. This would mean that the algorithm is likely to develop a preference to position components in these spaces. Although this is correct for the particular damage which is evaluated, it would mean all components are concentrated in a limited amount of spaces. This would mean that the resulting design becomes extremely vulnerable for threats where these spaces are actually susceptible to damage from other threats (e.g. due to internal causes such as fire). One of the ways to avoid this is to include all unaffected spaces in the damage case list, but assign a weight of 0. This means these spaces will not be taken into account when determining the vulnerability score, but the effects of unavailability of these spaces can be used to determine whether the residual capabilities are still sufficient. Another way is to include per definition all spaces in the damage hit list, with a uniform hit probability to reflect the probability that a space gets hit by e.g. a fire.

A second remark which can be made is about the calculation of the vulnerable areas and relative positions of the spaces. For spaces in the aft part of the ship, the vulnerable area and relative position is largely correct, as the spaces mostly resemble cubes. Moving forward to the bow, the actual spaces will increasingly start to differ from this cubic form and therefore exposed areas and positions will be harder to establish accurately: every space has a small part of the area in its YZ-plane exposed, but a significant part of the area is not exposed because it is obstructed by the space in front of it. Only spaces with significant projected areas when seen from the bow are defined as located at the bow, e.g. part of the superstructure (Figure 4.19). This means that the small areas are neglected. It should be noted that in this figure the front view exaggerates the effect of the neglected spaces, as the neglected areas seems to be the major part of the total frontal area. These red areas are however in majority areas of spaces which' side can't be hit anyway, as the side is obstructed by other spaces in front of it. For the level of detail in the early stage design process this simplification is deemed appropriate, as any differences will only have a small effect on the assigned weight to a damage hit case.


(a) 3D-view

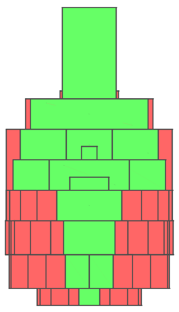

(b) Front view

Figure 4.19: Vulnerable areas taken into account for threat approaching from the bow in green, neglected areas in red.

The third remark is regarding the hit probability distribution. The chosen distributions suffice for the purpose of this thesis, as was the requirement for this part of the model. More realistic distributions will however add to the reliability of the generated solutions.

Damage effects

With all the possible damage cases known, the effect of each damage case on the system distribution network should be determined. The way this is done is mainly in line with the method as described in [Duchateau et al., 2018]. Based on x_{pos} and x_{route} it is determined which routing nodes hold which system component nodes and which specific paths (edges). Also a consolidated connection list is generated, where all unused spatially redundant paths are removed. Furthermore for all networks, $A_{net,intact}$ will be reconstructed based on x_{con}

The next step is to evaluate each damage case from the damage case list. For each damage case, the affected spaces will be evaluated. If all affected spaces are empty, the algorithm will set all systems to available and continue with the next damage case. If an affected space holds a path, the specific path will be removed from the connection list. If a space is hit which holds a system component, the action will be the same: per definition the space also holds all paths with the specific component involved, and all these paths are being removed from the connection list. Now the affected connections are known, and the damaged network adjacency matrices $A_{net,damaged}$ can be generated, removing all damaged connections from $A_{net,intact}$.

The intermediate step using a connection list before generation of $A_{net,damaged}$ is necessary to facilitate the spatially redundant paths. Introduction of these paths in the routing optimization mean that paths and connections are no longer the same. If for example one path is affected, but the other spatially redundant path is still intact, the connection is still intact and should therefore not be removed from $A_{net,intact}$. After all this is the purpose of a spatially redundant path.

With $A_{net,damaged}$ known for a specific damage case, the effect can be evaluated by the vulnerability assessment function.

4.3.7. Vulnerability assessment

For the actual vulnerability assessment the hurt-state percolation method will be used as discussed in Chapter 3. The implementation is an adapted form of the hurt-state percolation method as presented in [Duchateau et al., 2018]. The modifications concern minor modifications to improve the speed of the depth-first search and an improvement regarding the check of hub-hub connections. The implementation of the actual depth-first search will not be further discussed here, as there are no significant changes from the original algorithm. The output of the depth-first search is the availability of specified system nodes, in this model the effectors.

The next step is to translate the effector availability to capabilities and in the end the MOE. This translation will use the relations defined in the model input as discussed in Section 4.3.1. For each damage case the (sub)capability scores and MOE$_\text{d}$ will be determined. The total MOE[19] will be determined using a weighted average of the MOE$_\text{d}$ for each damage case. As the GA aims to minimize the objective functions, the vulnerability objective score will actually be calculated as a Measure of Ineffectiveness, or $1-MOE$ with N the number of damage cases, E the number of effectors and S the number of different (sub-)capabilities:

$$f_2 = 1 - \sum_{d=1}^{N} MOE_d \cdot w_d, \text{ with } MOE_d = \sum_{i=1}^{E}\sum_{j=1}^{S} a_{i,d} \cdot c_{i,j} \tag{4.4}$$

w_d = Weight of damage case d

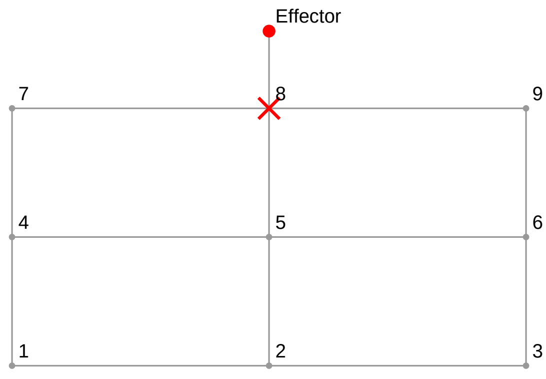
$a_{i,d}$ = Availability of effector i in damage case d

$c_{i,j}$ = Contribution of effector i to capability j

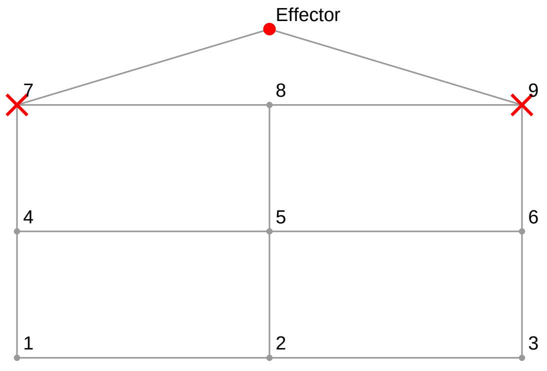
[19]The total MOE in damaged state will simply be called the MOE, as there can be no confusion in this research with the MOE in intact condition. When this confusion can occur, a distinction should be made. For example in [Goodfriend, 2015], the total MOE in damaged state is called the OMOV.

When comparing different system configurations, one should note that the factors used for determining f_2 should be equal. This means that the evaluated damage list should be the same, as the MOE is a weighted average based on the damage cases.

The last part of the vulnerability assessment is the degree of compliance of a design with the formulated RCM. A design is compliant with the requirements from the RCM, when the required level of capabilities are available in any defined damage case from the damage case list. In this case, a design is deemed feasible. For certain requirements however this will be impossible. This is the case when a capability is completely fulfilled by one particular effector, or when a capability needs to be available to the full extent. The damage case list will for most threats contain a case in which the effector itself is hit, thereby per definition violating the requirement in the RCM. This is also the case when the effector is only connected to one other node in the routing adjacency matrix. This is shown in Figure 4.20a, where a damage at node 8 (indicated by a red cross) will cause unavailability of the effector as well. Figure 4.20b shows the same principle for 2 hits, where 2 nodes are damaged and the effector can only be supplied through these nodes. This means that based on the damage size and number of damages, certain damage cases needs to be excluded from the RCM-compliance assessment.



(a) Single damage



(b) Double damage

Figure 4.20: Single points of failure due to constraining physical architecture

Even when these cases are excluded properly, experiments with 2 hit cases showed the model finds it difficult to generate a feasible design. This is because both the position, topology and routing needs to have the perfect balance to avoid creation of a SPF near the excluded nodes, if this is possible at all. One of the reasons is the fact that the maximum number of hub-user connections was manually determined to be 2 as discussed in Section 4.3.4. Therefore the hubs need to be co-located with the effectors, and the number of incoming connections to the hubs need to be 3 or more. The model showed that it is capable of improving the design, as the degree of infeasibility decreases with each generation. The generation of feasible designs will however take a significant amount of time, meaning there is less time for improvement of the feasible designs. Also the practical feasibility of a requirement where a capability always needs to be available after a hit is questionable. Therefore the percentage of damage cases for which the design needs to comply with the RCM, C_{RCM}, was introduced in Section 4.3.1. Another option is to deem the requirements set in the RCM too strict and decrease the level of capabilities required. The damage case percentage however gives the designer more flexibility and insight in the degree of residual capabilities for certain impacts.

When the design is compliant with the RCM given the C_{RCM}, the violation score $g = 0$. When the design is infeasible, a penalty will be awarded. This is based on the number of cases in which the RCM is violated and the impact category for which the RCM is violated. Violation of requirements for smaller impacts have a larger weight, as the design should be able to handle a smaller impact better than larger impacts. This gives the following constraint violation function, with $R_{cat,viol}$ the number of non-compliant damage cases for a certain category and w_{cat} the weight factor for that category:

$$g = \sum_{cat=1}^{n_{cat}} R_{cat,viol} \cdot w_{cat} \tag{4.5}$$

4.4. Feedback

In Chapter 3, the possibility of providing feedback to the outer GA has briefly been discussed for a more targeted optimization of the positions and topologies[20]. The rationale is that when it's known which factors are causing a vulnerability, it is more efficient to try to change that specific factor. The idea of implementation of this feedback mechanism was abandoned for several reasons.

First of all the problem remains whether it is possible to determine a source of vulnerability. Unavailability is seldom caused by the unavailability of a single system component or connection, as most system components and connections are at least functionally redundant. For a single network a more generalized approach could be used to look at layers of the network instead of specific nodes. It could be investigated which layer is failing often, for example the hub layer, and if there is a lack of redundancy in connections to other layers, connections could be added. This still leaves a lot of room for interpretation and requires definition of several cases and possible solutions. For more complex networks this is even more complicated. If nodes in a certain layer are failing often (e.g. CWPs in the CW-network), this might be because they are too close together, and the position might be changed. The unavailability of the CWP could however also be caused by a lack of power supply, meaning that the actual vulnerability is hidden in the electrical power network. This means that it is even complicated to pinpoint the exact network that should be improved to reduce vulnerability.

A feedback mechanism in which the source of vulnerability cannot be pinpointed with a certain degree of confidence might therefore not lead to better solutions. A bad implementation might even lead to *worse* designs than a model in which the feedback is not implemented, as it misdirects the GA in which parts of a design should be modified. Therefore the specific parts of the design which are actually causing the problem might be preserved. This problem could of course be circumvented by also changing other aspects of the design, but this would negate the potential gains of the feedback mechanism.

The third reason for abandoning the idea of a feedback mechanism is the amount of time available for the research. One might be able to solve the problems mentioned above, but this would take a considerable amount of time. Even when the aforementioned issues are solved, implementation in the model would require modification of the GA to be able to deal with the feedback. The time available for the research is also necessary for implementation of the model and analysis of generated solutions to be able to reflect on the design rules. This doesn't mean that the idea of a feedback mechanism should be abandoned at all, but if this idea is pursued it should be given the attention this mechanism requires to develop it well, as it might otherwise be counter-productive.

4.5. Verification and validation

For verification of the implementation several tests were conducted of the previously described functions. These will be discussed below. For validation the validation square as described in [Pedersen et al., 2000] will be used.

4.5.1. Verification

For verification[21] the various functions were tested in several ways, and to different extents. The input functions for example are verified, when the defined relations in the Excel-sheets are reflected in the variables in Matlab®, while the initial population (determine initial topologies and positions) is correct when it consists of the right number of genes and the values are within the prescribed bounds. Verification of the determination of k-possible paths was conducted by simply looking at some extremes (paths far apart, on both sides of the ship) and the results and consequences were already discussed in Section 4.3.2. The verification of the topology-repair function, damage simulation and vulnerability assessment required more effort and will be discussed below.

[20] The GA is providing feedback by means of evaluating designs, selection and modification (cross-over and mutation) based on these scores. In the context of this section, feedback is considered separate direction given to the outer GA on which exact part of the chromosome needs to be modified (which network, connections, positions, etc.), using the secondary output of the vulnerability analysis.

[21] In this research verification is considered the correct implementation of the functions as described in the mathematical model, i.e. code verification in some literature.

Repair function

For verification of the repair function, two extreme cases were considered: a completely disconnected network and a fully connected network as input. As mentioned earlier, the decisions which nodes to connect are based on the proximity of the nodes. The topology is placed in a constraining physical architecture where all nodes of the same type and network are in the same space. The repair function will be tested to its fullest extent in this case, as floating hubs will occur and multiple nodes are equally suitable to connect to.

The first extreme is a case with 3 networks, in which the input consists of no connections at all (all elements of x_{con} are 0). The expected output was a repaired network, in which all suppliers and end-users were connected and no floating hubs were present. The resulting repaired topology is shown in Figure 4.21.

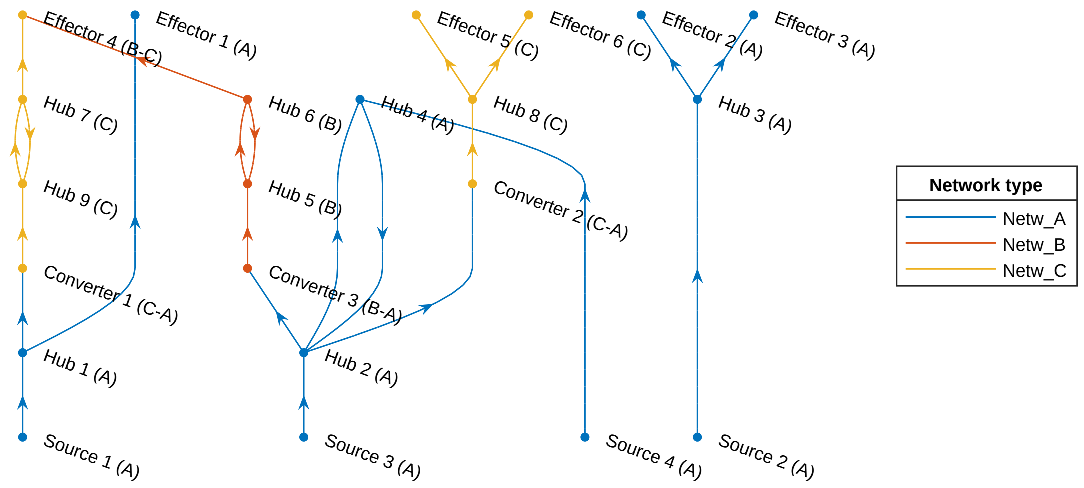


Figure 4.21: Repaired topology of unconnected network

As can be seen the topology is repaired properly. All suppliers have one outgoing connection, hubs are connecting suppliers or hubs to users or other hubs and all converters and effectors are receiving the required types of resources.

For the second case the same system is considered, but the input consists of a fully connected network (all elements of x_{con} are 1). The repair function should remove connections where the maximum amount of connections is exceeded. The resulting repaired topology is shown in Figure 4.22.

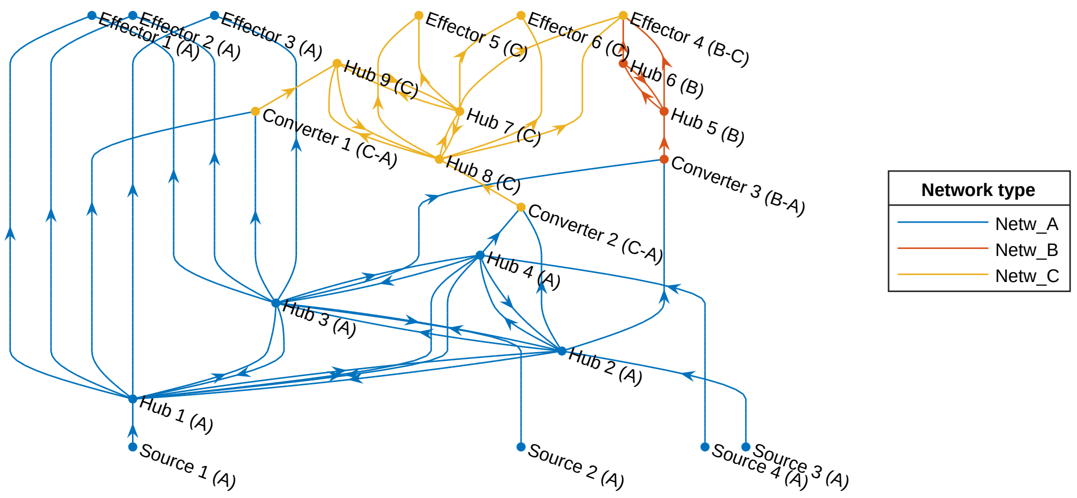


Figure 4.22: Repaired topology of fully connected network

Again the topology is repaired as expected. All sources have only one outgoing connection, and the users (effectors and converters) all have the maximum number of possible connections (2), while the hub layers remain fully connected.

Based on these two cases the repair function can be assumed to work in intermediate cases as well. To build further confidence this is actually the case, 1000 random topologies for the system were generated and checked with varying component positions. All these topologies satisfied the conditions for a feasible intact topology.

Damage simulation

Damage simulation consisted of 2 parts: damage generation (generating the damage case list) and infliction of damage (construction of $A_{net,damaged}$).

A first check for the damage case list is to check whether the sum of the weights equal 1. This test is always conducted as the last step of the damage case list generation. A second check is to see whether the affected spaces are as expected. A third check is to see whether spaces which have sides that can be hit from multiple threat directions (e.g. a space located starboard-aft has a higher hit probability than adjacent spaces) are awarded a higher weight then their neighbours with 1 threat direction.

These checks have been conducted for a grid of 4x10x4 nodes. In this specific case the spaces at the lowest layer in vertical direction were defined as invulnerable (under the waterline), and a hit would affect all neighbouring (i.e. directly adjacent) spaces. Figure 4.23 shows the physical layout with the resulting weights (times 100 for easier distinction between values) of the hit at that specific node. As can be seen, only the vulnerable spaces can be hit and are therefore assigned a weight. The values also reflect the PDF, with the highest hit probability just above the waterline amidships and at the centreline.

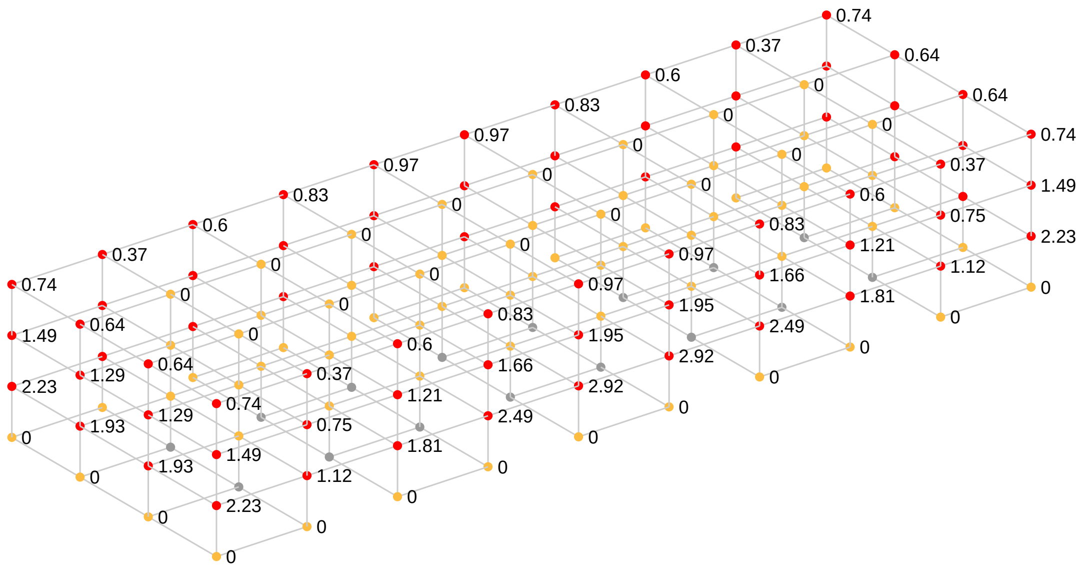
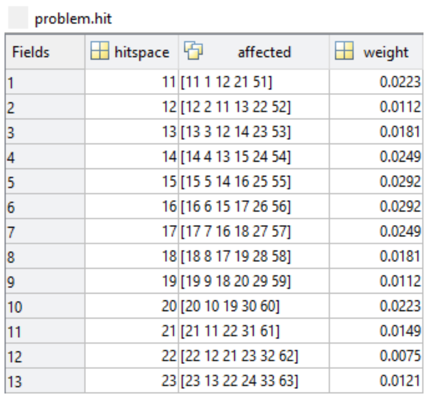


Figure 4.23: Weight per hit spaces. Only values of the outer shell are shown for readability. Red nodes can be hit directly, orange nodes can be affected as a result of a hit, grey nodes can neither be hit nor affected.

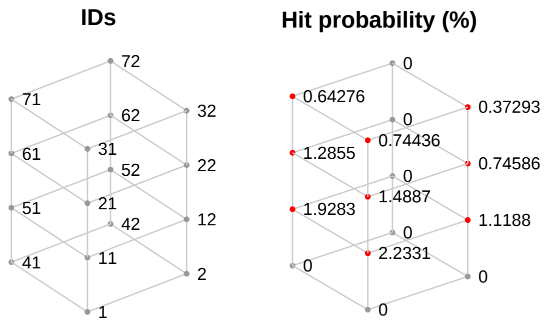
Figure 4.24a shows part of the resulting damage case list, and Figure 4.24b the starboard-aft corner of the design. The damage case list reflects that indeed all neighbouring nodes are affected, including the hit node itself. The corner spaces show the expected weights are higher for the corner spaces then its neighbours, as these spaces are vulnerable for hits from 2 threat directions. The weights for these spaces are also lower than the sum of the weight of both neighbouring spaces indicating that the increase is the result of the summation, and the individual hit probabilities are still following the PDF.



problem.hit

Fields	hitspace	affected	weight
1	11	[11 1 12 21 51]	0.0223
2	12	[12 2 11 13 22 52]	0.0112
3	13	[13 3 12 14 23 53]	0.0181
4	14	[14 4 13 15 24 54]	0.0249
5	15	[15 5 14 16 25 55]	0.0292
6	16	[16 6 15 17 26 56]	0.0292
7	17	[17 7 16 18 27 57]	0.0249
8	18	[18 8 17 19 28 58]	0.0181
9	19	[19 9 18 20 29 59]	0.0112
10	20	[20 10 19 30 60]	0.0223
11	21	[21 11 22 31 61]	0.0149
12	22	[22 12 21 23 32 62]	0.0075
13	23	[23 13 22 24 33 63]	0.0121

(a) Part of the damage case list



(b) Weight per hit spaces (corner)

Figure 4.24: Damage case list and weight of corner spaces

The damage effects should be reflected in $A_{net,damaged}$, which will be used for the depth-first search during vulnerability assessment. The only way to verify this is by manually checking the $A_{net,damaged}$ for each damage case, based on the chosen positions, topology and routing and damage case list. This has been done for two separate cases, being a 4x4x10 grid with a single network consisting of 14 nodes, and a notional OPV as used in [Duchateau et al., 2018] with 3 networks consisting of a total of 22 system nodes. Both were tested with a damage case list where 1 space was hit and the surrounding spaces are also affected. In all cases the appropriate edges were removed.

Vulnerability assessment

Together with the damage effect verification, the proper working of the vulnerability assessment was verified. This verification was conducted through the following steps:

1. Determine which edges in the topologies are affected.
2. Plot the topologies to check whether the edges are actually removed (last step of the verification of damage infliction as previously discussed).
3. Check which effectors are still connected (for all networks).
4. Check whether the vulnerability assessment calculated the same availability.
5. Manually check the calculated capability score, based on the effector availability and capability contributions.
6. Manually check whether RCM calculations are correct.

This was done for the same two cases as for verification of damage simulation. Initial verification showed handling of hub-hub relations during vulnerability assessment required a small modification, as 2 users connected through 2 hubs (so a user-hub-hub-user connection) was resulting in the availability of both users. After the modification all damage cases were correctly assessed and led to a vulnerability score and constraint score as expected.

4.5.2. Validation

The implemented model is deemed validated when the original problem can actually be solved by the model. As mentioned in [Pedersen et al., 2000], validation should be 'formal, rigorous and quantitative'. While this works for certain scientific areas, it it more problematic for design models as usefulness is often based on subjective and contextual rules, i.e. not absolute. This was already noticed in Chapter 3 when discussing the requirements. One of those requirements was that the model should be 'sufficiently fast for use in the early design stage', but the quantification of 'sufficiently fast' is rather subjective. The proposed validation approach by Pedersen for these kind of models, the validation square, will be used to validate the model. The validation square consists of structural validation (effectiveness) and performance validation (efficiency), both consisting of 3 steps. Structural validation will be discussed in this chapter. Performance validation will be conducted after the model is used for the design rule evaluation in the next chapter, as it concerns an evaluation of the generated solutions.

Structural validation consists of the following aspects:

- *Accepting the construct's validity*. A first step in structural validation is to build confidence in the individual parts used for the model, in this case the previously discussed functions. This is particularly valuable for the functions which are open for interpretation. In the implemented model, the determination of the k-shortest paths, topology repair, damage simulation and vulnerability assessment are deemed the functions open for interpretation, as the choice of which method to use for implementation is influencing the quality of the generated solutions of the model. Most of these functions are however based on well-known and widely accepted methods:
  - k-Shortest paths. The determination of k-shortest paths by the use of Yen's algorithm as proposed in [Yen, 1971] is widely accepted as a method to determine alternative paths. The novel approaches proposed by [Chondrogiannis et al., 2020] are less used, as well as the rough implementation used in the model. Section 4.3.2 however discussed the validity and the consequences of the used approach which is suitable for this specific model for the early design stage.
  - Topology repair. A repair function is commonly used to enable proper functioning of a GA and to increase computational efficiency of the model. In [van Leeuwen, 2017] and [de Vos, 2018] a repair function is used for repairing topologies. Although the implementation in this model is slightly different and a more generic approach (i.e. connecting nodes to other layers) the same principles as used in [de Vos, 2018] are applied: guaranteeing connectivity of users and suppliers, defining allowed number of connections, etc.
  - Damage simulation. Different methods for damage simulation have been discussed in the literature review, especially with regard to the definition of damage volumes. For defining hit probabilities, a relative simple method is used based on common probability theory. Whether the applied PDF is realistic can be debated and was in fact declared out-of-scope in the problem definition. Using Gaussian hit distributions is however not unique, as demonstrated in [Goodfriend, 2015].
  - Vulnerability assessment. The choice for the vulnerability assessment method was discussed during the literature review and solution generation. Besides this, percolation theory is a widely accepted method to assess vulnerability of networks by removing edges and nodes.

  All these functions are required in the wider optimization method. The choice for a nested GA has been discussed at the beginning of this chapter. For the decision of which particular GA to use, one of the requirements was in fact it's reputation and previous application to these types of problems. This increases the confidence that the chosen optimization method should be able to generate useful solutions.

- *Accepting method consistency*. This aspect is required to build confidence in the way in which the individual functions are put together. This validation step is normally presented as a flowchart with the information flow.. A flow chart was actually already presented in Chapter 3, with the visualisation of the mathematical model. The information required and generated per function was discussed at length in Section 4.3. Where assumptions were required, these were discussed and justified (e.g. the number of connections per node for the topology repair function), including possible consequences for the generated solutions.

- *Accepting the example problems*. The third factor is acceptance of the example problems which will be used for performance validation. In this research these example problems are also the testcases which will be used for design rule evaluation. The exact testcases will be discussed in the next chapter. First of all the example problems need to be similar for problems to which the separate functions are applied. As argued earlier, GAs are often used for optimization in ship design and distributed system design problems. The testcases will combine problems which are normally optimized in the same way (e.g. topology optimization in [de Vos, 2018], routing optimization in [Duchateau et al., 2018], etc.), but now combining multiple design variables all at once. The testcases are therefore similar to the problems for which the individual functions are developed. Secondly, the testcases need to resemble the actual problem for which the model is intended. This will be achieved by developing the testcases in such a way that it resembles actual design problems with regard to vulnerability; e.g. one variation in the testcases will be the damage size and number of damages, while another variation is the complexity of the networks to be evaluated (e.g. a single network vs. a network with multiple layers). The capability of the model to find appropriate solutions for these type of problems will show that the model is actually able to guide the design effort using the vulnerability of designs, and therefore the testcases are appropriate to validate the model.

4.6. Conclusion

This chapter showed the way in which the mathematical model presented in Chapter 3 is implemented, the choices that were made and the consequences for the generated solutions by the model. The first part of the research objective (how can vulnerability analysis be used to guide the early stage design effort in order to increase survivability) is partially answered by this implementation. However in order to be able to formally answer this part, also in relation to the requirements for the model set in Chapter 3, testcases are required for the last part of the validation of the model. These testcases are also required to achieve the second part of the research objective (what is the influence on contemporary vulnerability reduction measures). The last part of this chapter provided a sufficient level of confidence that the testcases can be successfully conducted.

5 Design rule evaluation and model validation

This chapter discusses the testcases which were conducted with the developed model. First the test-case design and purpose of the testcases will be discussed. Thereafter the generated solutions will be examined and used to reflect on the vulnerability reduction design rules. In the last part of this chapter the performance validation of the model will be discussed.

The purpose of the testcases which were conducted is twofold. First of all the generated solutions will be analysed and compared to the design rules as discussed in Chapter 2. The second purpose is that, based on the outcomes, the performance of the model can be compared to the requirements as set in Chapter 3 in order to validate the model.

5.1. Testplan

Based on the design rules, problem definition and the research objective a testplan is developed. The purpose of the testplan is to be able to achieve the research objective with a limited number of testcases, as the available computational capabilities are limited.

5.1.1. Input and variations

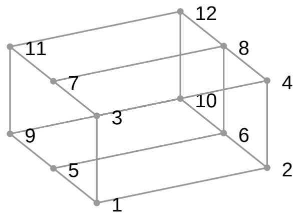
There are several input variables as discussed in Chapter 4 which need to be defined and can be varied for the separate testcases. To establish the requirements for the input, the design rules were reviewed and the requirements were drafted. This is shown in Appendix A. As shown in the table, the design rules constitute requirements for the system and physical input. Besides this, the assumptions for the design rules (single hit resulting in a large damage) should be tested, which requires a variation in damage size and number of damages. The input used for the testcases will be discussed below. This input is visualised in Appendix B. Figures regarding input are only shown in this chapter when this is required for readability or illustrative purposes.

Physical input

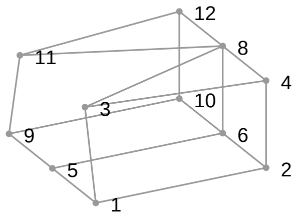
The physical input is visualised in Appendix B, Section B.1. Based on the requirements no variation of the physical constraining architecture is required. There are however some requirements for the physical architecture as formulated in Appendix A. The starting point for the physical input is the notional OPV-model which was also used in [Duchateau et al., 2018], [Habben Jansen et al., 2019b] and [Habben Jansen et al., 2019a]. This ship model was modified as follows:

- *Addition of layer in Y-direction*. The original ship model consisted of a maximum of 2 compartments in the Y-direction. This is sufficient to test separation of sources and paths in the XZ-plane for zone-deck sized damages. For the testcases, additional spaces need to be added in the Y-direction for 2 reasons. First of all it allows to examine separation in the third dimension (i.e. port/starboard separation, or concentration in the midplane). Secondly, it allows us to test whether a design rule (e.g. source separation) also applies when the assumption of one large damage is not fulfilled (e.g. does the rule also apply in case the damage consist of 2 small damages). Where the total width of the original hull exceeds 8 meters, an additional space is added in the midplane. The spaces on both sides are scaled down in this case, so that the total width remains the same.

- *Hangar*. The hangar is split in 2 separate spaces (port and starboard). The reason for this is that the hangar can be used to route paths via port or starboard side. Splitting the hangar in 3 is however not beneficial: it is not possible to route paths from beneath the hangar, through the centre of the hangar as this space needs to be empty. Addition of a centreplane with only routing possibilities to port, starboard and forward simply adds a node (node 7 in Figure 5.1a) through which all paths from the hangar sides are routed. Without this node the routing possibilities are unchanged as shown in Figure 5.1b.



(a) Routing adjacency with hangar split in 3 spaces (nodes 3, 7 and 11).



(b) Routing adjacency with hangar split in 2 spaces (nodes 3 and 11).

Figure 5.1: Routing possibilities for the hangar.

- *Close-In Weapon System (CIWS)*. A space is added on the topside to facilitate an additional effector.

For the routing adjacency network, the edges from and to spaces which are normally used as tanks or exhausts are removed. In the damage adjacency matrix, one blast bulkhead is modelled amidships, between spaces which are commonly used as engine rooms. The reason for this is that one of the system configurations consists of 4 large power suppliers (e.g. traditional Diesel Engines (DEs)). It would be unrealistic to distribute them through the ship due to their size and weight, and based on the design rules protection should be applied to prevent unavailability due to a single hit. Since the same physical architecture is applied for all testcases, this won't influence comparison of the testcases.

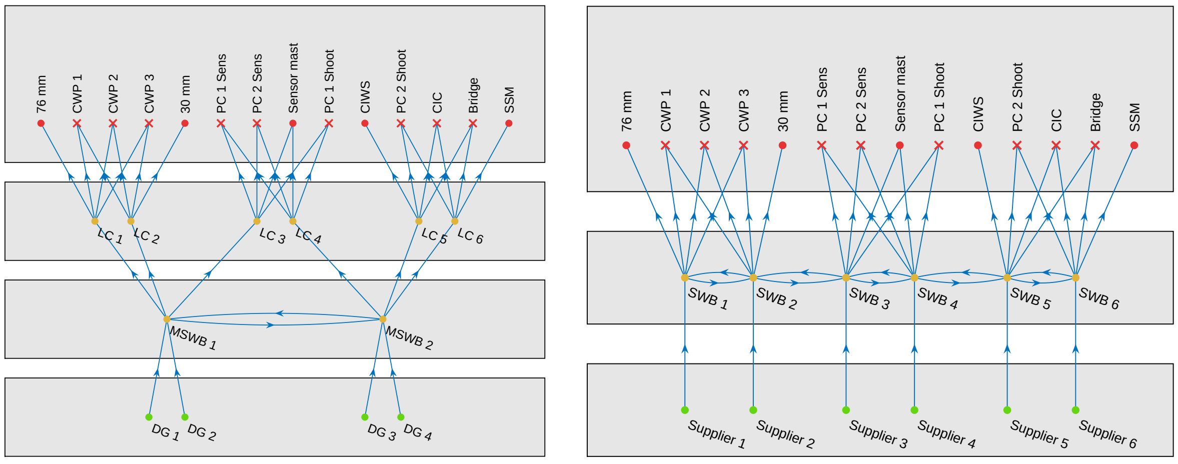
System input

The system input is visualised in Appendix B, Section B.2. Based on the requirements, 3 separate system configurations are defined. All configurations will have the same effectors, but the underlying distribution networks are varied. The effectors consist of a 76 mm gun, 30 mm gun, a sensor mast, a CIWS and Surface-to-Surface Missile (SSM).

1. *Configuration A* consists of a single, traditional 440V power distribution network as shown in Figure 5.2a. The network has 4 suppliers with fixed positions (DEs), a hub layer consisting of 2 hubs (MSWBs), a hub layer consisting of 6 hubs (LCs), 5 effectors and 7 other users. The MSWBs can connect to the suppliers, each other and the LCs, while the LCs can connect only to the MSWBs and users. In should be noted that in this configuration unavailability of the other users (i.e. users which are not effectors) will not influence the vulnerability score. They are however included to be able to compare the generated solutions to system configurations where they are influencing the vulnerability score (configuration C).

2. *Configuration B* consists of a single, decentralized 440V power distribution network as shown in Figure 5.2b. The network has 6 smaller suppliers which' positions are variable (e.g. Fuel Cells (FCs)), a hub layer consisting of 6 SWBs and the same users and effectors as in configuration

A. This configuration can be compared with configuration A, thereby enabling the evaluation of design rule 1 (avoidance of central distribution systems).



(a) System configuration A

(b) System configuration B

Figure 5.2: System configurations variations (indicating all types of possible connections, but not showing all actual possible connections).

3. *Configuration C* consists of 4 interdependent distribution networks. Due to readability the reader is referred to Appendix B, Section B.2.3 and Appendix D, Section D.5 for a visualisation of the networks. The 440V power distribution network is the same as for configuration A. The CW network consists of 3 CWPs and 6 hubs (main supply lines). The number of hubs is based on the number of zones that could be distinguished in the physical architecture. The third network is the sensor data network (DataSens), consisting of the sensor mast as supplier, 2 hubs (PC-rooms) and 2 users (the Combat Information Centre (CIC) and bridge) which can convert the data to a Fire Control Solution (FCS). The FCS-data is distributed to the effectors requiring this data through the DataShoot-network. These use the same PC-rooms as hubs, and the 76 mm gun, 30 mm gun and SSM are the users. As the PC-rooms have a role as hub in both data-networks, they require to be modelled as separate system nodes. The distinction will only be artificial, as the positions of the separate nodes will be repaired so they are always the same (i.e. PC1 Sens and PC1 Shoot will always be co-located, as will PC2 Sens and PC2 Shoot). Configuration C will be used to evaluate rule 10 (combine paths of distribution systems for essential capabilities). Also the degree of freedom for positioning certain components will likely reduce as there are more dependencies in the network (i.e. the unavailability of the 'other users' in configuration A and B now could influence the vulnerability score). Inclusion of hubs which could all connect to each other in the CW-network also facilitates the evaluation of rule 10+ (ring-shaped configurations are preferred).

Capability and RCM input

The RCM and capability input is visualised in Appendix B, Section B.3. The capability and RCM matrix are the same for all testcases. The requirements for these two factors are more driven by validation of the model than evaluation of the design rules.

The capability matrix should reflect at least one effector contributing to multiple capabilities, as this is the distinction between design effort driven by availability of an effector or by capabilities. For some damage cases, the RCM should be used to see whether non-compliance with the RCM (i.e. generation of an infeasible design) will influence the design generation. The RCM-requirement in these testcases essentially means that the CIWS should be available after 2 small simultaneous damages, and there are no requirements for 1 large damage.

Variable component position ranges input

The input for the variable position ranges is visualised in Appendix B, Section B.4. The variable position ranges are defined per layer of the respective networks. Limitations are based on technical feasibility considerations only. That is ranges are not limited to force components in a certain part of the ship (e.g. each zone with a dedicated LC) based on preference, but on practical feasibility (e.g. a CWP will not be positioned in the superstructure). A visual representation for position ranges of the different nodes per system network is attached in Appendix B, Section B.4. The initial population will be based on an initial separation as described in Section 4.3.3 of Chapter 4 to increase the probability of starting with feasible designs.

Threat input

The threat input consists of 2 types of damage: a large hit which will damage the zone-deck compartment and its neighbouring compartments (Figure 5.3a), and a small hit which will damage the hit space and the space in the extension of the hit space (Figure 5.3b). Based on these damage sizes, 3 damage case lists are compiled.

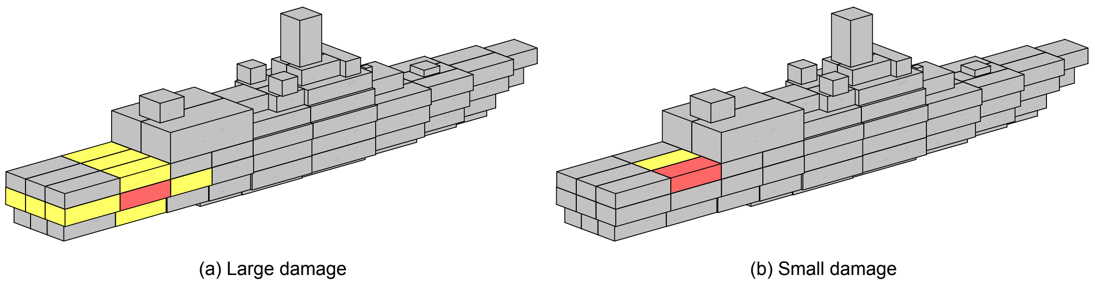


Figure 5.3: Different damage sizes. Spaces in red are directly hit, spaces in yellow are affected as a result of the hit space and therefore also damaged.

Damage case list A represents all damage cases for a hit by a SSM resulting in one large damage. This means the damage case list consists of 35 damage cases.

Damage case list B represents all damage cases for two simultaneous hits by a RPG-type of threat, resulting in two separate damages. As the ship network consists of 112 nodes, this results in a total of $\binom{112}{2} = 6216$ damage cases. This number neglects the two hit cases resulting in one damage (i.e. the same space is hit twice), and can be further reduced by systematically eliminating cases where damage is irrelevant or impossible. This process is shortly described here and visually represented in Appendix C. Spaces without system nodes or edges (i.e. unconnected nodes in the routing adjacency matrix) can be neglected and spaces that cannot be directly hit can be neglected. Hits which result in the same damage case can be eliminated by removing duplicate damage cases, where all weights are summed to the remaining unique damage case. The latter are often cases involving a hit in the bow, where a hit from port or starboard results in the same damage as the same 2 spaces are affected due to absence of a third space in the centreplane. This results in a reduction of the damage case list with almost 75%, to a list with 1590 unique damage cases. This could be further reduced based on operational or design considerations. Such considerations are e.g. only 2 hit cases on the same sides are likely and therefore relevant, based on Damage Control (DC) capacity of the ship only 2 hit cases resulting in one damage in the foreship and one damage in the stern are relevant, or based on watertight subdivision, floodable length and construction only two simultaneous damages further apart are relevant. For the testcases this reduction is not pursued, as it will limit the insights to more specific cases. Testcases where only the damage case list is varied will test the underlying assumption of the design rules.

Damage case list C represents a mixed threat and contains all damage cases present in list A and B. Part of the added value of the model can be demonstrated with a testcase using damage case list C, as this shows the difference in optimizing for separate threats or a set of threats. On the other hand, if there is no difference in the generated solutions for damage case list A, B and C then the method might have limited or no added value at all. The list is generated by simply adding both damage case list A and B. The weights of the separaate damage cases need to be rescaled , as simply adding the lists result in a total weight of 2. For the mixed threat list, the event of 1 large damage is deemed equally important as 2 simultaneous small damages, and therefore the weights are multiplied by 0.5.

All damage case lists use the same hit PDF for the physical architecture. The settings of the PDF and resulting weights per space are qualitatively shown in Appendix B, Section B.5.1.

Other input

There is one other input parameter which needs to be defined for the testcases which is the use of spatially redundant paths. Initially possibilities were defined to allow the algorithm to generate spatially redundant paths as part of the routing problem. During the first testcases however errors occurred due to the use of the parallel computing toolbox in Matlab®. The cause of the error could not be isolated and fixed, as it was not reproducible (sometimes the error occurred after 1 topology/position generation, and sometimes after 15 topology/position generations). The option for spatially redundant paths was therefore disabled for the testcases. This enabled the use of parallel computing, reducing the calculation time on average in the used setup by a factor of 3. For the results of the testcases disabling the option is not a significant problem, as there are still possibilities for functionally redundant paths and the number of nodes which could form spatially redundant paths was already limited (only certain nodes in the data networks, and the MSWBs in system configuration A and C).

5.1.2. Testcase definition

The previously described input and variations are combined in distinct testcases as show in Table 5.1 (a more elaborate version is enclosed in Appendix A, Table A.3). With these testcases all the desired design rules can be evaluated and assumptions can be tested. The first 3 testcases are more theoretical as they only consider 1 network. The main purpose of these cases is to test the assumptions of the specified design rules and establish a baseline for the remaining design rules and testcases. Testcase 4 examines usability of the design rules and the model for an alternative system configuration. Testcase 5 examines a more complicated and realistic set of distributed networks.

Testcase	*System configuration*	*Damage case list*
1	A - single 440V network - traditional	1 large damage (list A)
2	A - single 440V network - traditional	2 small damages (list B)
3	A - single 440V network - traditional	Mixed damages (list A+B)
4	B - single 440V network - decentralized	1 large damage (list A)
5	C - multi-network	1 large damage (list A)

Table 5.1: Variations in testcases

5.1.3. Optimization settings

The final input required for the testcases are settings of the NSGA-II optimization algorithm (Table 5.2). Small testcases were conducted to examine and determine the influence of the population and generation sizes, cross-over and mutation probabilities. The used cross-over and mutation distribution indices are default settings which are commonly used for the NSGA-II [Duchateau, 2016], [Deb and Agrawal, 1995]

		Outer GA	*Inner GA*
Cross-over probability	p_c	1.0	1.0
Cross-over distribution index	η_c	2	2
Mutation probability	p_m	0.1	0.1
Mutation distribution index	η_m	5	5

Table 5.2: NSGA-II-settings for cross-over and mutation

The number of generations n_{gen} and population size n_{pop} for the outer and inner GA are limited by the time available and the damage cases. However these parameters need to be large enough to generate diverse solutions which are converging to Pareto-optimal solutions. To determine these settings, for each testcase a small population is generated and evaluated, and the time required per design is used to determine the final GA settings (Table 5.3) and resulting calculation times.

		Testcase 1	*Testcase 2*	*Testcase 3*	*Testcase 4*	*Testcase 5*
	n_{gen}	50	25	25	50	50
Outer	n_{pop}	60	30	30	60	60
	Variations	3.060	780	780	3.060	3.060
	n_{gen}	30	20	20	30	30
Inner	n_{pop}	30	20	20	30	30
	Variations	930	420	420	930	930
Nr. of designs evaluated		2.845.800	327.600	327.600	2.845.800	2.845.800

Table 5.3: NSGA-II settings for number of generations and population size per testcase.

5.2. Analysis of testcase results

Before going into a detailed analysis for each of the evaluated design rules, the general results of the testcases will be discussed followed by the methodology used for evaluation of the design rules.

5.2.1. Population evaluation

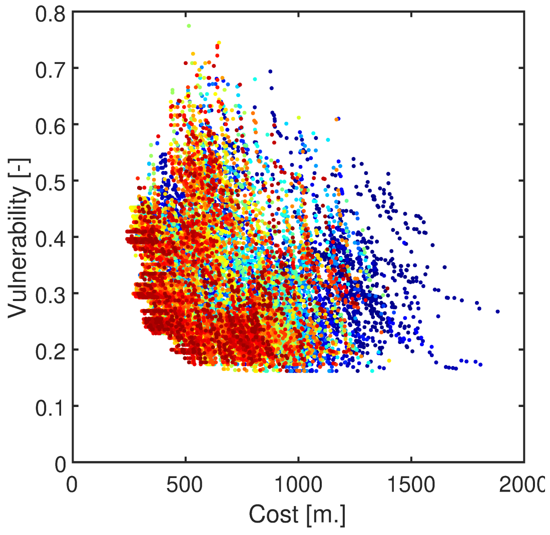
First of all Table 5.4 shows the number of unique designs in the generated population in the outer optimization loop. This is an indication of the diversity of the created populations for each testcase.

		Testcase 1	*Testcase 2*	*Testcase 3*	*Testcase 4*	*Testcase 5*
	Total nr. of variations	3060	780	780	3060	3060
Outer	*Unique combinations*	3004	770	774	3016	3044
	Unique topologies	2065	641	629	2555	2710
	Unique positions	2968	759	765	2981	3023
Time required[22]		9.5 hrs.	54 hrs.	57 hrs.	11 hrs.	34 hrs.

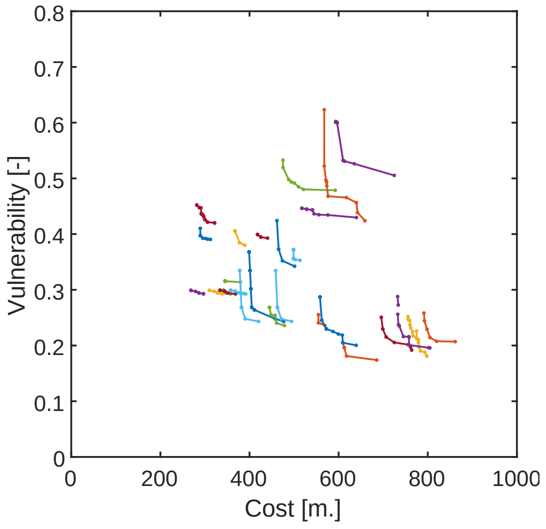
Table 5.4: Diversity of generated solutions and required generation time per testcase.

Figure 5.4a shows a plot of the population of each generation of the outer optimization (topology and variable positions) after the routing optimization for testcase 1 (traditional power network with 1 large damage). This means that for each topology/position the Pareto-front of the routing optimization is plotted. The colors indicate which generation the designs were generated; blue indicates the first generation, whereas red is the latest generation (50th generation for testcase 1).

[22]This is the time required to conduct the testcases on a Quad-core laptop given the input, meaning the calculation of k-possible paths based on the physical input is excluded. Time required to generate these paths is an additional (one-time) 3.5 hours for physical input of the considered testcases.



(a) Generation plot of outer optimization for testcase 1



(b) Routing Pareto-fronts of 30 randomly selected topology/position combinations

Figure 5.4: Results testcase 1

Several observations can be made based on the plots. First of all the Pareto-front looks well developed, which means there are no significant steps which can be identified in the front. Secondly a significant amount of the designs of the last 10 generations are located close to the Pareto-front, indicating the front has been explored quite well. Thirdly, it stands out that already in the first generations including the initial population solutions are found with a minimum vulnerability score. Finally, when the plot was built during the generations it could be observed that from the generation plot the routing Pareto-fronts can clearly be identified. This is illustrated in Figure 5.4b where the Pareto-fronts are plotted for 30 randomly chosen topology/position combinations. This plot clearly shows that the routing optimization is a local optimization, as the lower bounds for the costs and vulnerability are defined by the topology and positions of the components.

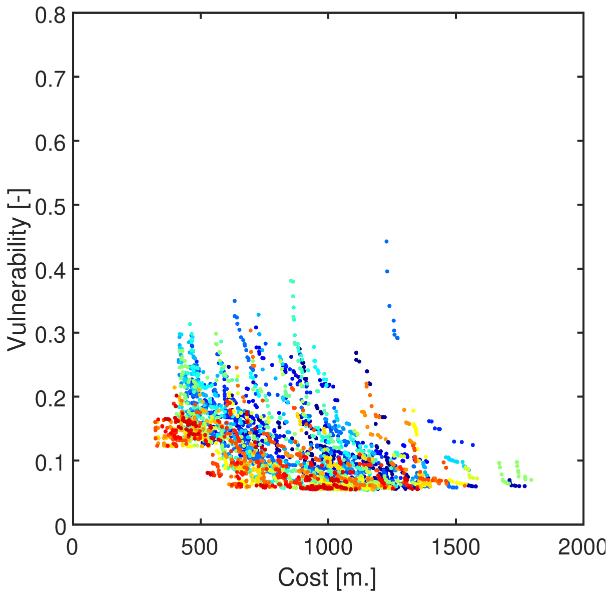


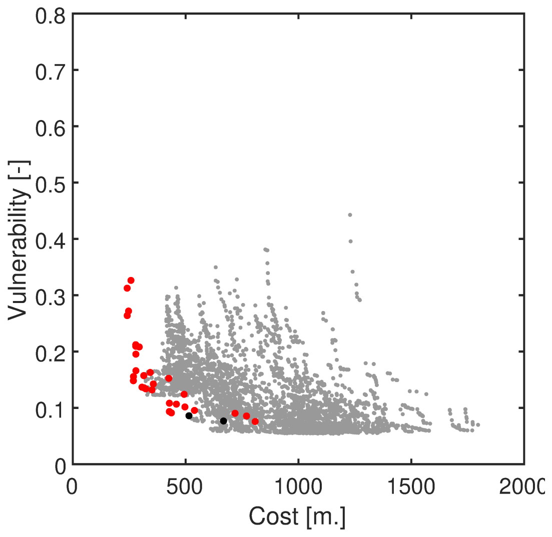
Figure 5.5: Generation plot of outer optimization for testcase 2

Figure 5.5 shows the generation plot for testcase 2 (traditional power network with 2 small simultaneous damages). This plot differs from the plot of testcase 1 in several ways. First of all it should be noted that the vulnerability scores of the two cases cannot be compared, as different damage case lists are used (see Chapter 4, Section 4.3.7). The Pareto-front of Figure 5.5 shows that the front is less researched when compared to testcase 1. This is due to the lower number of generations and population size, and indicates that there might be further room for improvement. This Pareto-front does show a step; at a total path length of approximately 500 meters, the vulnerability score rapidly increases for a further decrease in costs. Analysis of this step shows that system design of the less expensive and more vulnerable designs on the front is separated in two islands, without any cross-connection. All less vulnerable designs on the front do have at least one cross-connection. The less vulnerable designs also have an increasing number of hub-effector connections. The vulnerable designs left of the step only have a redundant connection to the CIWS, while the number of redundant connections increases right of the step with decreasing vulnerability. To check whether this step is purely a result of the lower population size and generations for this testcase, additional generations are performed using the Pareto-front of testcase 2 as a starting point. Although additional designs are found with a cost reduction of approximately 10%, the step in the Pareto-front remains present, for the same reasons. The underlying reason for this is further discussed in Section 5.2.3

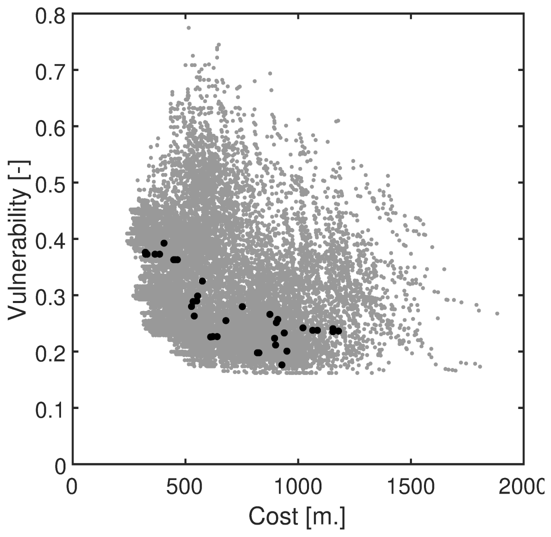
Testcase 3 is a traditional power network with a combination of the damage lists of both testcase 1 and 2 (1 large or 2 simultaneous small damages). Before discussing the results of testcase 3, it is interesting to see whether this testcase is actually necessary. If the Pareto-optimal designs of testcase 1 (1 large damage) are also the optimal designs for testcase 2 (2 simultaneous small damages) or vice versa, it is not necessary to use a mixed damage case list. The Pareto-optimal solutions for testcase 1 are therefore evaluated for 2 simultaneous small damages, and the solutions for testcase 2 for 1 large damage. The results are shown in Figure 5.6. The plot in Figure 5.6a show that of the 31 (unique) designs which are optimal for 1 large damage, only 2 designs are feasible for 2 small damages. This is caused by the RCM-requirements which are not present in the case of 1 large damage. The two solutions that are feasible, are close to the Pareto-front of testcase 2. A visual inspection reveals that all the infeasible designs have either only one connection to the CIWS, or when the topology has two connections, both connections are routed through the same spaces. Both feasible designs have two connections to the CIWS which' routings are separated in the YZ-plane.

Figure 5.6b show that the optimal designs for testcase 2 are all feasible for 1 large damage, as there is no RCM-requirement in this case. However, it also shows that the found solutions are not optimal for 1 large damage. The reason for this difference is less obvious and can be different for each generated design. Most optimal designs for small damages show similar hubs in the same layer (e.g. LCs) close together. For small damages this is not a problem, as long as they are separated in width by at least one space which is sometimes the case. To deal with a large damage this type of separation is not sufficient, as multiple hubs can be affected with a single hit. In optimal designs for large damages hubs might still be close together but then the effectors are often 'skipping' hubs, i.e. they are not connected to the closest hubs but hubs further apart. Even designs with hubs that are close together, but which can not be affected in a single hit, are likely more vulnerable as path separation is harder to achieve. Instead of the vulnerability point of view, another view is that the designs optimized for 2 small damages are generally more expensive when comparing them to designs optimized for a large damage with the same vulnerability. The designs optimized for 2 small damages often have redundant connections to all hubs and users, as this is effective to decrease vulnerability for multiple hits. To limit the costs the path and source separation is minimized. Again, large damages than nullify the effect of the added redundancy, i.e. there are less expensive designs with the same vulnerability but without the (ineffective) redundant connections.

This shows that when a design needs to be optimized for different damage cases, the optimization should be based on all of these damage cases, especially when the number of different damage extents and hits increases.



(a) Pareto-optimal solutions testcase 1 for 2 small damages



(b) Pareto-optimal solutions testcase 2 for 1 large damage

Figure 5.6: Re-evaluation of Pareto-optimal designs for other damage cases. Dots in red indicate scores of infeasible designs, dots in black indicate scores of feasible designs.

The plot for testcase 3 shows that again the Pareto-front might still be improved as a result of the lower number of generations and population size. The Pareto-front is however fairly continuous, as was the case for testcase 1.

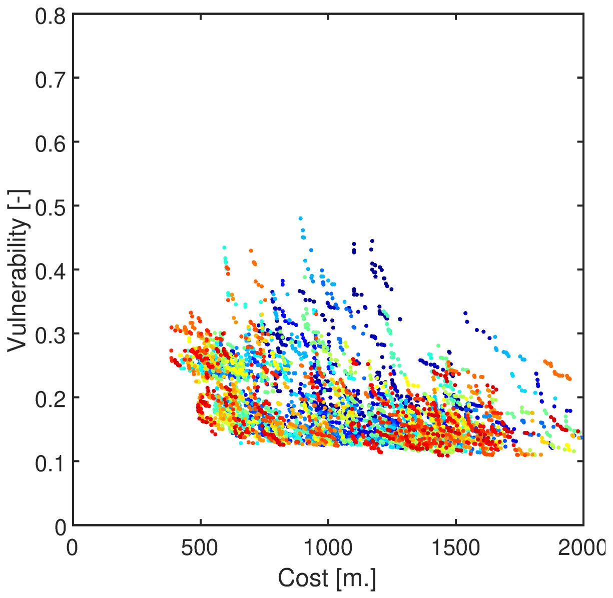


Figure 5.7: Generation plot of outer optimization for testcase 3

The results of testcase 4 (decentralized power network) can be compared to testcase 1, as the effectors and damage case lists are the same for both cases. Again, the Pareto-front does not show significant steps and is well explored due to the relative high number of generations and population size.

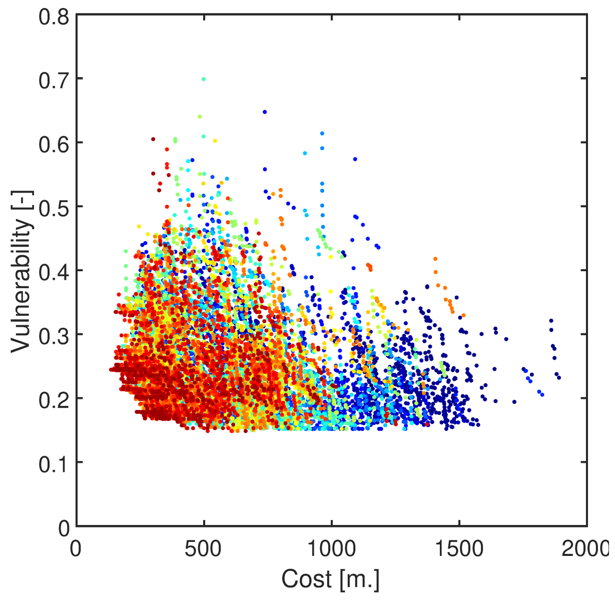


Figure 5.8: Generation plot of outer optimization for testcase 4

The scores for testcase 5 (4 interdependent networks with 1 large damage) is shown in Figure 5.9. The power network and damage case list are the same as for testcase 1. Due to the increased requirements for the effectors to function (dependence on several types of resources) and the dependencies between the networks, the solutions are more vulnerable than for one network. Here the Pareto-front also shows a step, around a path length of 1700 m. The reason for this step is not clearly identified as was the case for testcase 2. Some designs show apparent weaknesses in the power network, whereas other designs seem more vulnerable for malfunctions in the CW-network. This also illustrates the problem with cascading failures when distribution networks are interdependent: it is harder to identify where the actual vulnerability is, and shows why the design process of all distribution networks should be integrated.

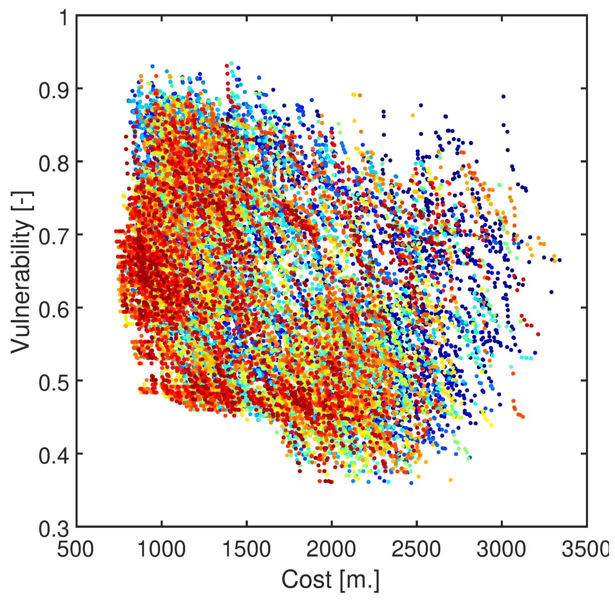


Figure 5.9: Generation plot of outer optimization for testcase 5

5.2.2. Methodology

In the initial testplan metrics are developed per design rule to quantify compliance of a generated solution with a rule. An example of such a metric for rule 2 is the average distance between all source nodes feeding the same node in the layer above it (e.g. all hubs supplying a user, or all suppliers supplying a hub). A larger distance between the source nodes should reflect more source separation and therefore reduce vulnerability. These kind of metrics however proved to be difficult to use for a variety of reasons. This will be illustrated with the example of the metric for rule 2.

First of all it is hardly possible to compare the generated designs. Let's assume a topology in which only one of the five effectors is fed by 2 redundant sources, which are maximally separated. This design would have a large average distance between sources, but has a high vulnerability score (meaning the design is vulnerable) due to the absence of redundant connections for the other effectors. A design where all effectors are fed redundantly with a smaller separation distance would likely have a lower vulnerability. Another example is a design in which a node is fed by 3 redundant connections. Assuming 2 sources are positioned in the same positions as before, and the 3rd source is in between these two sources, the average separation between paths is per definition less then with 2 sources, while the vulnerability is likely decreased in the 3 source design (depending on the damage case list).

This already illustrates the second factor which is complicating these kind of analyses. The vulnerability score is an aggregate of numerous design variables (positions, routing, topologies, etc.), and isolation of the contribution of a single factor such as separation of redundant sources is complicated. Hubs that are widely separated and feed the same effector, but which are both depending on a single supplier can be more vulnerable than hubs that are closer together, but are both fed by multiple suppliers and cross-connected.

The third complicating factor is the dependency between certain design rules. For example the redundant path separation of rule 4 is related to redundant source separation: small source separation will likely result in small path separation, depending on the calculation method. The question is than whether a vulnerable design is caused by inadequate routing as path separation is small, or by a bad source separation (or maybe both are not optimal, but the real vulnerability is a bad topology).

Finally the metrics are formulated in such a way that it analyses a design based on a certain relationship between nodes. For example the metric for rule 2 requires to identify per node the sources, and analyze these relationships. One should however look at the complete chain between the suppliers and effectors to properly assess vulnerability. Besides that, these analyses provide large datasets which cannot be easily compared. It proved for example very difficult to compare the separation of sources of 22 nodes between 3600 topologies. This and the other 3 factors will become even more complicated when considering a multi-network system configuration (testcase 5).

Based on these experiences the analysis methodology was adapted to follow two tracks. The first track is a quantitative analysis looking at the complete population (either of the outer or inner optimization) using the developed metrics. The complicating factors doesn't mean that the metrics are completely useless. It might still be possible to signal certain trends and lower or upper boundaries. These insights will supplement the second track, being a more empirical analysis. For this analysis the Pareto-front of the generated solutions will be used and designs on the extremes of the front will be compared (Figure 5.10). This means that when source separation is considered, the most vulnerable (also the least expensive) and least vulnerable designs on the Pareto-front of the outer GA (the position optimization) will be considered and compared. When the routing optimization should be analysed, for example for path separation, the least vulnerable topology/position combination will be selected, and the extremes on the Pareto-front of the routing optimization will be analyzed. If the difference in vulnerability is very small, a design will be selected which has similar costs as the least vulnerable design, but is more vulnerable. For each testcase the generated designs (layout and topology) at the extremes of the Pareto-front of the outer optimization loop are enclosed in Appendix D.

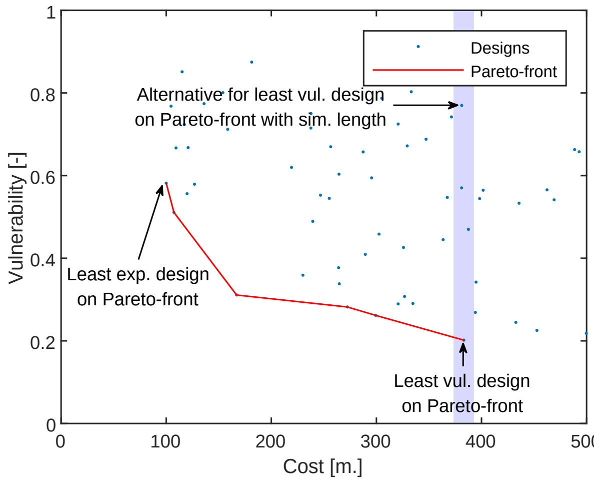


Figure 5.10: Selection of designs for manual evaluation

5.2.3. Rule 1 - Avoid central distribution systems

The first rule to evaluate, states that central distribution systems should be avoided. To evaluate this rule, the system configuration (input) needs to be varied. For this purpose, testcase 1 consists of a traditional power network with fixed positions for the 4 suppliers, and testcase 4 has 6 suppliers with variable positions, allowing for the generation of more decentralized distribution systems. As the same damage case list (for 1 large damage) is used to evaluate the generated designs in both testcases, vulnerability scores can be compared.

A comparison of the population plots in Figure 5.4 and 5.8 shows that for the decentralized distribution system the vulnerability scores and costs are lower than for the traditional configuration. One possible reason for the lower scores is that for the decentralized configuration, the algorithm can position all components in spaces with a relative low hit probability whereas in the traditional configuration the suppliers have a fixed position in high-risk spaces. Also the lower scores are not necessarily due to the fact that the system design is decentralized: maybe the chosen paths are shorter, or the number of connections in the topologies are less. Only if this is both the case, one can truly consider the resulting configuration a decentralized distribution system. For this the least vulnerable designs on the Pareto-fronts are compared. The results are shown in Table 5.5. This table shows that both the number of connections and the costs are much lower for testcase 4. This indicates the lower vulnerability is due to a decentralized configuration where the power is generated locally.

	Testcase 1	*Testcase 4*
Topology ID	1990	2764
Routing ID	916	399
Vulnerability [-]	0.1620	0.1486
Cost [m.]	808.0445	513.8954
Nr. sup-hub connections	4	6
Nr. hub-hub connections (bi)	0	5
Nr. hub-hub connections (uni)	9	0
Nr. hub-effector connections	10	7
Total connections	23	18
Avg. cost/connection [m.]	35.1324	28.5497

Table 5.5: Comparison of least vulnerable designs on the Pareto-front for testcase 1 (centralized distribution) and testcase 2 (decentralized distribution) for 1 large damage.

To check whether this is also the case for multiple small damages, an additional testcase was conducted (testcase 4B) where designs were generated using the decentralized distribution configuration, but a damage case list consisting of 2 small damages. Now a similar comparison can be made between testcase 2 and 4B. The results are shown in Table 5.6. This table shows the decentralized configuration is still performing (slightly) better with regards to the vulnerability, but to achieve this it requires *more* connections and path length than the traditional configuration.

	Testcase 2	*Testcase 4B*
Topology ID	388	432
Routing ID	406	397
Vulnerability [-]	0.0541	0.0509
Cost [m.]	1178.3904	1380.2271
Nr. sup-hub connections	4	6
Nr. hub-hub connections (bi)	0	12
Nr. hub-hub connections (uni)	8	0
Nr. hub-effector connections	9	7
Total connections	21	25
Avg. cost/connection [m.]	56.1138	55.2045

Table 5.6: Comparison of least vulnerable designs on the Pareto-front for testcase 2 (centralized distribution) and testcase 4B (decentralized distribution) for 2 simultaneous small damages.

The difference in length and number of connections of testcase 4B with testcase 4 is caused by both the damage size and damage numbers. For 1 large damage, all suppliers and hubs can be positioned close enough to the effectors. That means that when the suppliers and hubs are damaged, so is the effector or all paths to the effector. For a small damage size, this is not the case[23] and therefore redundant paths need to be formed to decrease vulnerability. This requires separation of sources and paths, resulting in a more connected system which is spread out over the ship. This does not fully explain the cost difference between testcase 2 and 4B. The larger costs in testcase 4B are mainly caused by a larger number of connections in the hub-layer[24]. Not all of these connections are necessary, and it can be expected that the algorithm will remove some of these connections in search of a less expensive design when additional generations are performed.

The conclusion for this design rule is that its applicability depends on the threat (damage extent and number of hits). A local, decentralized configuration such as system configuration B is likely to generate the least vulnerable designs, but when this is not possible due to the threat a distributed and separated configuration is viable option.

5.2.4. Rule 2 - Separate Redundant Sources

The second design rule states that redundant sources should be separated well apart, preferably in all directions (fore/aft, port/starboard and high/low). In this context sources are considered all nodes able to supply nodes in a higher layer (so for configuration A the suppliers are sources for the MSWBs, the MSWBs are sources for the LCs, etc.). To examine this rule the population of the outer optimization loop has been examined. One indicator for separation is the standard deviation of the positions for all sources with variable positions, which is determined for each testcase in all directions. A result of such plot is shown in Figure 5.11.

[23]This is true for the testcases, where input defined that the suppliers could not be positioned in the same space as the effector due to limited volume available. If this was possible, it would likely result in a configuration where the suppliers and hubs were all co-located with the effector.

[24]In system configuration A (testcase 1) the maximum number of hub-hub connections is 14, while the maximum number of connections for configuration B (testcase 4b) is 30.

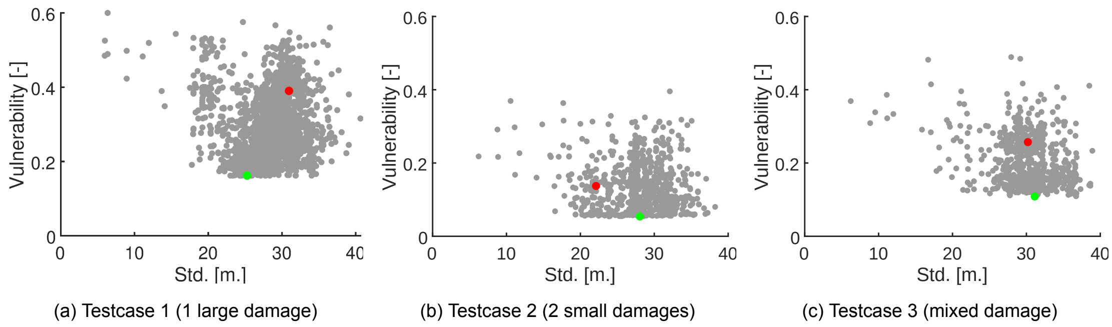


Figure 5.11: Standard deviation in longitudinal direction for hub layer 2 (LCs). The green dot indicates the least vulnerable design on the Pareto-front of the outer optimization loop, while the red dot indicates the most vulnerable design on the Pareto front.

The plots show that an increase in source separation not necessarily means that the vulnerability decreases. This is due to the other factors that influence the vulnerability, like the topology. The graphs show the standard deviation of all hubs in that layer in longitudinal direction, but does not take into account which hubs are actually connected to the effectors, or how they are separated in other directions. What the graphs does show, is that more separation does not necessarily decrease vulnerability, but some degree of separation is required to be able to obtain a low vulnerability score. For example in Figure 5.11a there are no designs with a low vulnerability (below 0.3) when the standard deviation is below 18 meters. The graphs for other directions and layers (not shown) indicate in general similar results, albeit less pronounced due to a smaller bandwidth (there are less possibilities for separation in transverse and vertical direction), with one additional observation. There is no boundary for testcase 1 in transverse direction, while testcase 2 and 3 does show such a boundary. This makes sense as for large damages all spaces over the total width are damaged and separation in this direction does not decrease vulnerability. It does show that separation is depending on the damage extent. This relation cannot be quantified, i.e. the damage extent cannot be directly translated to a prescribed separation distance as this depends on the direction, and the damage volumes considered are discrete.

The separation of the power network in testcase 1 and 5 can also be compared, as the system configuration of these networks are the same and so are the considered damages. The results of the comparison are shown in Figure 5.12. This figure suggests that the required separation for multi-networks is more narrowly distributed then when considering a single network. This means that not only the damage size, but also the network complexity influences the required degree of source separation. This advocates an integral approach when designing these kind of complex networks where also the interaction between networks is considered. It should be noted that it might still be possible that designs with a lower separation are feasible in testcase 5, but were simply not found.

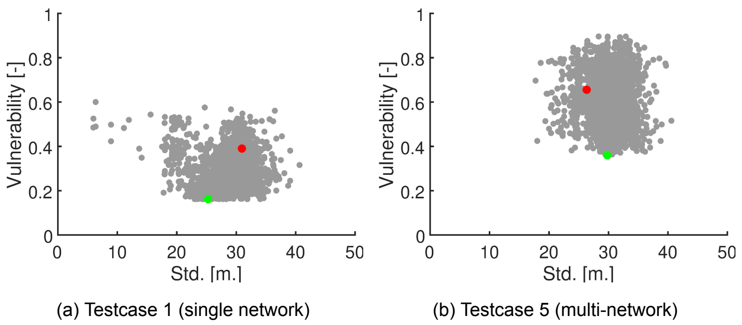
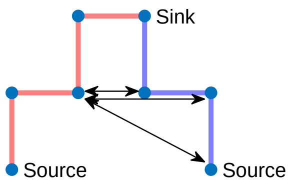


Figure 5.12: Standard deviation in longitudinal direction for hub layer 2 (LC) of the E440V network for 1 large damage.

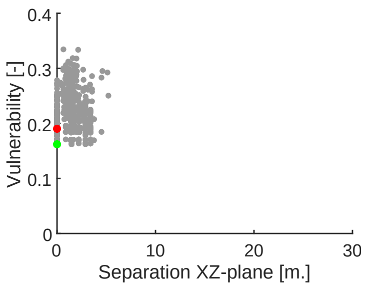
The conclusion for this design rule is that it is a valid rule, but the gains of applying this rule are depending on other features of the design such as topology and successful appliance of other design rules. The degree of implementation (meaning a definition of what ‘well apart’ means) is depending on the damage considered and the complexity of the network.

5.2.5. Rule 4 - Separate Redundant Paths

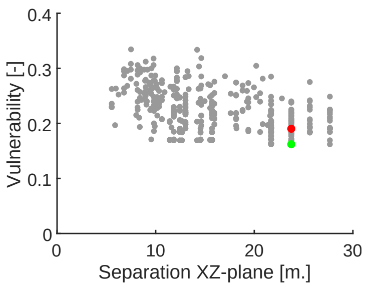
Rule 4 states that redundant paths should be separated well apart, again preferably in all directions. Quantifying the separation of paths is harder than quantifying separation of components, as each path consists of multiple positions and redundant paths may consist of a different number of nodes. There are several techniques available to quantify path separation, for example Path Similarity Analysis (PSA) which is a complex technique used in biology [Seyler et al., 2015], or the Fréchet- or Hausdorff-distance [Singhal, 2004]. These methods are either not readily available and too complex to implement, or unable to quantify path separation without taking other factors such as source separation into account. Instead another method was used. All routings of the least vulnerable topology/position of a testcase were evaluated and redundant paths for each node were identified. For all nodes of both paths the shortest distance is determined to a node of the other path as shown in Figure 5.13a for the blue path. All these distances are summed, and divided by the total number of nodes in the paths. This is done for each pair of redundant paths in a design.



(a) Calculation method



(b) Example plot testcase 1, top. 1990, node MSWB1



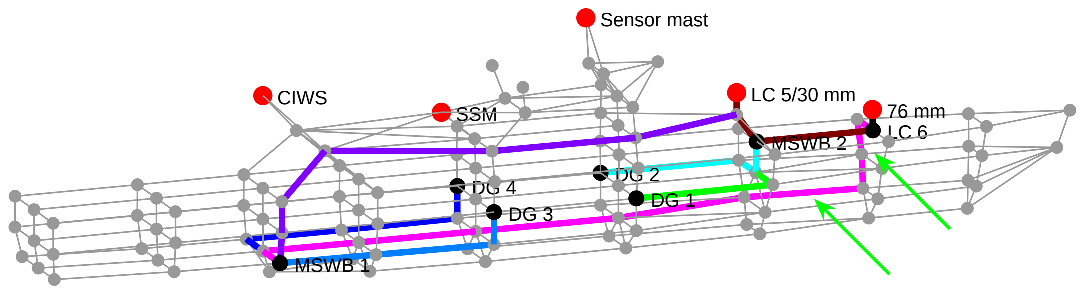
(c) Example plot testcase 1, top. 1990, node LC5

Figure 5.13: Calculation method and example path separation plots, showing path separation for all generated routings

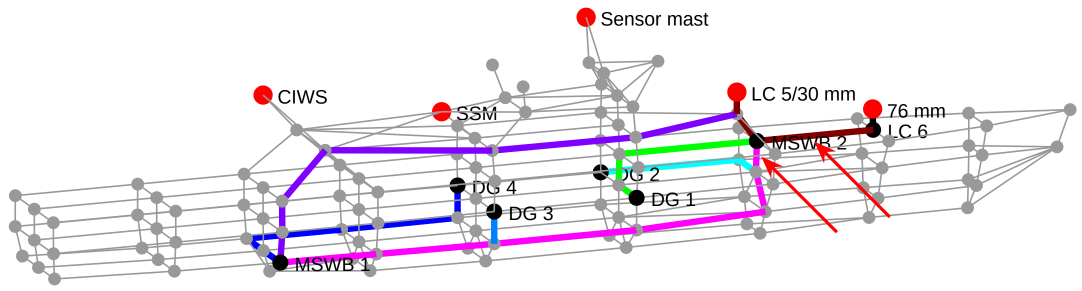
This provides for each topology/position combination a plot with the path separation for each node with redundant paths. This means that the insights are only valid for the topology considered. Figure 5.13c and 5.13c shows examples of such plots for the least vulnerable Pareto solution for testcase 1. The plot shows that it is not possible to find one general correlation for all nodes between path separation and vulnerability. It again depends on other factors as well, such as source separation and topology. Other testcases and topologies showed similar plots. One common feature between these plots is that further away from the effectors (both physically and topologically) the path separation seems to become less critical. This can be explained by the fact that there are more alternative routes available from supplier to effector.

Although hard to quantify, a qualitative analysis of least vulnerable and most vulnerable designs on the routing Pareto-front shows that path separation is indeed used by the algorithm to decrease the vulnerability. In Figure 5.14 the paths relevant for the 76 mm gun are shown. For this testcase with 1 large damage only the separation in XZ-plane is relevant. The main difference in this respect is the routing to LC6. In the most vulnerable variant, the routing from MSWB1 (pink path) and MSWB2 (brown path) is not separated (indicated by the red arrows in Figure 5.14b); in the least vulnerable variant these paths are separated (indicated by green arrows in Figure 5.14a).

Another observation from the qualitative analysis is that the required degree of path (and source) separation is threat dependent in both size and direction. Figure 5.15 shows the distribution of paths in the YZ-plane for the least vulnerable designs of testcase 1 through 3. The designs for testcase 2 and 3 show more separation in transverse direction than the design for testcase 1, as these cases are optimized taking small damages into account. The system components are also more evenly distributed over port and starboard side and in vertical direction, while the centreplane is avoided as these spaces can be affected by hits from both sides.

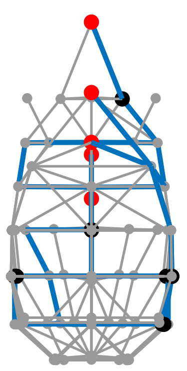


(a) Least vulnerable Pareto-routing

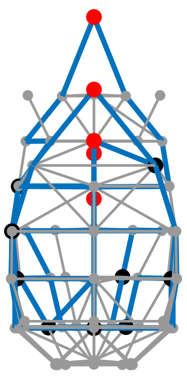


(b) Most vulnerable Pareto-routing

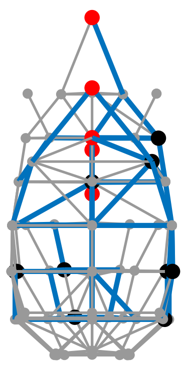
Figure 5.14: Paths relevant for availability 76 mm gun for testcase 1, topology 1990 (least vulnerable design on Pareto-front)



(a) Testcase 1 (1 large damage)



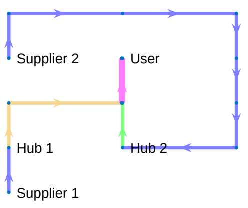
(b) Testcase 2 (2 small damages)



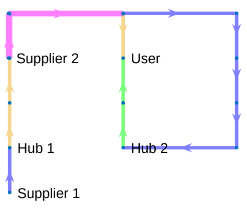
(c) Testcase 3 (mixed damages)

Figure 5.15: Path separation in the YZ-plane, showing least vulnerable designs.

The final observation is that path separation can even be counter-productive for vulnerability reduction when this results in nullifying the effect of other vulnerability reduction measures. This is illustrated with an example in Figure 5.16. In this case the user is supplied by redundant nodes, which are independently supplied by different suppliers. As the paths from the hubs to the user are partly overlapping, these paths are separated. This is actually increasing vulnerability, as the damage envelope is increased and a hit can now disable the path from hub 1 to the user, and from supplier 2 to hub 2 with a single hit. This example seems trivial, but especially when the networks become more complex (e.g. when interdependent networks are introduced) it quickly becomes harder to identify this kind of dependencies (see also Section 5.2.8). It also shows that the complete chain from supplier to user should be taken into account.



(a) Original situation



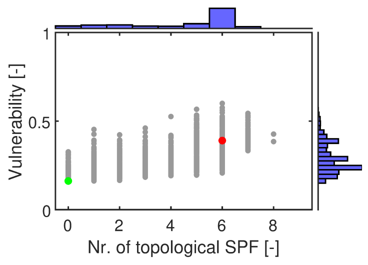
(b) Situation after path separation

Figure 5.16: Counter-productive path separation. Overlapping paths in purple.

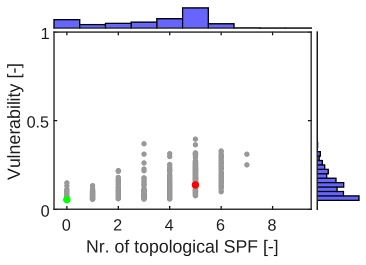
The conclusion for this design rule is that the influence of path separation on vulnerability reduction is bounded by other factors of the design, such as source separation and topology. Path separation could even be counter-productive, as it might generate overlap with other paths introducing other vulnerabilities. That being said, from the qualitative analysis it follows that path separation does contribute to generation of less vulnerable designs. Driving factors in the required degree of path separation is the threat, resulting in a damage size, although this can't be expressed in a single, specific number.

5.2.6. Rule 7 - Avoid Single Point of Failure (SPF)

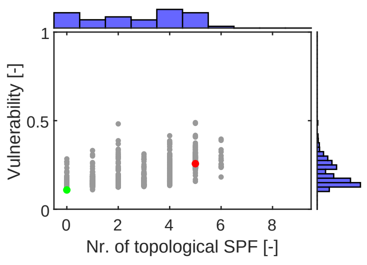
Rule 7 states that SPFs should be avoided so that one hit cannot kill the essential capabilities (in this context effectors). SPFs can be eliminated by adding redundancy and increasing the physical separation. The latter was already discussed, and turns out not to be a guarantee for decreased vulnerability. The idea of eliminating SPF can however be considered for both the topology (logical layer) and the physical implementation (physical layer). For a single network topological SPF can easily be found by removing a node and checking with Dijkstra's algorithm [Dijkstra, 1959] whether all effectors are still connected. This operation was performed for all suppliers and hubs in all topologies. The results of this analysis are shown in Figure 5.17.



(a) Testcase 1 (1 large damage)



(b) Testcase 2 (2 small damages)



(c) Testcase 3 (mixed damages)

Figure 5.17: Nr. of SPFs per topology for testcase 1 through 3. Histograms indicate the distribution of the generated designs.

As can be seen the least vulnerable Pareto-designs have no SPF, while the most vulnerable designs on the Pareto-front have several SPF. From a vulnerability point of view, eliminating SPF is very effective for vulnerability reduction. Contrary to rule 2 and 4, blind application of rule 7 can not have adverse effects on vulnerability, although costs will increase. As shown in Figure 5.17 designs can have topological SPFs and still have a low vulnerability. This is for example the case when a topological SPF is co-located with an effector (e.g. LC 5 in Figure 5.14a). Another example is when a topological SPF overlaps with a physical SPF due to input (e.g. LC6 in Figure 5.14a is a topological SPF, which is also located at the only position having access to the 76 mm gun). A third example of a topological SPF in a survivable design is when the component is positioned in a space that can not be affected. This is for example the case for the decks below the waterline when only small hits are considered. Topological SPF can also be caused by the system input. This is the case for the multi-network configuration in testcase 5. For this testcase, the only supplier in the DataSens network is the sensor mast. The effect of damaging this SPF propagates through the network, as the users of the DataSens network are suppliers in the DataShoot network.

To illustrate the effect a SPF can have, the least vulnerable designs for testcase 1 and 5 are compared.

In testcase 5, the sensor mast is a SPF for the SSM, 30 mm gun and 76 mm gun as these effectors are depending on data which can only be available when the sensor mast is available. In testcase 1, 6 of the 35 damage cases will result in unavailability of the sensor mast. This will cause a total decrease of 0.4% of the MOE, which is 2.2% of the total MOE decrease of this design (16.2%). For testcase 5, 7 out of 35 damage cases will result in unavailability of the sensor mast. This alone will decrease the MOE with 0.4% due to unavailability of the sensor mast as effector. This is comparable to the decrease in testcase 1. However in testcase 5, other effectors are depending on the data of the sensor mast, and unavailability of the sensor mast as only data supplier will result in unavailability of the dependent effectors. This will result in an additional decrease of the MOE of 23.3%. This secondary effect is responsible for 64.7% of the total MOE decrease. This illustrates the significant effect a SPF can have in interdependent networks.

SPFs due to physical implementation (e.g. redundant paths routed through the same spaces) are harder to quantify. A qualitative analysis of the Pareto-front extremes shows that the most vulnerable designs on the routing Pareto-front do have SPFs due to sub-optimal routing. In the least vulnerable designs these SPFs are most of the times resolved. An example is shown in Figure 5.14. The most vulnerable designs show that the redundant paths to LC6 are both routed through the space of MSWB2, as well as the path of LC5 to the effector. If this space is hit, the effector will no longer function as the path of LC5 to the effector is damaged, and also both paths to LC6. In the least vulnerable variant this SPF is eliminated, as the redundant paths to LC6 are separated. When after the optimization SPFs still exist, this is often due to input. Examples of such SPFs are the routing nodes directly below the 30 mm and 76 mm guns and the 2 routing nodes below the sensor mast (as both of these nodes are affected when one node is hit for the considered damage sizes).

The conclusion for this rule is that application of the rule is usually very effective and should always be applied from a vulnerability point of view, as it usually has no adverse affects on the vulnerability. The discussion of the results also showed that, although this cannot always be avoided, the problem definition for the algorithm should avoid SPF as much as possible.

5.2.7. Rule 8 & 9 - Implement cross-over near system's essential users or sources

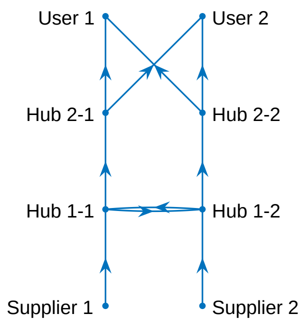
Rule 8 and 9 states cross-overs should be implemented near the system's essential users, or near the systems' sources. Analysis of this rule is complicated due to several factors. First of all, the algorithm has limited possibilities to truly form cross-overs. To form cross-overs in the topological layer, at least 2 hub layers are required which can connect to hubs in the same layer. None of the considered system configurations have these hub layers. System configuration A (traditional power system) is the only configuration with 2 hub layers, but the hubs in one of the layers (the LCs) can only connect to hubs in the other layer (MSWBs) or users and not to each other. Another possibility is to review the physical dimension of the connections. If for example the redundant paths to a node show a lot of the same nodes, this would in fact indicate a cross-over near the sources. There is however no incentive for the algorithm to form these kind of connections as opposed to forming two distinct separated paths, as the costs of both options are the same. A second complicating factor is that cross-overs are hard to identify in the generated designs. Traditional designs often show symmetry (e.g. relative to the centreplane) which makes it easier to identify cross-overs. In the generated designs, components are positioned by the algorithm and it is hard to distinguish a cross-over from a redundant separate path. The third complicating factor is that the rules are only valid when the hit probability is uniform, which was not the case for the testcases.

Due to these constraints no clear analysis of the *physical* location of cross-overs could be performed. An analysis of the *topological* location[25] of cross-overs was performed for the power network[26] of testcases 1 through 3. The problem of the absence of a second inter-connectable hub-layer was circum-

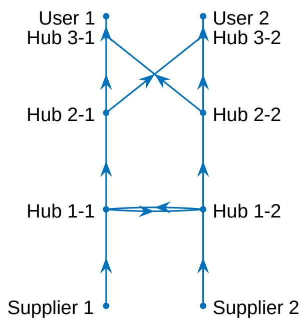
[25] Although in [Spruit et al., 2009] rule 8 and 9 specifically mention 'location' which is illustrated as the physical location, the original research [Streppel, 2004] only analysed the topologies of distributed systems.

[26] The power network in these testcases is the only network with 2 hub layers and therefore able to make a distinction between 'near suppliers' and 'near users'. In reality cross-overs in power networks are less common for practical reasons than for example in CW. However, for the topology analysis this is not a problem although it is accepted that the value of this analysis is largely theoretical in this case.

vented by another interpretation of the generated topologies. Instead of considering the connections from the LCs to the users to be a direct connection, it could also be seen as a connection to a hub co-located with the user, which is directly connected to the user (Figure 5.18). This effectively introduces the possibilities for cross-overs (topologically) near users. The hub layer near the suppliers (MSWBs) can already form cross-connections.



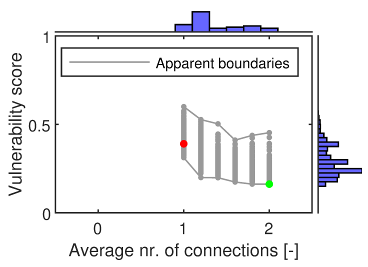
(a) Actual topology



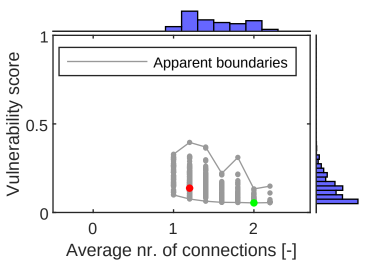
(b) Interpretation

Figure 5.18: Interpretation of topology for evaluation of rule 8 and 9.

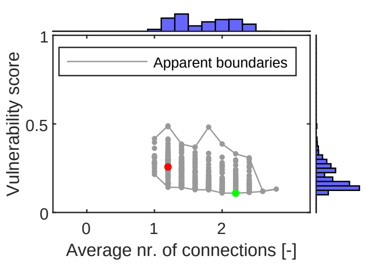
The results for the hub layer near the users are shown in Figure 5.19. The figures show that topologies which has more topological cross-overs have a smaller bandwidth with respect to their vulnerability. Especially the upper boundary is lower for an increasing number of cross-overs. A remark is however that this decrease could also be caused by smaller number of examined designs, and the fact that the upper boundary is not actively explored by the algorithm. The results for the cross-over near the systems' sources is shown in Figure 5.20. This figure shows no clear correlation between vulnerability and number of cross-overs. This is likely due to the small number of hubs in this hub layer and other redundancy options, like redundant connections directly to the LC.



(a) Testcase 1 (1 large damage)

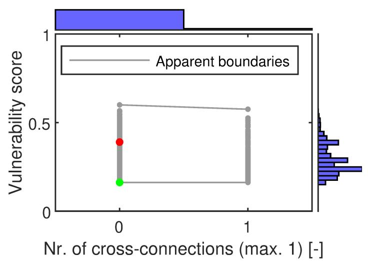


(b) Testcase 2 (2 small damages)

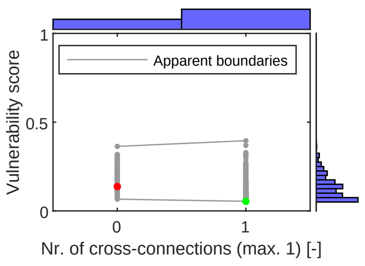


(c) Testcase 3 (mixed damage)

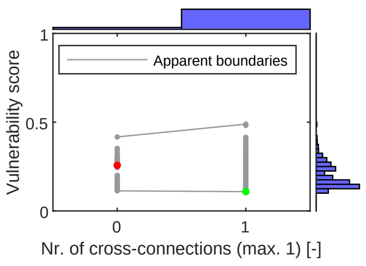
Figure 5.19: Number of connections per topology from LCs to users (near users).



(a) Testcase 1 (1 large damage)



(b) Testcase 2 (2 small damages)



(c) Testcase 3 (mixed damage)

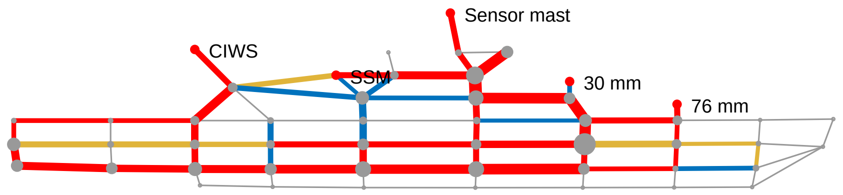
Figure 5.20: Number of connections per topology between MSWBs (near suppliers).

The conclusion for rules 8 and 9 is that, with all the assumptions made for analysis, rule 8 seems topologically valid. For a topological analysis of rule 9 and physical analysis of both rules the model needs further refinement and better suited input. The refinement of the model should create an incentive for the algorithm to create cross-overs rather than separate redundant paths in the cost function. The system input should consist of a network with multiple hub layers which can cross-connect between hubs in the same layer.

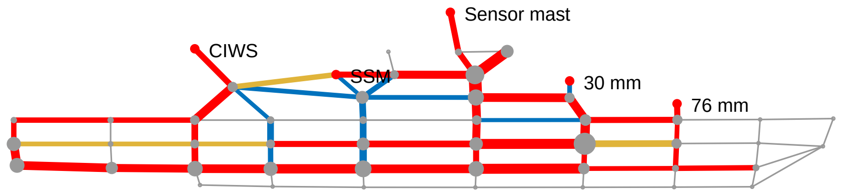
5.2.8. Rule 10 - Combine paths of distribution systems for essential capabilities

Rule 10 states that the damage envelope should be minimized for essential capabilities by combining paths of different required distribution systems. The effect of a combination of paths is again complex to analyse, as not all paths are equally important (e.g. a redundant path is less critical than a non-redundant path) and not all paths are contributing to the same effectors. Testcase 5 is the only testcase where the network consists of multiple distribution networks and therefore the only testcase considered for this rule. This means a path combination should be seen as a path traversing the same zone-deck compartments, as the damage cases consist of 1 large damage.

Comparing the complete network doesn't provide a general insight whether this rule is beneficial. An example of a comparison for the E440V- and CW-network is shown in Figure 5.21. For these 2 networks, the most vulnerable design on the routing Pareto-front has more combined paths in this comparison, especially in the bow section, where less combined routings should be expected based on the design rule. Opposite observations can be made when other networks are combined.



(a) Least vulnerable Pareto-routing

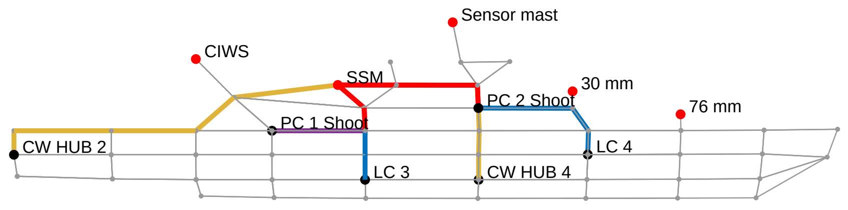


(b) Most vulnerable Pareto-routing

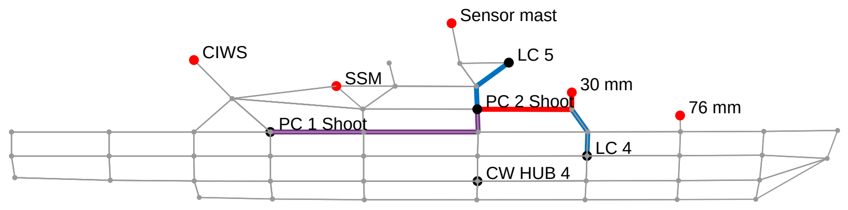
Figure 5.21: The least vulnerable topology/position designs with paths of the E440V- (blue) and CW-network (orange) overlayed on each other. Red edges indicate overlapping paths of these networks. Size of nodes and edges indicate number of paths routed through that node or edge.

To get a good insight whether paths are actually combined resulting in a less vulnerable design, a decomposition is made per node and it's supplying nodes, for all networks. The result of such a comparison is shown in Figure 5.22a. In this case paths of different networks are well separated, and combined while keeping redundancy through separation intact. Figure 5.22b shows what happens when paths are combined while not sufficiently separated. In this case, both (redundant) DataShoot-paths are combined with one E440V-path. This increases the vulnerability, as both DataShoot-paths can be damaged in a single hit.

Application of rule 10 is therefore found valid, only when other rules are also applied. As damage size is not identified as a complicating factor, this conclusion is valid for all damage extents. Combination of paths should not compromise other aspects of vulnerability reduction as this can be counter-productive.



(a) Good example of combination for SSM resulting in smaller damage envelope.



(b) Example for 30 mm gun where combination of DataShoot- with E440V-paths are counterproductive.

Figure 5.22: combination (red) of E440V- (blue), CW- (orange) and DataShoot-network (purple) paths

5.2.9. Rule 10+ - Ring-distribution systems are preferred

Rule 10+ was an addition to the original rules and states that ring configurations are preferred configuration. To test this rule the generated designs for the CW-network of testcase 5 were analysed, as this is the only distribution network in the testcases able to form a ring-distribution. For each design the maximum number of hubs in a ring-configuration were determined. The result is shown in Figure 5.23.

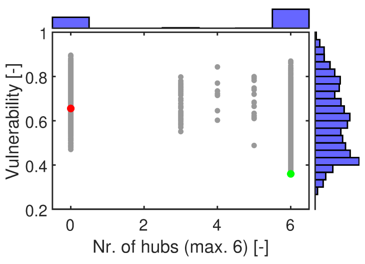


Figure 5.23: Number of hubs in ring configuration

From this figure it can be concluded that ring-distributions are potentially less vulnerable. As shown in the figure, even full ring-configurations can still be vulnerable. This is for example the case when the hubs in the configuration are all in the same space. Therefore rule 10+ is valid, when other rules are also complied with.

5.2.10. Other observations

While evaluating designs for the design rule evaluation, several other observations were made:

- Solutions are heavily threat dependent. It was already discussed in Section 5.2.1 that the best solutions for 1 large damage were not optimal for 2 small damages and vice versa. Another threat-related observation is that paths and components for the small damage cases were preferably positioned in spaces below the waterline, as these spaces are not susceptible to hits.
- Even though the E440V-networks were the same in single- and multi-network configurations, different designs for these networks were generated. In multi-network configuration the components were more separated and evenly distributed, resembling realistic distributed ship designs. This implies that the dependencies between components in different networks limit their positions in less vulnerable designs.
- Non-essential users in system configuration A and B are clustered in the generated designs, as they have no influence on the vulnerability. Positioning all these users in the same space,

close to the suppliers and hubs is the least expensive solution. These users are separated in system configuration C, as their unavailability will influence the vulnerability of the design in this configuration. This shows again the necessity to for an integral design approach of all types of networks.

- Solutions are actually influenced by the defined RCM and capabilities. A good example is the CIWS. For the testcases with 2 small damages (testcases 2 and 3) the availability of the CIWS was optimized e.g. by adding more redundant connections, as a requirement for the CIWS was incorporated in the RCM for this damage type. Another example is the sensor mast. Looking at the designs on the Pareto-front in testcase 1, the first hub-effector connection that was ‘removed’ to reduce costs was the redundant connection to the sensor mast, as the mast contribution to the total MOE only constitutes 1.6%. The last removed connection is the redundant connection to the SSM, which’ availability contributes 37.5% to the MOE. In testcase 5 other redundant hub-effector connections than the ones leading to the sensor mast are removed first, as availability of the sensor mast is a prerequisite for availability of multiple effectors due to the dependency in the data-networks. It shows the necessity to optimize capabilities instead of system availabilities.

5.2.11. Conclusion

Looking at the evaluation of the design rules, it can be concluded that no design rules were assessed as invalid for the testcases considered. There is however a significant grey area to which extent the rules should be applied. It was shown that the unlimited application of certain rules might even be counter-productive. Based on the observations the following is concluded regarding the validity of the evaluated design rules:

- Rule 1 and 7 are valid for all considered cases and can be applied without adverse effects on vulnerability. There is no reason why this would not apply to other operational conditions, such as changing threats and number of hits.
- Rule 2 and 4 are valid, but increasing the separation might not decrease vulnerability and can even be counter-productive. The extent to which separation should be applied is depending on damage size, hit probability distribution, topology and network complexity.
- Rule 8 is topologically valid for all considered cases. The validity of this rule in the physical layer could not be confirmed, as well as the validity of rule 9 due to modelling constraints.
- Rule 10 and rule 10+ are valid for the original assumption when other rules are also sufficiently applied. As there was no relation observed with damage size and number of hits, there is no reason why this rule would not be valid in other operational conditions than the one considered.

5.3. Performance validation

Based on the results of the performed testcases the performance validation can be conducted [Pedersen et al., 2000]. Performance validation consists of 3 aspects, which will be discussed below.

- *Accepting that the found solutions are useful with respect to the initial purpose*. The initial purpose of the method is to find better (less vulnerable) ship designs, by guiding the early stage design effort based on residual operational capabilities for different damage cases. The testcases showed that the method is able to generate a variety of solutions with increasing overall residual capabilities. These solutions are found without strict application of best practices or design rules which are normally used in early stage design. Design rules such as source separation are used to determine an initial starting point for the algorithm, but are not enforced during solution generation. The method is not only guided by the residual operational capabilities after a damage, but also incorporates the set requirements for these capabilities. The time required by the model for generating these solutions shows that the method can be applied in the early stage design process for single hits. For multiple hits or more complicated damage case lists, the usefulness (i.e. the ability to generate solutions given a limited timespan) might decrease. For these cases the usefulness depends on the ability to employ sufficient computing power, simplifying the problem (e.g. further limitation of the damage cases as discussed in Appendix C) or choosing a good starting point for the algorithm.

- *Accepting that the achieved usefulness is linked to applying the method*. For this aspect the generated solutions are normally compared to solutions generated with existing design methods. This step was actually performed during the design rule evaluation. The comparison showed that a blind application of design rules will not necessarily mean that the designs become less vulnerable. This is due to the fact that some rules might be contradictory for complicated networks (e.g. rule 4 and 10), or that some parts of a network are more critical than other parts (e.g. due to number of components or redundant connections). This shows that the ability to generate better designs is actually linked to the method: instead of applying design rules, the method is able to generate optimized designs based on the resulting residual capabilities after a specified damage.
- *Accepting that the usefulness of the method is beyond the case studies*. The testcases show the ability of the model to deal with a number of variations in the input, such as the damage extent, number of hits, system configurations and the complexity of the network. The algorithm is able to generate solutions for each variation, and there is no reason why this wouldn't be the case for a combination of these variations. The time required for the calculations will however increase for more complex combinations and this could limit the applicability of the method for the early stage design process.

Although not strictly fitting to the method of the validation square, the usefulness of the developed method can also be examined by evaluating the compliance of the method with the requirements as formulated in Chapter 3, Section 3.1. Four out of six of the requirements are completely fulfilled; the method is operationally driven, able to deal with requirements for different operational capabilities on different levels, able to vary multiple DVs and evaluate different damage cases. Fulfilment of the two remaining requirements, suitability for the early design stage and easy integration in existing tools, is less obvious. The level of detail the model can handle is in line with the level of detail in the early design stage. The speed of the model can however be a limiting factor. This depends on the number of damage cases the designer wants to evaluate and the computing power at hand. Using the model in practice will provide more insight whether this requirement has been achieved. The same counts for integration with existing design tools. The developed tool for example cannot be incorporated one-on-one in a ship synthesis model, nor does it have the capabilities to replace these models. It is however capable to cooperate with the other tools. This means the output of tools like Packing at DMO can be used to partly define the input of the model (i.e. positions of large components and effectors), where the model can decide on the remaining DV, which' results can then be used by other tools.

6 Conclusions

The problem identified in Chapter 1 consists of 2 parts. First of all designs are largely influenced by best practices and design rules which are used in the early stage design process, while their validity can be questioned. The second part of the problem definition is that new techniques for vulnerability analysis are not yet used to their full potential to drive the design effort. The objective of this research therefore is to integrate recent research on vulnerability assessment of distributed ship systems in the early stage ship design process. This integration should result in vulnerability reduction driven by vulnerability analyses rather than the application of design rules and best practices, as these rules might not always be applicable due to a changing operational environment and novel system configurations. This objective led to the following main research question:

How can vulnerability analyses be used to guide the early stage design effort in order to increase survivability, and how does this influence the contemporary vulnerability reduction design rules?

Four research questions were defined leading to the building blocks of the developed model and the ability to use this model to evaluate the best practices in use.

1. *How are designs created in early stage design and what factors can be used to influence or steer the design effort in this stage?*
   In early stage design two rather distinct processes are taking place at the same time. First of all, in the ship design process the large components (including effectors) and their positions are chosen based on their effectiveness, weight and volume requirements. Secondly, in the distributed system design process the topology and sometimes the smaller components are determined, after which the routing of paths and protection measures are determined in the ship design process. All five Design Variables (DVs) (number and type of components, positions, topology, routing and protection) determining the vulnerability of a system are sequentially determined by hand or using algorithms with (local) optimization techniques, thereby locking large parts of the design space. These optimization techniques often include Multi-Objective Genetic Optimization (MOGO) as used in the discussed ship synthesis models and the Automatic Topology Generation (ATG) tool (Chapter 2).
   The design effort can be guided by choosing the appropriate objective functions for optimization of the DV. As navy ships are build to accomplish a specified mission, the ability of the ship to accomplish this mission (quantified by the Measure of Effectiveness (MOE)) in damaged state should be one of the objective functions. The other objective function selected for the model is a cost function in which the path length is considered, which acts as a counter-weight to the MOE. Based on the findings of this research question, it is concluded that the new method should be able to deal with the vulnerability resulting from all the aspects (DV) of the resulting design.

2. *Which methods exist to assess vulnerability of distributed ship systems and what are their characteristics relevant for integration in the ship design process?*
   From the literature review in Chapter 2 three methods to assess vulnerability in an early stage are

identified. The max-flow-between-hubs method is an a-priori method using metrics associated with vulnerability to assess the vulnerability of a topology. Its inability to incorporate the physical aspects of a design makes it unsuitable for integration. The Markov-based method uses a discrete time Markov chain to determine the probability that systems are available after a number of hits. Computational limitations limit the size of the network which can be considered, although the number of hits is large. The Markov-based method can also provide some guidance on the cause of vulnerability by analysing the eigenvalues and eigenvectors of the transition matrix. Hurt-state percolation determines the availability of systems after a hit by employing a depth-first search of the network. This method is computationally limited in the number of hits it can assess, but the size of the networks it can handle resemble realistic networks. Hurt-state percolation can also identify which component or connection is failing.
Based on these characteristics hurt-state percolation was chosen for implementation, as issues associated with increasing the number of hits for hurt-state percolation were deemed smaller than the issues associated with increasing the complexity of the network for the Markov-based method (Chapter 3).

3. *Which factors of the operational environment are relevant for vulnerability reduction and how are or can they be incorporated to guide the design effort?*
   Three factors of the operational domain are influencing vulnerability reduction (Chapter 2). First of all the required operational output, i.e. the operational capabilities required to satisfy the mission profile. This is incorporated in the developed method by means of an objective function for specified damage cases. These damage cases are the result of the second relevant operational factor, being the threat. The threat the ship faces is integrated in the model by means of differentiated hit probabilities per space depending on threat characteristics (trajectory, fuze setting, weapon domain) and affected surrounding spaces (i.e. the damage extent) given a hit. The third factor, being the employment of tactics and the consequences for vulnerability was not incorporated for the reasons described in Chapter 2, Section 2.5.2.

4. *Which design rules and best practices for vulnerability reduction are used in the design process for distributed systems aboard warships and what are they based on?*
   Numerous design rules and best practices are identified for vulnerability reduction, of which 11 were selected which are commonly used at the Defence Materiel Organisation (DMO). These rules are mostly aimed at the physical implementation of a design through separation and protection, and less focussed on the logical layer of the distributed system. The design rules are all based on the same assumptions. The rules were developed for (1) a single hit with (2) a large damage extent.

The answer of the first 3 research questions lead to an answer of the first part of the main research question. Based on these findings, and confirmed by the testcases as discussed in Chapter 5, it is concluded that to optimize the survivability of a design, the effect of all DVs need to be evaluated and used to optimize the design. This means the DVs are globally optimized. Two methods are considered for this optimization, being a targeted optimization based on the found vulnerabilities of a design during vulnerability analysis, or optimization through systematic variation of all design variables. The targeted optimization option is abandoned due to doubts whether the cause of a vulnerability can be identified with the required level of certainty, especially as this means the second part of the main research question can't be answered. Instead a model is developed which is able to optimize 3 DVs by employing a nested Genetic Algorithm (GA). An objective function quantifying the operational capabilities after sustaining damage ensures that the design is correctly optimized, i.e. optimized to maximize the operational output. The operational environment is further incorporated in the model through variable hit probabilities and damage sizes as it defines the damage cases which are evaluated.

Using the result of the first part of the main research question and the answer to the fourth research question, several testcases were conducted to evaluate the applicability of the design rules for various operational conditions (i.e. damage extents and number of damages) and networks. This leads to the conclusion that some of the design rules (e.g. avoid central distribution systems and Single Point of Failure (SPF)) are also valid when the original assumption (single hit resulting in large damage) is violated. For most rules however such a clear conclusion cannot be drawn. Principles (separation, duplication, etc.) always apply, the extent of implementation however varies and is complicated. Application of the rules *can* result in less vulnerable designs, but will not *guarantee* this and might result in higher costs. On the other hand, not applying a rule might nullify the effect of all the other measures taken to decrease vulnerability. The effect an application of a design rule has, is often related to the application of other rules. As with many complex problems the solution to the problem is not black or white, but a matter of finding the right balance between competing aspects. In this research these aspects are the DVs for which the balance will vary based on the operational situation (damage cases and required operational capabilities), system configuration and operational requirements. The developed method can be used to find this balance, where the current vulnerability reduction rules can help the model to find the optimal designs efficiently by defining suitable starting points.

At the start of this research the expected end state was defined as (1) availability of a method in which distributed system design is more integrated in early stage ship design, (2) where vulnerability assessment techniques are directly and automatically driving improvement of ship designs and (3) validated or refined vulnerability reduction measures. The first two aspects of this expected endstate are fully achieved. The validation or refinement of the design rules is partly achieved. Not all rules could be fully evaluated and no clear defined improvements for the design rules are suggested. The research did however provide insight in how to deal with the design rules, and how the developed method can help with interpretation of the rules.

6.1. Limitations

Before discussing the implications of the research, the limitations of the developed model and the resulting findings are discussed.

6.1.1. Limitations resulting from method

There are still some limitations resulting from the way the model was defined.

1. The fact that a nested optimization with GAs is used means that the number of calculations required to assess the vulnerability will explode with increasing number of hits. There are ways to reduce the number of damage cases as described in Appendix C. This problem might be reduced by employing different optimization techniques (e.g. a separate method requiring less iterations to generate good routing solutions) or choosing better starting points, but the nature of nested optimization will mean the required calculation time remains sensitive to the number of hits.

2. Another limitation is related to the DVs which are not varied, being the system components and the protection. This means that when several system configurations need to be evaluated, again the required time will increase significantly. Besides this, it also means that the generated solutions are bounded by the possibilities of the chosen system configuration. When a system configuration has vulnerabilities such as SPFs, the least vulnerable solutions will be bound by this SPF. Variation of protection requires a modification of the constraining physical architecture, once again increasing the computational effort required. This last effect can however be partly mitigated, as (at least certain types of) protection are applied as a repair measure, which can be applied afterwards. This means only the effect of the applied protection needs to be evaluated.

6.1.2. Limitations resulting from implementation

During implementation several simplifications and assumptions were made, limiting the abilities of the model or the quality of the resulting solutions.

1. One limitation resulting from a simplification was already discovered during the design rule evaluation. This is the cost objective function which creates no incentive to form cross-overs. A more sophisticated cost function taking common paths into account would provide this incentive.

2. Other assumptions result in limitation of the design space. For example the number of paths the routing optimization can choose from is limited by the designer as mentioned in Chapter 4.

Although this was not experienced as a limitation during the testcases, the design space to be explored during routing optimization is undeniably being limited by the limited number of possible paths.

3. The last important limitation is the feasibility of resulting designs. Although some feasibility checks are incorporated (e.g. connectivity of all effectors in an intact network), the generated solutions might still be infeasible as certain ship design aspects are consciously ignored. For example weight and volume restrictions can be manually imposed by the designer through definition of the variable position ranges per component. That begin said, the algorithm can still position several components in the same space exceeding the capacity of the space, as no volume or weight calculations are performed.

6.1.3. Limitations in design analysis

The limitations above are related to the model which generate distributed systems designs. There were also some limitations regarding the analysis of the results for evaluation of the design rules. These limitations are related to the complexity of the design problem in relation to the generated solutions.

1. The complexity of the networks and resulting designs required advanced methods to quantify specific aspects of the design, such as path separation. Even if an aspect could be quantified, the contribution of this specific aspect to the vulnerability was hard to quantify using simple methods, as argued in Chapter 5. It is however doubtful whether application of complex analysis techniques would paint a more unambiguous picture, as the problem by its nature is complex.

2. Another limitation is the applicability of the results of the analysis outside the considered cases. The considered system configurations, damage sizes and number of damages were all varied in the testcases. Constant factors however were the (non-uniform) hit probabilities, variable position input and residual capability requirements. Especially the hit probability distribution has a large influence in the generated designs. Evaluation of the design rules against other threats will therefore likely show different results.

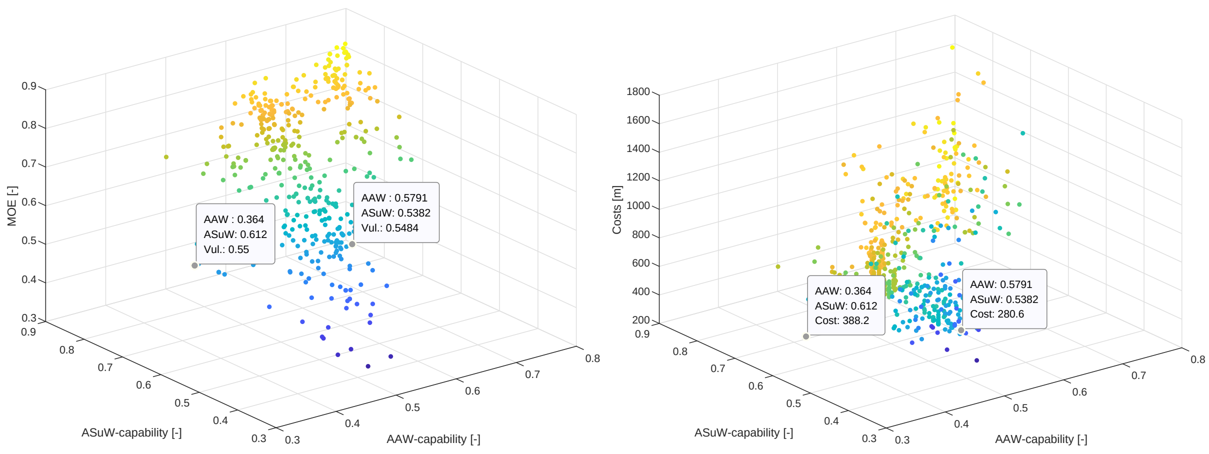
6.1.4. Research limitations

The last limitation is due to the scope set at the beginning of the research. As mentioned in Chapter 1, resource allocation is not part of this thesis and vulnerability is only evaluated based on connectivity of users to suppliers of certain resources. This again could mean that the generated designs are in reality infeasible as the connected sources are unable to deliver their service to all connected users. This limitation was also identified by earlier research ([de Vos, 2018], [Duchateau et al., 2018], [Habben Jansen et al., 2019a]). It should be noted however that of the Pareto-optimal designs, no obvious infeasible designs were observed.

6.2. Implications

Even with the limitations mentioned, the method is capable of globally optimizing multiple design variables for different damages, integrating the distributed ship system design and ship design processes. It can assist the designer in evaluating the effect of a certain system configuration on the vulnerability, and finding the right balance in application of the design rules. In practice two scenarios in which the model can be of value during concept exploration stage are foreseen.

First of all, the method can be used to generate concepts using the experience and knowledge of the designer. The testcases already showed the method can be used in the timeframe and with the level of detail associated with the early stage design for single hit damage cases. For the testcases the input was mostly unrestricted (e.g. a wide variety of positions were available for all components) and the starting point was only roughly guided. In practice the designer should be able to guide the algorithm through both of these factors, making more efficient use of the model. The input can be refined based on other aspects of the design process, e.g the ship synthesis model can be used for obtaining component positions. This implicates all positions of large components are fixed and only small components have variable positions. Based on his experience and design rules the designer can define an initial population with promising designs. The added value of application of the method in this scenario is an increased insight in the potential of concepts with regard to survivability in different operational situations, assisting the designer in finding the right balance in application of the design rules and facilitating the dialogue with the users setting the requirements. The dialogue could be further facilitated by dividing the single MOE-objective in separate objective functions quantifying the separate capabilities (e.g. a subdivision in warfare capabilities as shown in Figure 6.1). As these are already calculated, a simple change in the output of the vulnerability assessment would provide this additional insight and improve the ability to conduct trade-offs between the capabilities.



(a) Capability vs. MOE (b) Capability vs. cost

Figure 6.1: Example of two generated designs in testcase 1 with a similar MOE, but different mix of residual operational capabilities and costs.

Besides concept exploration from a system design point of view, the model can assist in making decisions which are more related to the ship design process. The constraining physical architecture, system configurations, effector positions and protection features are used as input for the model. Changes in e.g. the overall length of the ship or a system configuration will have a significant impact on the vulnerability of the distributed ship system. Although these changes will seldom be based solely on vulnerability considerations, the model can assist in exploring the possible benefits of such changes on vulnerability. This provides the designer with more information, again facilitating the dialogue and enabling the stakeholders to make better informed decisions. As this requires several runs with the model, the variations in the input should however be significant without compromising the comparability of the cases (e.g. the defined spaces should be of similar size). Later on in the concept exploration state the range of variations will become smaller. In that case modification of the earlier found optimal solutions and re-evaluating them might suffice instead of generation of a whole new set of solutions.

In the second scenario, the model is used to explore the vulnerability features of new technical concepts. This use case resembles the testcases which were conducted in which time is of lesser importance, and a more thorough evaluation of a technical concept is conducted. The designer can still define the starting point for the algorithm based on the design rules, but should limit the imposed restrictions only to feasibility constraints. The added value of the model in this case is the fact that the designs are generated and improved based on an evaluation of the features of the design with limited assumptions, while nowadays designs are largely based on design rules which might not apply, or apply to a different extent for the new concepts.

These two applications are in line with the research objective, focussing on increasing survivability of navy ships. The developed method can, in fact, also be applied to any distributed system for which a constraining physical architecture is available, especially when the system consists of multiple networks. One example is an application of the model to decrease vulnerability of ships other than naval ships, but now to counter internal threats such as fire. Another possibility is to use the model in a later stage of the (early) design process when a specific part of the ship is optimized, e.g. a zone-deck compartment.

6.3. Recommendations

As discussed the current model has certain limitations. Based on these limitations, the issues encountered during implementation as discussed in Chapter 4, and the insights gained during the testcases several recommendations are made. These recommendations are subdivided in improvement of the current implementation and additional, more fundamental research directions. Recommendations on how to use the model efficiently, i.e. directions for users, are not discussed in this section but are included in Appendix E.

6.3.1. Current implementation

The following aspects of the current implementation should be improved to increase the flexibility and reliability of the model and confidence in the model:

- *Improve determination of k-shortest paths*. Implementation of a k-Shortest Path with Limited Overlap (kSPwLO)-algorithm will make the calculation of the input paths more efficient as argued in Chapter 4, Section 4.3.2.
- *Develop refined cost function*. Refining the cost objective function to include common paths will likely result in designs where the routing is more realistic. This is particularly the case for networks where cross-overs are common, such as Chilled Water (CW)-networks. The generation of more realistic routed paths will also help build confidence in the model.
- *Increase quality of the topology feasibility check*. In the current implementation the topology repair function only checks connectivity of all essential users to suppliers providing the required resources. Including a check whether the suppliers are actually able to provide a sufficient *amount* of the required resources in intact condition, i.e. load balancing, would increase the quality of the generated solutions, as designs which are in reality unfeasible are eliminated in an early stage. This would also improve the efficiency of the model, as routings for clearly infeasible topologies will not be determined. This improvement would mainly influence the least expensive designs, as these designs often show zonal topologies with an uneven supplier-user ratio (e.g. in extremis one source supplying all users except one, and all remaining sources supplying just one user).
- *Increase possibilities of component positioning routine*. When the model is being used for exploration of new concepts, the variable component positions should be largely unrestricted. In this case the positioning routine can be improved, by adding the possibility to position large components in multiple spaces. This requires a position repair function taking volume and weight restrictions into account. When the variable positions are only used for positioning small components the need to further develop this part of the model decreases, as it is unlikely from a vulnerability point of view that a large number of small components is positioned in the same space.
- *Develop realistic hit probability functions*. As discussed in Chapter 5, the threat has a significant influence in the way components are positioned and paths are routed. In the testcases a bivariate joint Probability Density Function (PDF) was used to describe the hit probability of a space for 6 threat directions, with 2 discrete damage volumes. Hit probabilities and damage extents reflecting actual threats would increase the trustworthiness of the generated solutions.

6.3.2. Additional research directions

During the research several possible research directions were identified which were not pursued for a variety of reasons as discussed in the respective chapters. Nevertheless these research directions could improve the capabilities of the method, make it more efficient or improve the confidence in the generated solutions.

- *Optimization techniques*. Focus of this research was the integration of new vulnerability assessment techniques in the ship design process. The optimization method was selected based on several grounds of which performance was only one. The performance requirement was that the performance should be ‘good enough’. This means that the used optimization techniques most likely can be improved, enhancing the overall performance of the method. More specifically research on improving the optimization technique for the routing problem can be beneficial in the short term, as this requires limited modification of the outer optimization.

- *Resource allocation*. Resource allocation was declared out-of-scope at the beginning of the research. To improve the reliability and quality of the generated designs, resource allocation in damaged state should be further researched. The capability tree which is used for determination of the vulnerability score could be used as a decision making tool when an insufficient amount of resources is available to accommodate all essential users.

- *Directionality*. The possibility to use additional insights of the vulnerability analysis has been discussed in several places in this thesis. The ability to use these insights can provide a more direct insight in why certain designs are more vulnerable and ways to improve a design. Depending on the accurateness with which the vulnerability can be pinpointed, this can be used to improve the current method or develop the targeted improvement method proposed in Chapter 3.

- *Influence of tactics on vulnerability and vice versa*. As mentioned in Chapter 2, the influence of operational considerations (i.e. tactics) on susceptibility is a constant area of attention. The influence of the same tactics on vulnerability of a ship has been less well researched, or at least documented. A better insight in the consequences of tactics on the vulnerabilities would provide valuable insights, and facilitate a more integral approach in increasing survivability spanning from the early design stage to the operational use of the ship.

- *Further develop application of Residual Capability Matrix (RCM)*. The testcases show the added value of including residual capability requirements in the early design stage. Further research in how to quantify these requirements can improve the mutual understanding between the designer and the operational user setting the requirements. The developed model can facilitate this research.

6.4. Personal reflection

As all research, this project started with the problem formulation. Formulating and understanding the exact problem and developing a good plan of approach proved challenging and at times even frustrating for me. However as the research progressed, I realized that the time invested in this early stage of the project was time well spent, as it provided a solid framework guiding my research effort. The literature review helped me in forming my own opinion and forced me to convey my train of thought in a structured way. This last part was especially challenging in the literature review, but became more natural later on in the research project.

During the implementation of the model again it proved beneficial to look from a distance at a specific issue (e.g. implementation of the topology repair function which required several iterations) instead of trying to solve the issue immediately without overseeing the implications of the decisions made. On the other hand, accepting that some problems simply require several iterations to overcome due to complexities that are not immediately apparent was also a valuable experience. Finding the balance between taking the time for analysis and accepting specific uncertainties while proceeding is not something I master after this research. I did however gain additional skills to recognize these situations, which will be further developed throughout my professional career.

The final part of the research was running the testcases and analysing the results. In this stage of the research I realized it has became a more natural habit to critically face observations and results. The analysis of the testcases helped me in further accepting the complexity of the problem and developing a more nuanced opinion about how vulnerability reductions can be achieved.

Besides the skills and competences I developed during the research I gained a better understanding in the early stage design process and the complexities in this process and distributed ship system design. Conducting the research at DMO was a positive experience for me and I believe the research can contribute in the design of more survivable navy ships.

A

Testcase requirements and testplan

Table A.1 shows the required variations in input in order to be able to test the design rules. Table A.2 show the variations based on these requirements (see Appendix B for a more detailed and visual overview of this input). Table A.3 shows per testcase which variation is used, and what the purpose of the testcase is.

Rule		Input variable	Requirement
1	Avoid central distribution systems	System	Multiple system configurations, including both central distribution systems and zonal
2	Separate redundant sources	System Physical	Redundant sources should be included Constraining physical architecture should facilitate separation
3	Apply protection if sources must be in each other's vicinity	-	Will not be tested separately, as this is one of the excluded design variables for the model (i.e. protection can be input, but the model cannot vary protection).
4	Separate redundant paths	System Physical	Redundant sources should be included (functional and spatial) Constraining physical architecture should facilitate separation
5	Apply protection where redundant paths need to be together	-	Will not be tested separately, as this is one of the excluded design variables for the model (i.e. protection can be input, but the model cannot vary protection).
6	Arrange feed and return lines for closed loop system next to each other	-	Will not be tested, as this requires modelling of one distribution network as two separate networks: a feed- and return network. It is not expected that this rule is influenced by e.g. damage size and number, and the modelling effort required is assessed as disproportional compared to the possible gains. This rule could also be seen as a further specification of rule 10.
7	Avoid single points of failure	System	Allow for sufficient possibilities to avoid SPFs.
8	Implement a cross-over near the system's essential users	System	Redundant connections (i.e. two sources to one user) should be possible.
9	Implement a cross-over near the system's sources	System	Redundant connections (i.e. two sources to one user) should be possible.
10	Combine paths of distribution systems for essential capabilities	System	Effectors should require multiple types of resources
10+	Ring-shaped systems are preferred solutions	System	Number of hubs and connection possibilities between hubs should allow for generation of ring-shaped systems.

Table A.1: Testcase requirements per rule.

Setting	Option	Description	Reason for variation
Physical	OPV	▪ OPV-type of ship, approx. 110 meters long, 15 meters wide	Rule 2
		▪ 3 spaces in y-direction where minimum hull width is 8 meters or more	Rule 4
		▪ Results in a total of 112 nodes	
		▪ Hit adjacency matrix where there is only a barrier between traditional machinery spaces, as these positions will not be varied and are necessarily close together.	
System	Common to all configurations	▪ Redundant sources	Rule 2
		▪ Initial adjacency matrix will only reflect feasibility constraints (no steering by limiting possible connections)	Rule 7
		▪ Users should be able to receive resources from multiple sources (redundant sources)	Rule 8 and 9
		▪ Effectors	
		• 76 mm – AAW and ASuW	
		• 30 mm – ASuW	
		• Sensor mast – AAW and ASuW	
		• CIWS – AAW	
		• SSM – ASuW	
	Conf. 1	440V – Conventional	Rule 1
		▪ 4 suppliers	Rule 2
		▪ 2 MSWBs (Hub layer 1)	Rule 7
		▪ 6 Load Centres (Hub layer 2)	
		▪ 7 other users (converters) as defined in conf. 3	
	Conf. 2	440V – Zonal	Rule 1
		▪ 6 suppliers	Rule 2
		▪ 6 SWBs	
		▪ 7 other users (converters) as defined in conf. 3	
	Conf. 3	'Full' network	Rule 1
		▪ Multiple networks:	Rule 2
		• 440V (similar to configuration 1)	Rule 7
		• CW	Rule 10
		• DataSens	
		• DataShoot	
Damage	Hit list A	Compartment-sized damage (large damage extent)	Test design rule assumptions
	Hit list B	Multiple simultaneous space-sized damages (small damage extent)	
	Hit list C	Mixed damages comprising list A and B	
Positions	Standard	▪ Effector positions are fixed and spread over the top-side	-
		▪ Hub positions are variable in the defined range	
		▪ Supplier positions are	
		• Fixed or variable but very restricted due to size and weight limitations in configuration 1	
		• Variable in configuration 2	
		• Fixed or very restricted for 440V network due to size and weight limitations in configuration 3, more variable for other networks	

Table A.2: Variations of the DV.

ID	Purpose	Settings		Rules to evaluate
		System configuration	Damage	
1	Validation of model, setting up of bench-mark and initial verification of rules	Conf. A (440V)	List A	▪ Rule 1 (central distribution systems). ▪ Rule 2 (source separation). Apply to hubs. ▪ Rule 4 (path separation). Only in 2D (zone-deck separation) ▪ Rule 7 (SPF). ▪ Rule 8 and 9 (cross-overs).
2	Test assumption (single large hit vs. multiple small hits	Conf. A (440V)	List B	▪ Rule 1 (central distribution systems). ▪ Rule 2 (source separation). Apply to hubs. ▪ Rule 4 (path separation). Separation in 3D. ▪ Rule 7 (SPF). ▪ Rule 8 and 9 (cross-overs). *Additional over previous cases (added value of the case):* ▪ Assumption of 1 large hit vs 2 small hits
3	Test assumption (single large hit vs. combined design threat)	Conf. A (440V)	List C	▪ Rule 1 (central distribution systems). ▪ Rule 2 (source separation). Apply to hubs. ▪ Rule 4 (path separation). Separation in 3D. ▪ Rule 7 (SPF). ▪ Rule 8 and 9 (cross-overs). *Additional over previous cases (added value of the case):* ▪ Assumption of 1 large hit vs 1 large <u>or</u> 2 small hits
4	Test alternative (more decentralized) configuration	Conf. B (440V)	List A	▪ Rule 1 (central distribution systems). ▪ Rule 2 (source separation). Apply to suppliers and hubs. ▪ Rule 4 (path separation). Separation in 2D/3D (depending on damage case). ▪ Rule 7 (SPF). ▪ Rule 8 and 9 (cross-overs). *Additional over previous cases (added value of the case):* ▪ Additional insight wrt zonal configuration
5	Check results for multi-network configuration	Conf. C (Full)	List A	▪ Rule 2 (source separation). Apply to hubs. ▪ Rule 4 (path separation). Separation in 2D/3D (depending on damage case). ▪ Rule 7 (SPF). ▪ Rule 8 and 9 (cross-overs). ▪ Rule 10+ (ring distribution). *Additional over previous cases (added value of the case):* ▪ Applicability of the rules in interdependent networks ▪ Rule 10 (combine paths).

Table A.3: Definition of testcases with variations per testcase and purpose.

B Input for testcases

B.1. Physical input

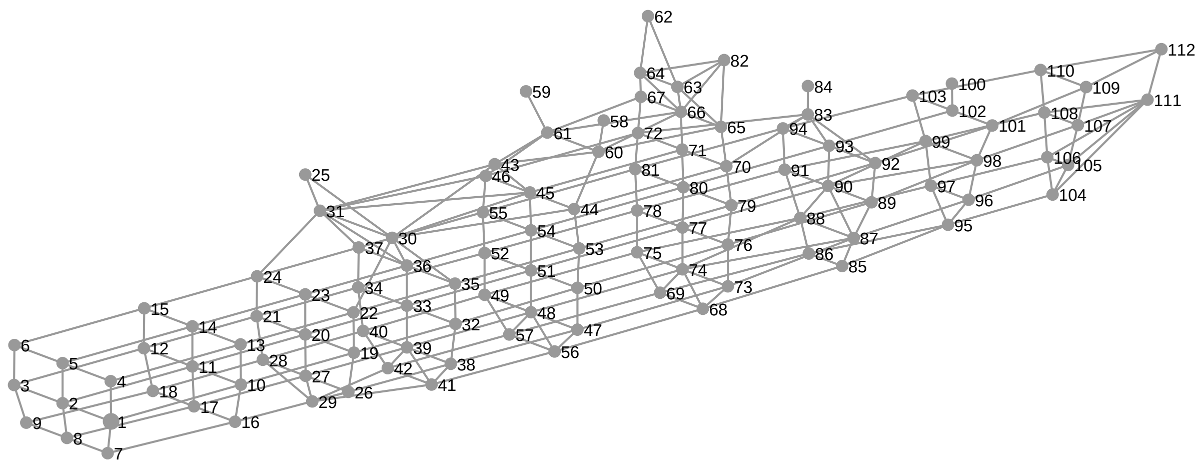


Figure B.1: Subdivision adjacency network

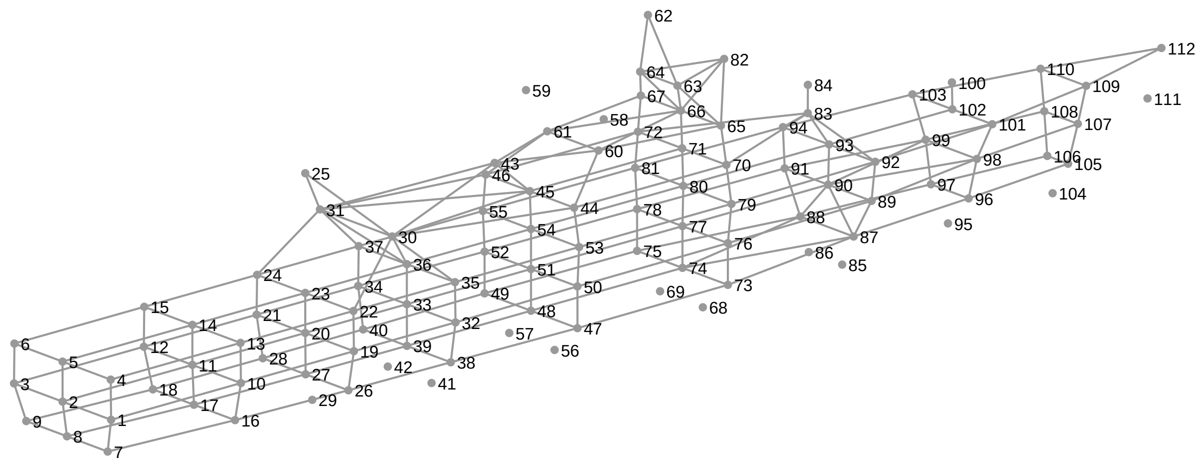


Figure B.2: Routing adjacency network

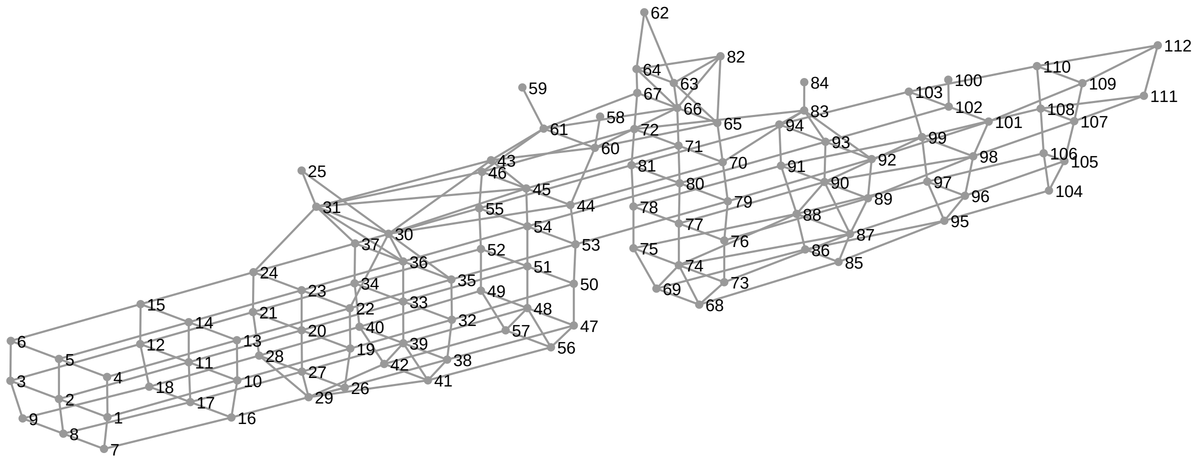


Figure B.3: Damage adjacency network

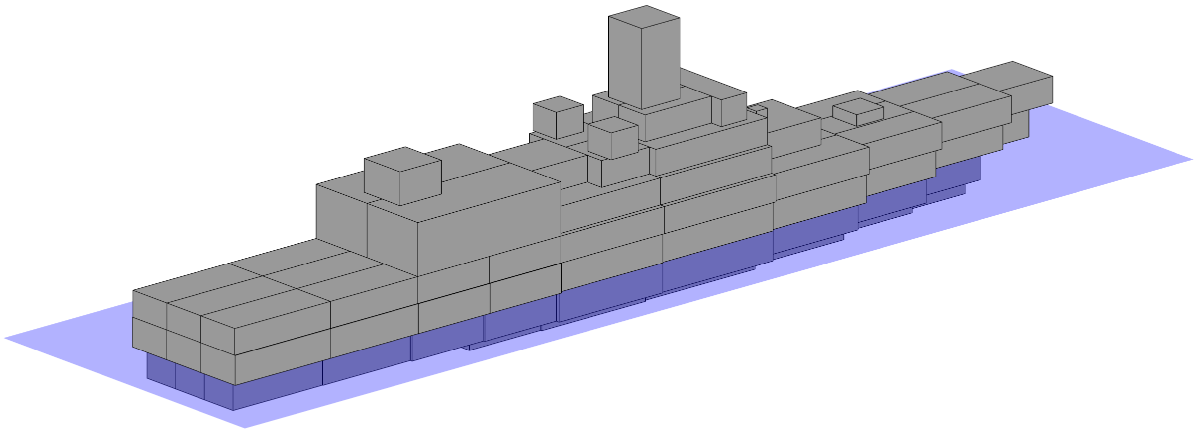


Figure B.4: 3D representation of subdivision network incl. waterline

B.2. System input

This section shows the network diagrams illustrating possible connections between components in different layers and the same layers. Colors of the nodes indicate the function (green is supplier, orange is hub, red is user. The marker for users indicate the type of user (dot is effector, cross is other user).

B.2.1. System configuration A

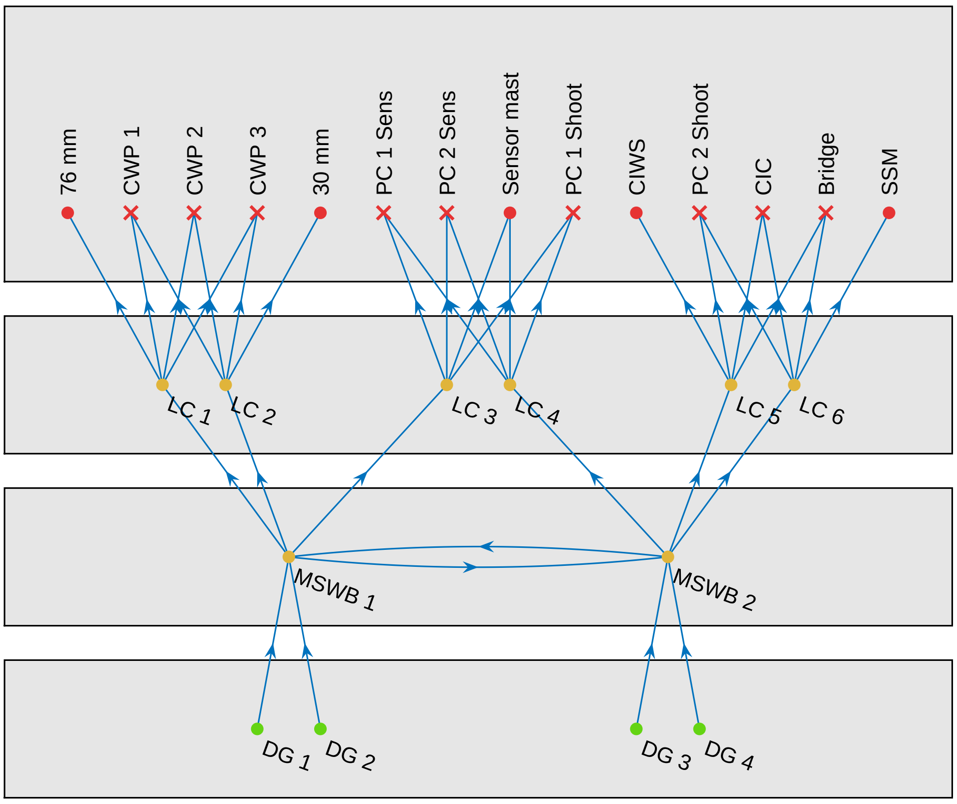


Figure B.5: Network diagram for E440V-network in system configuration A and C. Not all possible connections are shown.

B.2.2. System configuration B

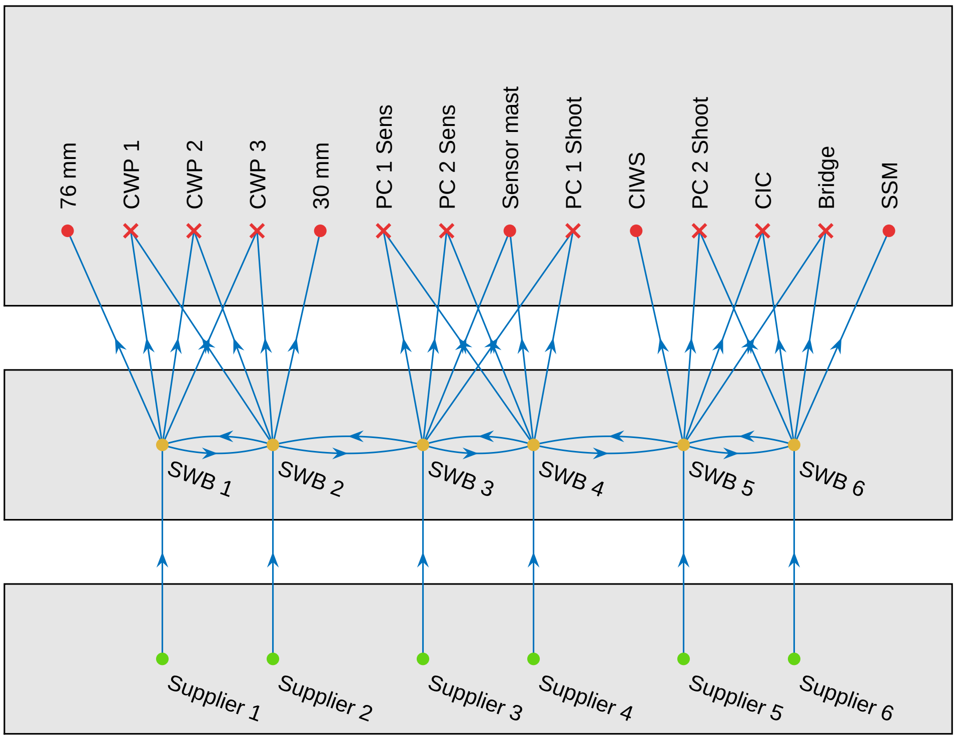


Figure B.6: Network diagram E440V-network in system configuration B. Not all possible connections are shown.

B.2.3. System configuration C

The used E440V network is the same as for system configuration A.

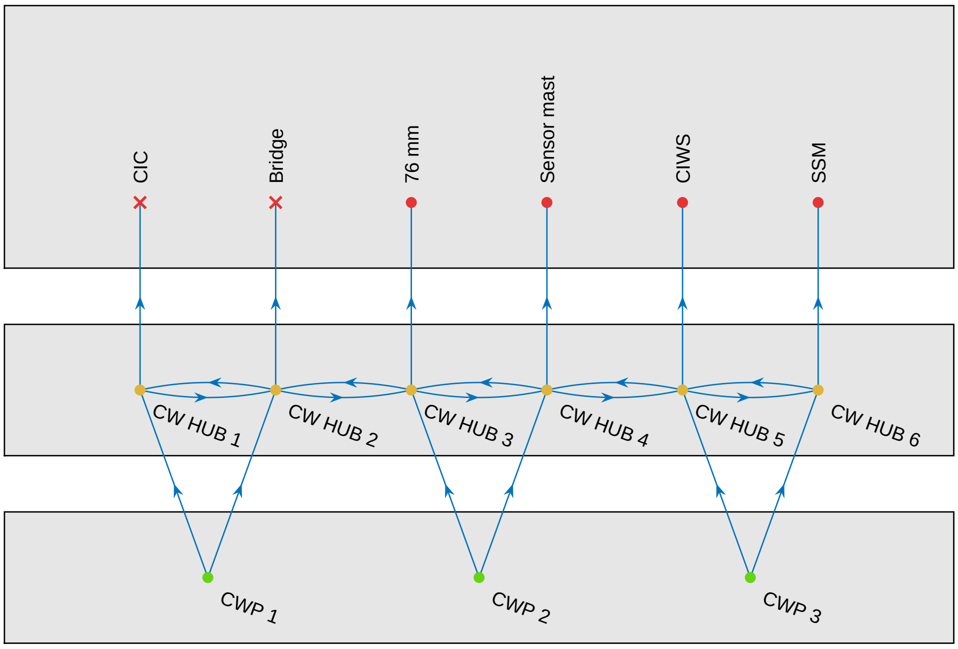


Figure B.7: Network diagram CW-network in system configuration C. Not all possible connections are shown.

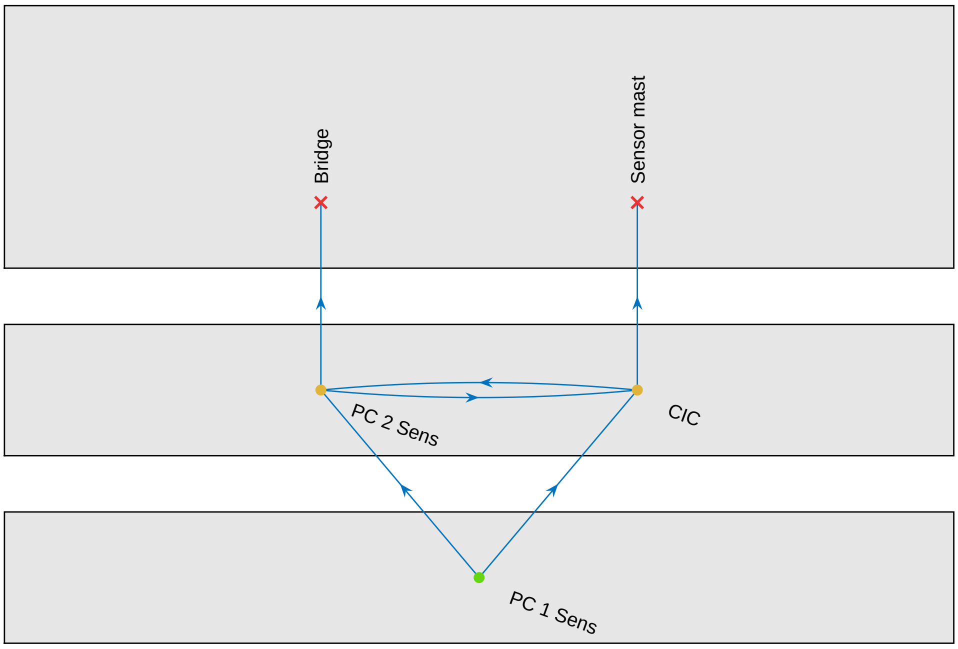


Figure B.8: Network diagram DataSens-network in system configuration C. Not all possible connections are shown.

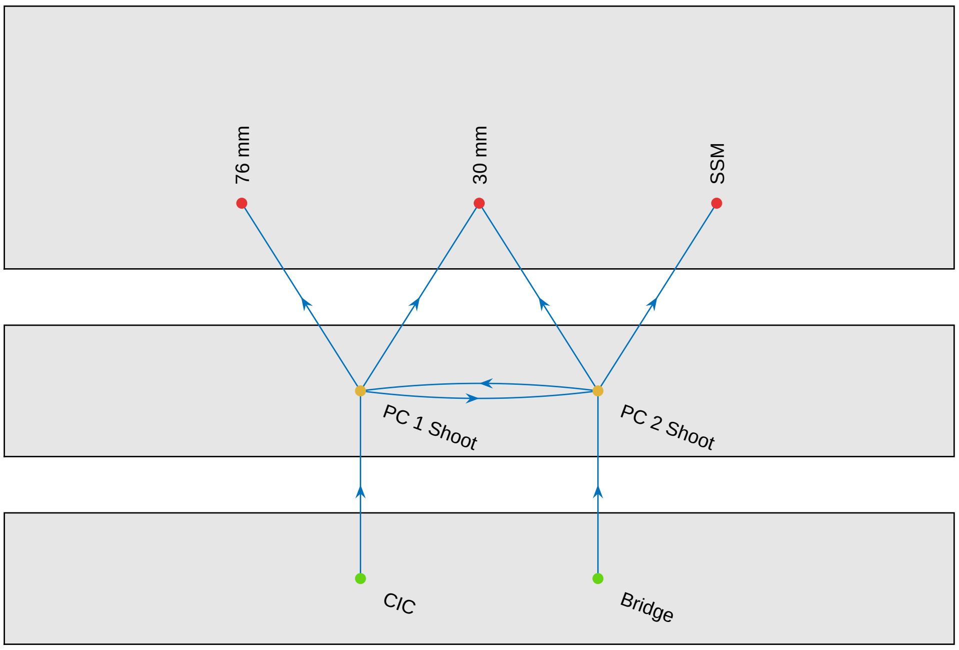
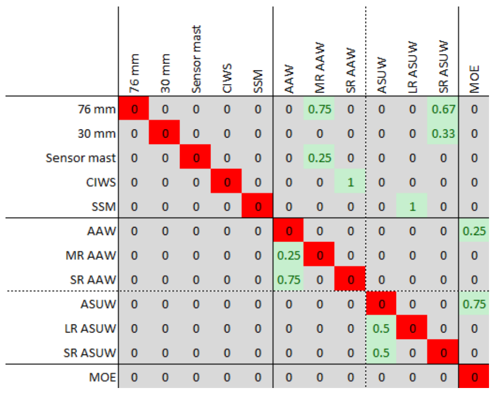


Figure B.9: Network diagram showing possibilities DataShoot-network in system configuration C. Not all possible connections are shown.

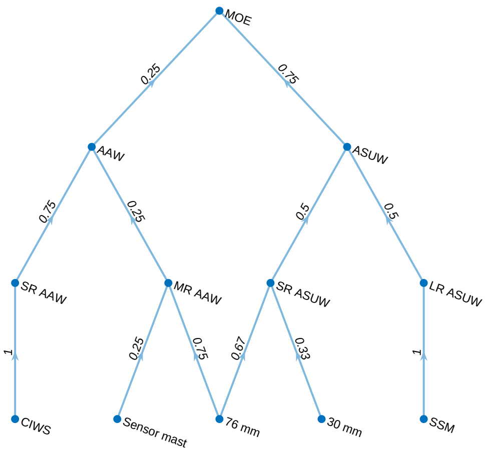
B.3. Capabilities and RCM input

The capability matrix is shown with the resulting capability tree in Figure B.10. As discussed in Chapter 3, Section 3.2.2 the weights of the capabilities are normally determined using expert-opinion and Analytical Hierarchy Process (AHP). For the testcases the capability weights were chosen such that the capability tree resemble a notional Ocean Going Patrol vessel (OPV) with as main task Anti-Surface Warfare (ASuW), and the ability to defend itself against air threats.



	76 mm	30 mm	Sensor mast	CIWS	SSM	AAW	MR AAW	SR AAW	ASUW	LR ASUW	SR ASUW	MOE
76 mm	0	0	0	0	0	0	0.75	0	0	0	0.67	0
30 mm	0	0	0	0	0	0	0	0	0	0	0.33	0
Sensor mast	0	0	0	0	0	0	0.25	0	0	0	0	0
CIWS	0	0	0	0	0	0	0	1	0	0	0	0
SSM	0	0	0	0	0	0	0	0	0	1	0	0
AAW	0	0	0	0	0	0	0	0	0	0	0	0.25
MR AAW	0	0	0	0	0	0.25	0	0	0	0	0	0
SR AAW	0	0	0	0	0	0.75	0	0	0	0	0	0
ASUW	0	0	0	0	0	0	0	0	0	0	0	0.75
LR ASUW	0	0	0	0	0	0	0	0	0.5	0	0	0
SR ASUW	0	0	0	0	0	0	0	0	0.5	0	0	0
MOE	0	0	0	0	0	0	0	0	0	0	0	0

(a) Capability matrix

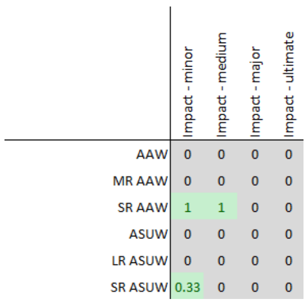


(b) Capability tree

Figure B.10: Input for capability contributions of effectors.

The RCM matrix is shown in Figure B.11. For the testcases where RCM apply is $C_{RCM} = 0.95$. In the testcases the following distinction is made regarding the impact sizes:

- Impact - minor. Damage consisting of 1 hit, small damage extent (not evaluated in testcases).
- Impact - medium. Damage consisting of 2 hits, small damage extent (evaluated in testcase 2 and 3).
- Impact - major. Damage consisting of 1 hit, large damage extent (evaluated in testcase 1, 4 and 5).
- Impact - ultimate. Damage consisting of 2 hits or more, large damage extent (not evaluated in testcases).



	Impact - minor	Impact - medium	Impact - major	Impact - ultimate
AAW	0	0	0	0
MR AAW	0	0	0	0
SR AAW	1	1	0	0
ASUW	0	0	0	0
LR ASUW	0	0	0	0
SR ASUW	0.33	0	0	0

Figure B.11: Input for residual capabilities.

In this case the RCM dictates that for 2 small damages, the Short Range (SR) Anti-Air Warfare (AAW) capability needs to be fully available in 95% of the damage cases ($C_{RCM} = 0.95$).

B.4. Variable position input

The possible positions in this section are indicated by colored nodes. The color of the node is indicating the component type (green for suppliers, yellow for hubs, red for users).

B.4.1. System configuration A

Non-essential users are omitted from this section as they have no effect on the vulnerability in this system configuration. The components and variable position ranges are the same as for system configuration C. In this configuration suppliers have fixed positions and are positioned in spaces 47 through 52 and spaces 73 through 78.

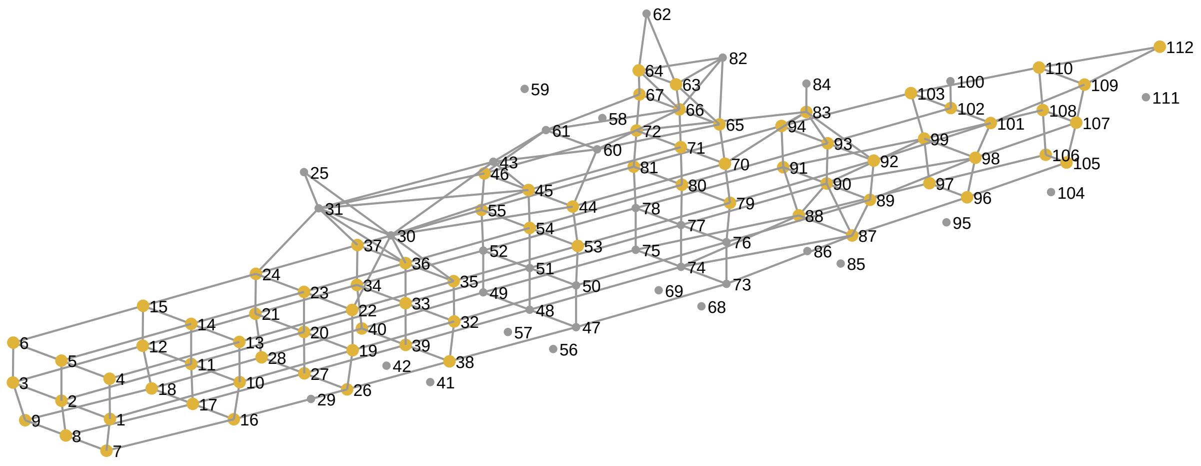


Figure B.12: Possible positions of hubs in hub layer 1 (Main Switch Boards (MSWBs)), E440V-network, system configuration A

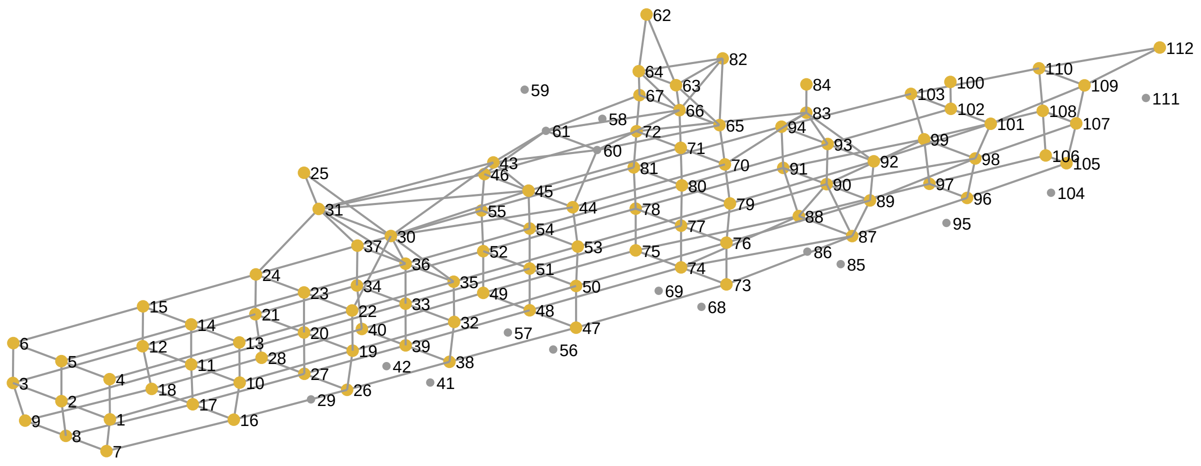


Figure B.13: Possible positions of hubs in hub layer 2 (Load Centres (LCs)), E440V-network, system configuration A

B.4.2. System configuration B

Non-essential users are omitted from this section as they have no effect on the vulnerability in this system configuration. The components and variable position ranges are the same as for system configuration C.

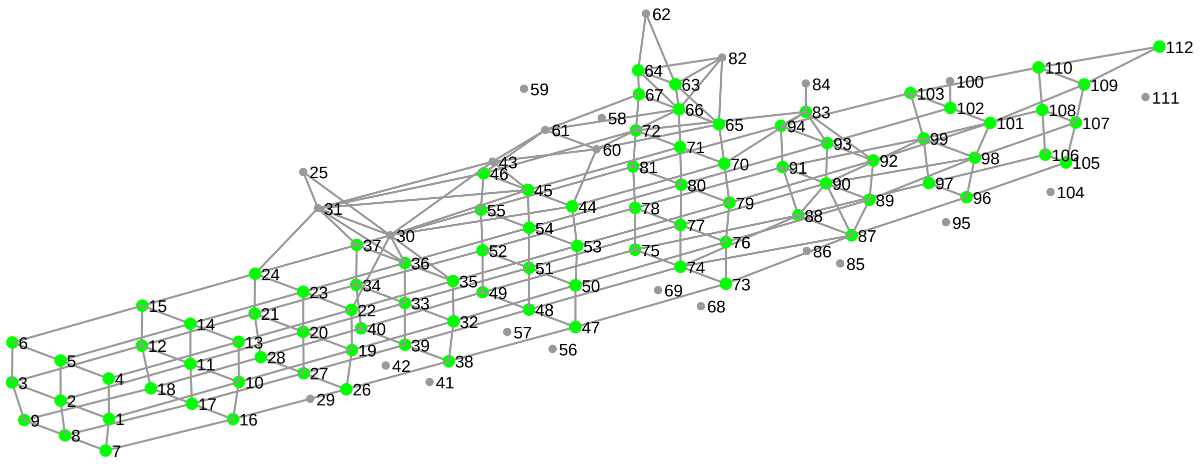


Figure B.14: Possible positions of suppliers, E440V-network, system configuration B

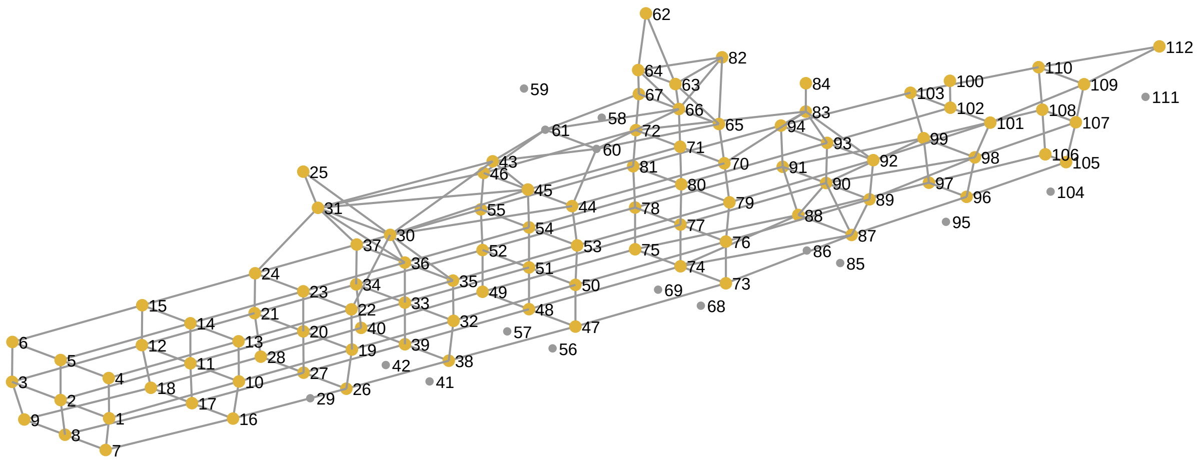


Figure B.15: Possible positions of hubs, E440V-network, system configuration B

B.4.3. System configuration C

All users with variable positions in networks are suppliers in other networks and therefore not explicitly shown. Variable positions of E440V network are the same as for system configuration A and therefore not shown.

CW-network

Users of the CW-network are either effectors or suppliers in the DataShoot-network.

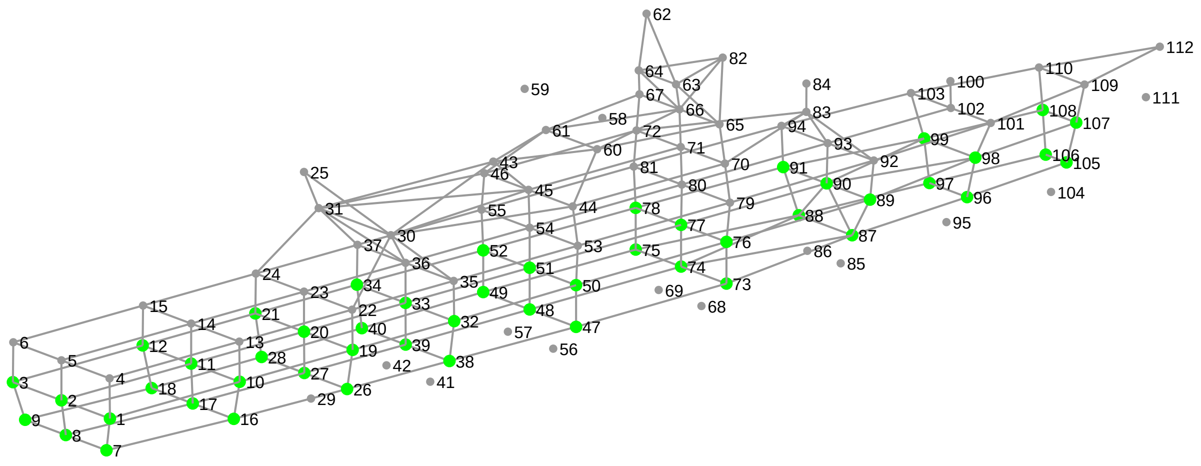


Figure B.16: Possible positions of suppliers, CW-network, system configuration C

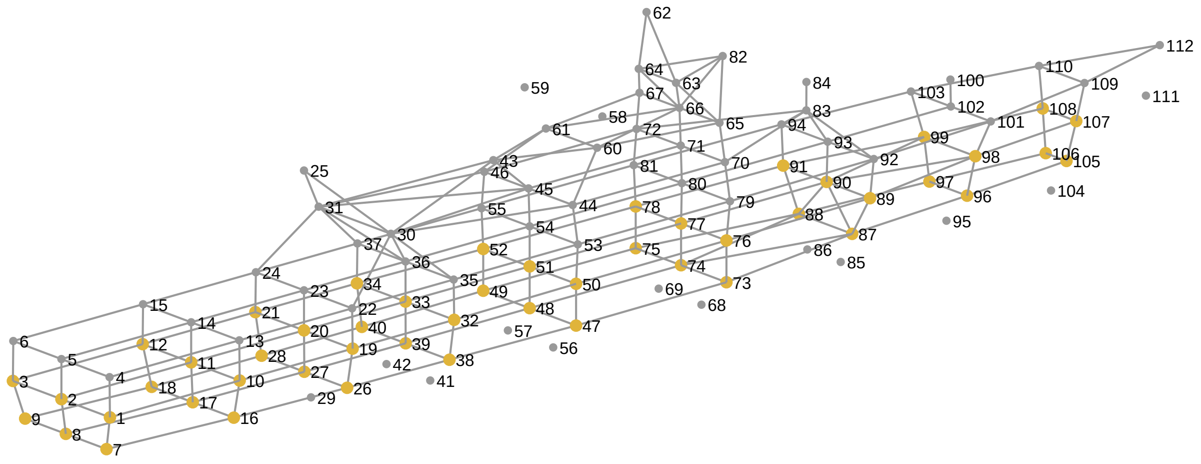


Figure B.17: Possible positions of hubs, CW-network, system configuration C

DataSens-network

The supplier in the DataSens-network is the sensor mast which' position is fixed. The users in the DataSens-network are the suppliers in the DataShoot-network.

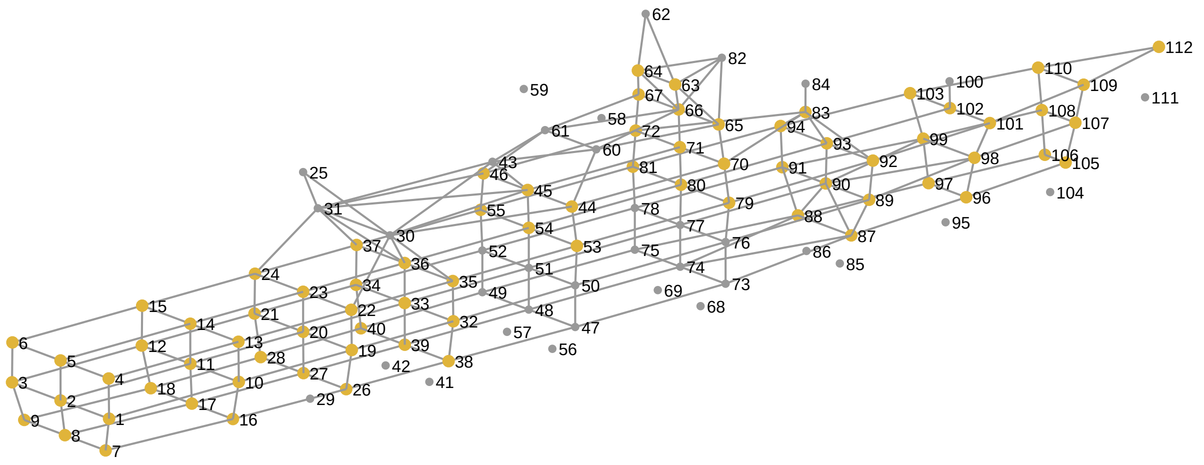


Figure B.18: Possible positions of hubs, DataSens-network, system configuration C

DataShoot-network

One supplier position is fixed (bridge position). The hubs in the DataShoot-network are the same as the hubs in the DataSens-network. All users in the DataShoot-network are effectors with fixed positions.

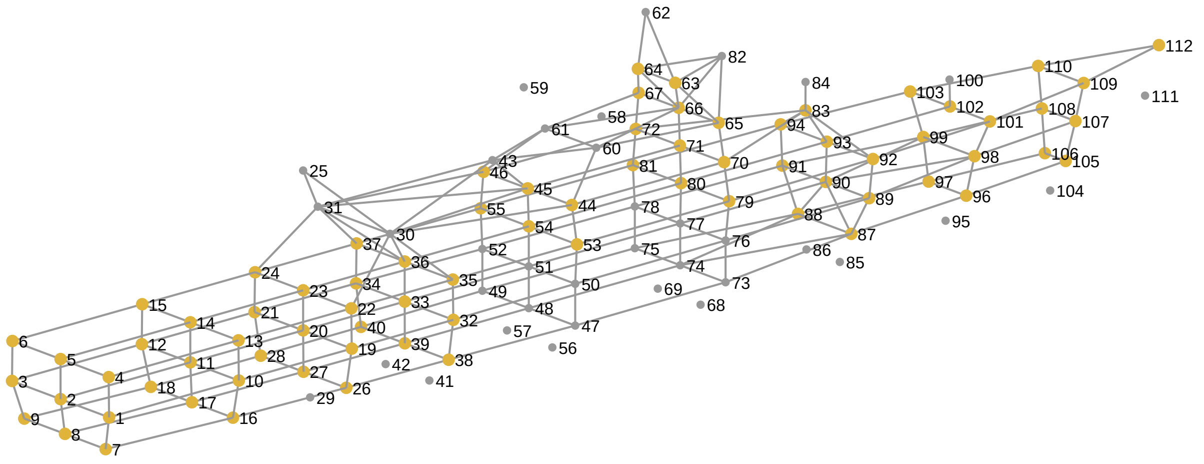


Figure B.19: Possible positions of supplier (Combat Information Centre (CIC)), DataShoot-network, system configuration C

B.5. Threat input

The damage extent for the testcases are divided in small and large damage sizes as discussed in Chapter 5, Section 5.1.1. The hit probabilities are defined using the bivariate normal distribution as discussed in Section 4.3.6, with for each plane with the following parameters (ρ is the correlation coefficient):

$$\begin{aligned}
\rho &= 0 \\
\sigma_x &= 40.00\ [m.] \\
\sigma_y &= 40.00\ [m.] \\
\sigma_z &= 12.00\ [m.] \\
\mu_x &= 48.74\ [m.] \text{ (based on COG in longitudinal direction)} \\
\mu_y &= 0.00\ [m.] \text{ (based on COG in transverse direction)} \\
\mu_z &= 5.50\ [m.] \text{ (based on aim point 1 m. above waterline)}
\end{aligned}$$

The hit probabilities in the XY-plane are not relevant for the testcases, as there is no high-diving threat. Therefore there are no spaces susceptible for hits in the XY-plane.

B.5.1. Hit probabilities in XZ-plane

The hit probability distribution in the XZ plane are shown in Figure B.20

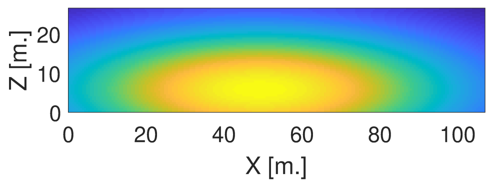


Figure B.20: Hit probability distribution in XZ-plane.

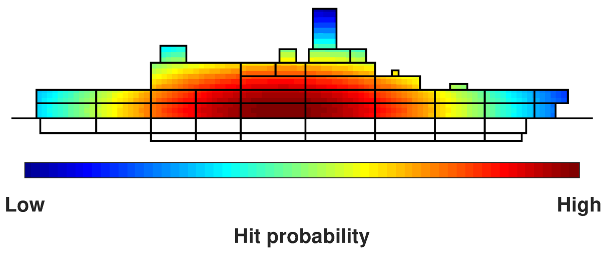


Figure B.21: Hit probability distribution in XZ-plane with ship silhouette overlay.

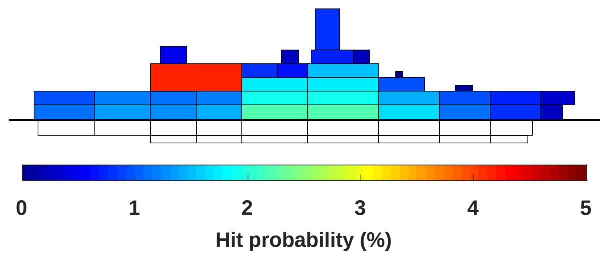


Figure B.22: Hit probability distribution in XZ-plane per space for hit from starboard side.

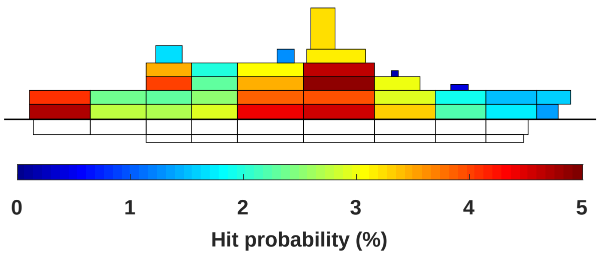


Figure B.23: Hit probability distribution in XZ-plane per zone-deck. The hangar is subdivided in 4 quadrants to differentiate the other affected compartments (hangar will be damaged regardless the exact hit position). The hit probability is the sum of the hit probabilities of a zone-deck from all threat directions.

B.5.2. Hit probabilities in YZ-plane

The zone-deck hit probability in YZ-plane is the same as in the XZ-plane, as zone-decks are hit over the full width.

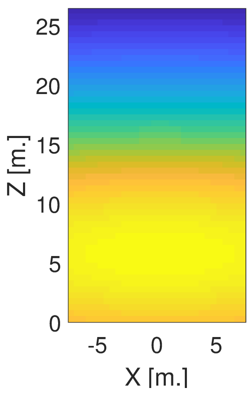


Figure B.24: Hit probability distribution in YZ-plane.

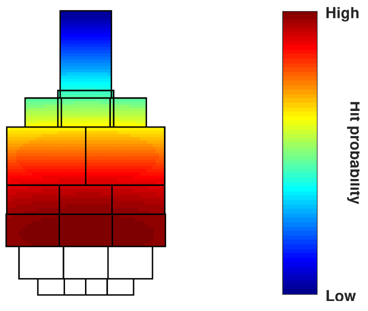


Figure B.25: Hit probability distribution in YZ-plane with ship silhouette overlay.

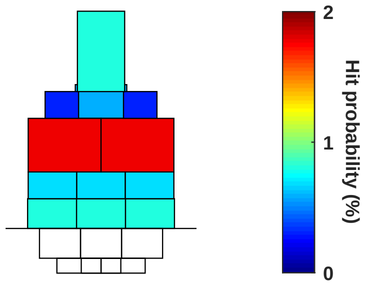


Figure B.26: Hit probability distribution in YZ-plane per space for hit from aft side.

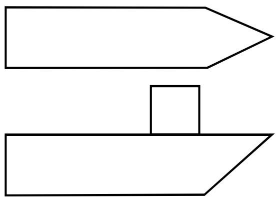
C Reduction

This appendix visualises the steps taken to reduce the number of damage cases to consider for 2 hits resulting in 2 damages. As such the number of possible combinations for n spaces and h simultaneous hits is:

$$C_h = \binom{n}{h}$$

Figure C.1 show the reductions actually used for the testcases. Figure C.2 show possible further reductions, which were not used.

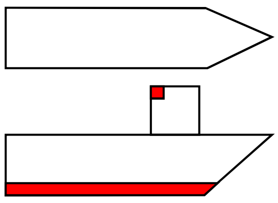
112 spaces



1 damaged space	:	112	cases
2 damaged spaces	:	$\binom{112}{2} = 6216$	cases

Only assess spaces which can hold path or component (tanks, exhausts, etc. are excluded)

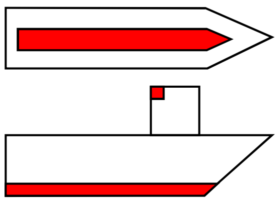
98 spaces



1 damaged space	:	98	cases
2 damaged spaces	:	$\binom{98}{2} = 4753$	cases

Limit spaces that can be hit directly to the outer shell (a space at the centerline can be affected, but this is a result of a space in the outer shell being hit)

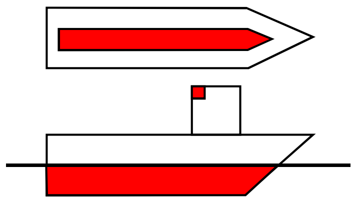
81 spaces



1 damaged space	:	81	cases
2 damaged spaces	:	$\binom{81}{2} = 3240$	cases

Limit spaces that can be hit to spaces above waterline

62 spaces

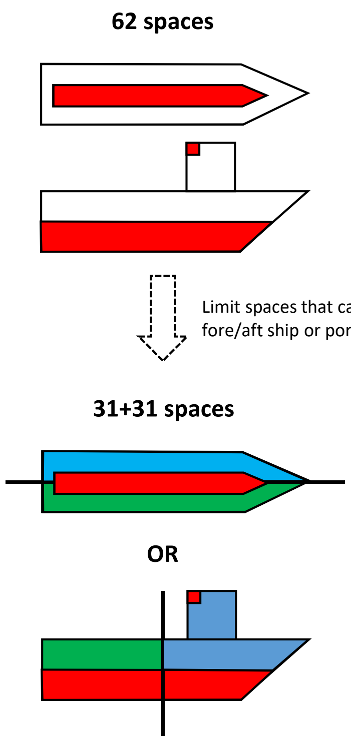


1 damaged space	:	62	cases
2 damaged spaces	:	$\binom{62}{2} = 1891$	cases

Remove similar hitcases

2 damaged spaces 1590 cases

Figure C.1: Reduction of nr. of damage cases to evaluate for 2 damages resulting from 2 hits.


62 spaces
Limit spaces that can be damaged simultaneously to fore/aft ship or port/starboard side
31+31 spaces
OR

1 damaged space	:	62	cases
2 damaged spaces	:	$\binom{62}{2} = 1891$	cases

1 damaged space	:	62	cases
2 damaged spaces (same part/side)	:	$\binom{31}{2} + \binom{31}{2} = 930$	cases
2 damaged spaces (dif. part/side)	:	$31 \times 31 = 961$	cases

Figure C.2: Possible further reductions.

D Generated designs

D.1. Testcase 1

D.1.1. Least vulnerable design

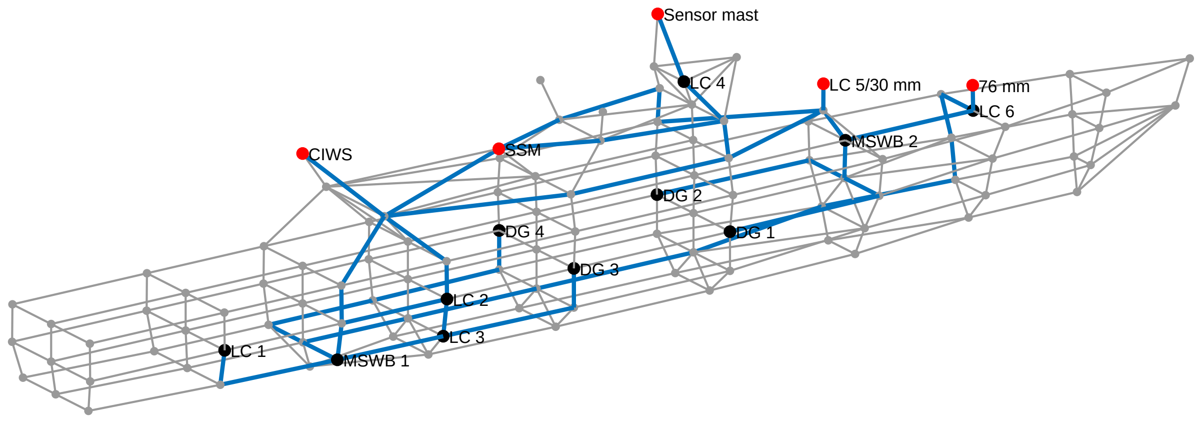


Figure D.1: Physical layout testcase 1, topology ID 1990, routing ID 916 (least vulnerable design on Pareto-front).

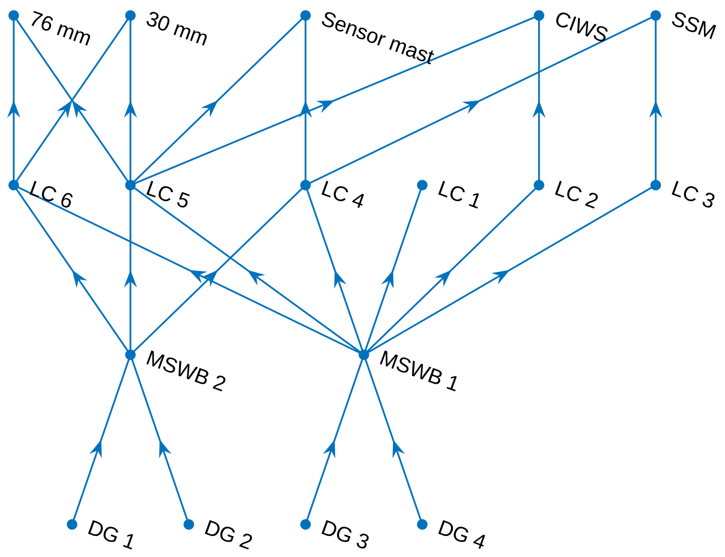


Figure D.2: Topology testcase 1, topology ID 1990, routing ID 916 (least vulnerable design on Pareto-front).

D.1.2. Least expensive design

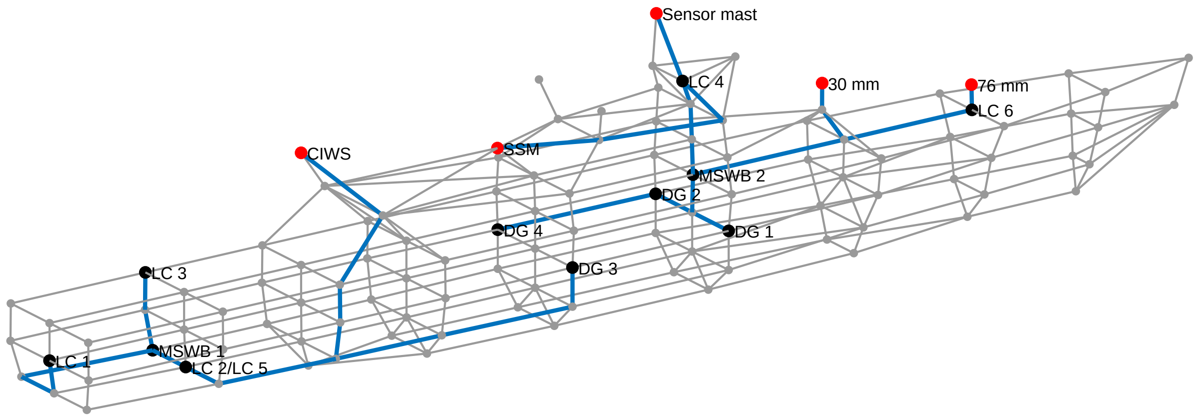


Figure D.3: Physical layout testcase 1, topology ID 2899, routing ID 485 (least expensive design on Pareto-front).

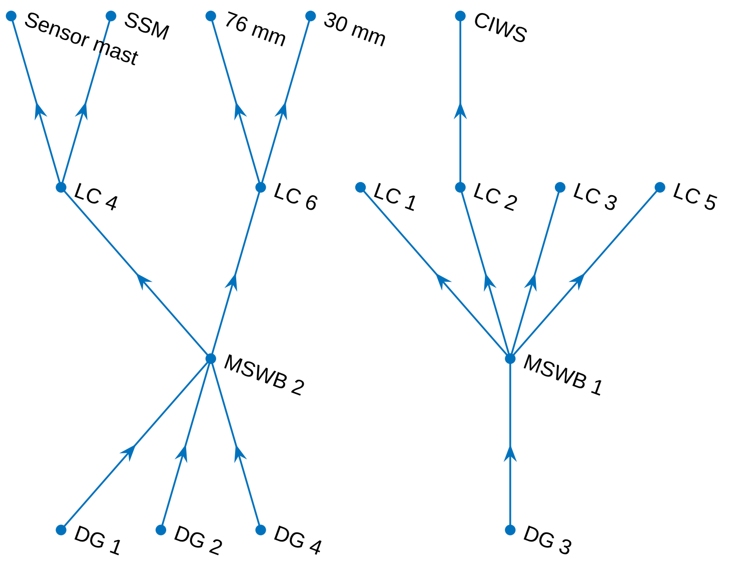


Figure D.4: Topology testcase 1, topology ID 1990, routing ID 916 (least expensive design on Pareto-front).

D.2. Testcase 2

D.2.1. Least vulnerable design

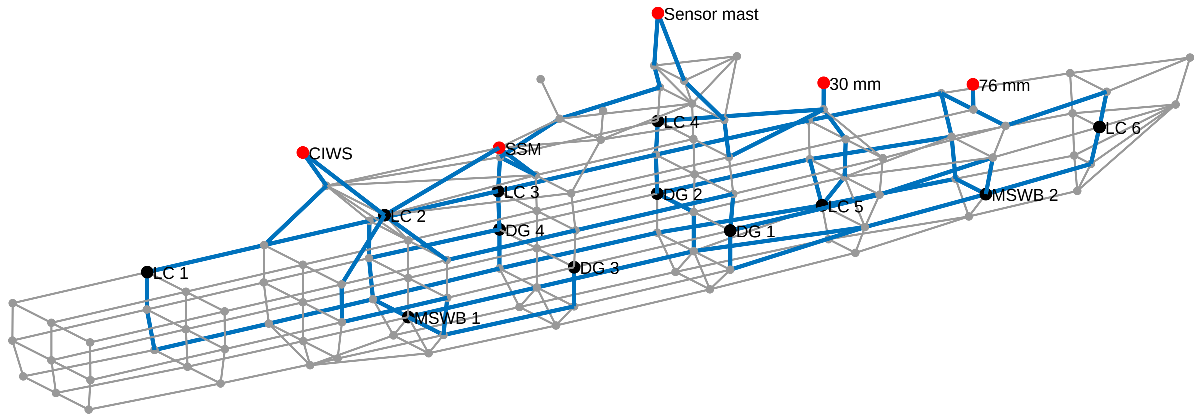


Figure D.5: Physical layout testcase 2, topology ID 388, routing ID 406 (least vulnerable design on Pareto-front).

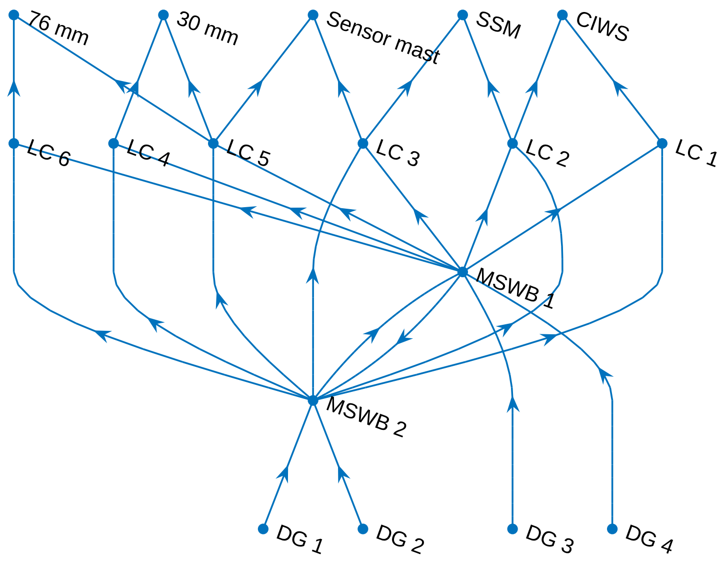


Figure D.6: Topology testcase 2, topology ID 388, routing ID 406 (least vulnerable design on Pareto-front).

D.2.2. Least expensive design

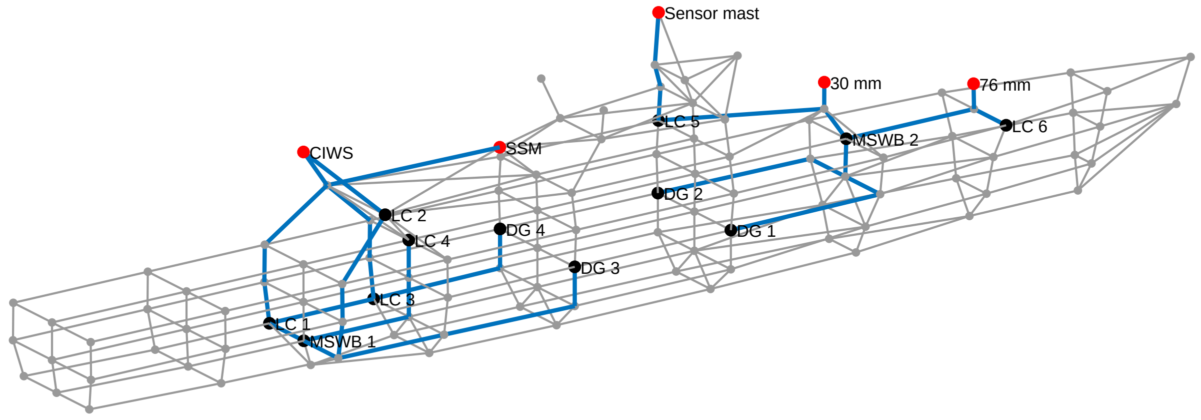


Figure D.7: Physical layout testcase 2, topology ID 613, routing ID 401 (least expensive design on Pareto-front).

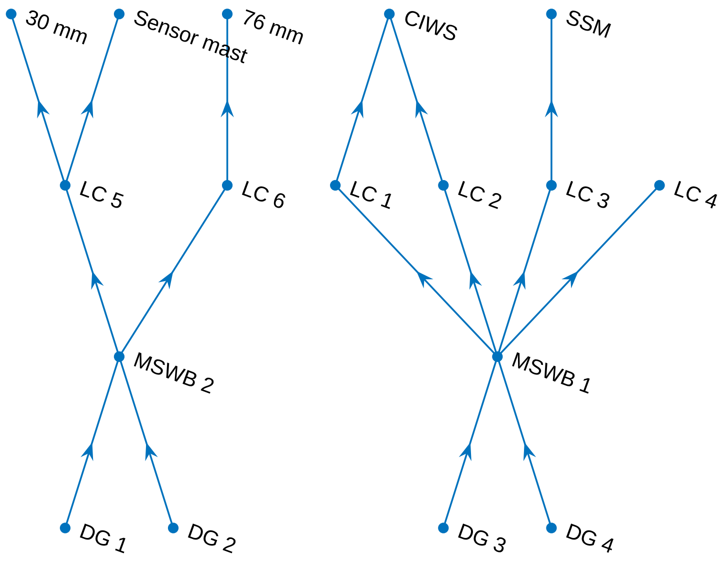


Figure D.8: Topology testcase 2, topology ID 1990, topology ID 613, routing ID 401 (least expensive design on Pareto-front).

D.3. Testcase 3

D.3.1. Least vulnerable design

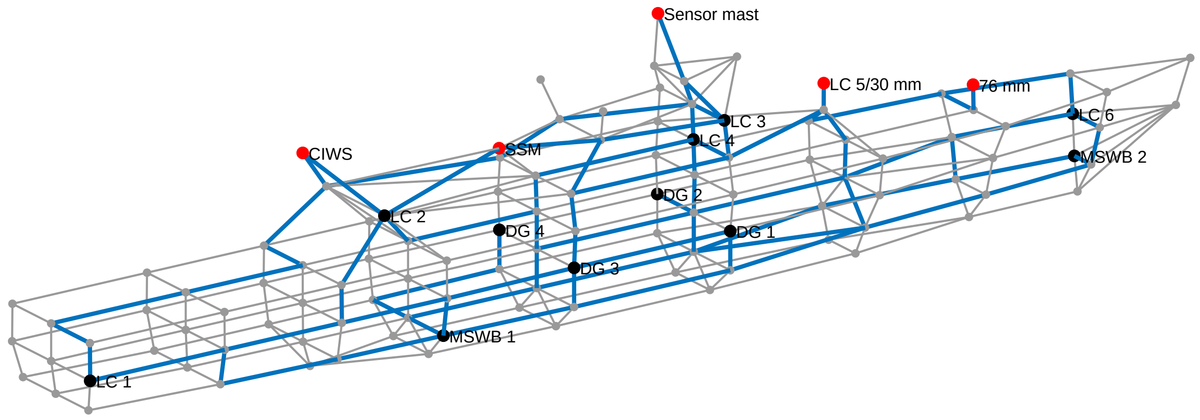


Figure D.9: Physical layout testcase 3, topology ID 766, routing ID 420 (least vulnerable design on Pareto-front).

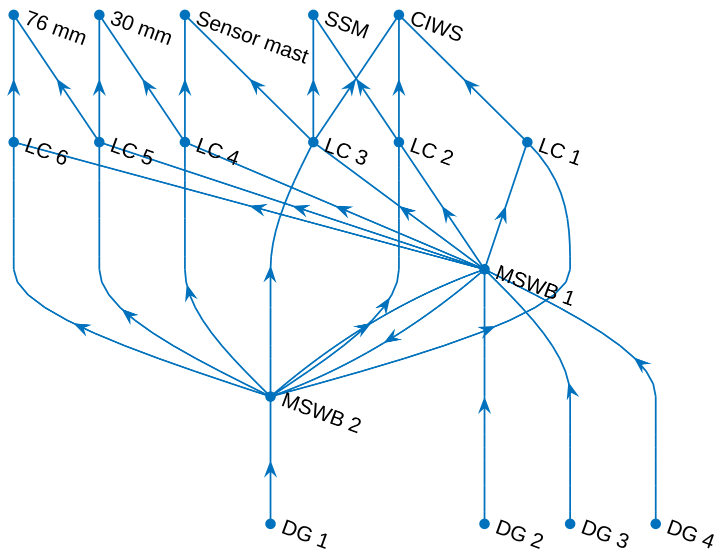


Figure D.10: Topology testcase 3, topology ID 766, routing ID 420 (least vulnerable design on Pareto-front).

D.3.2. Least expensive design

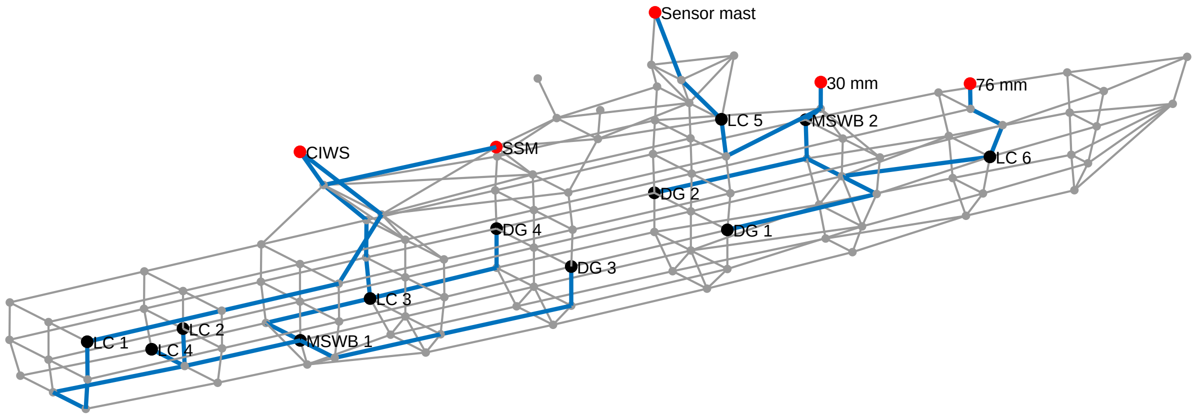


Figure D.11: Physical layout testcase 3, topology ID 693, routing ID 371 (least expensive design on Pareto-front).

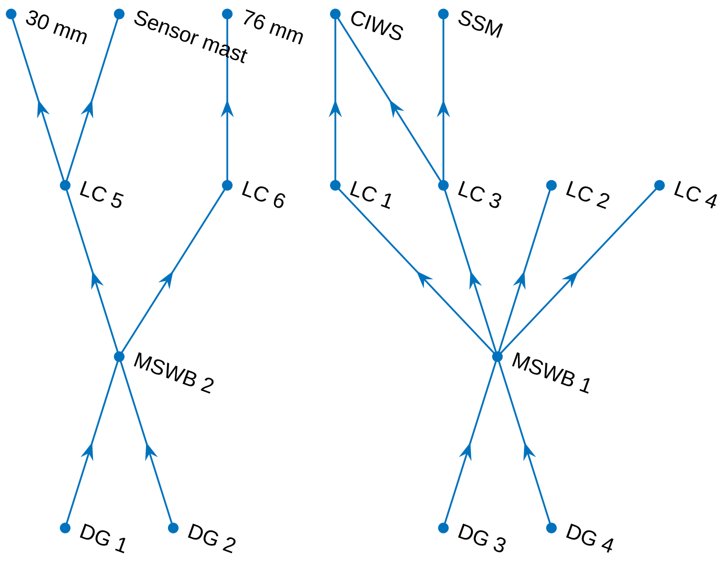


Figure D.12: Topology testcase 3, topology ID 693, routing ID 371 (least expensive design on Pareto-front).

D.4. Testcase 4

D.4.1. Least vulnerable design

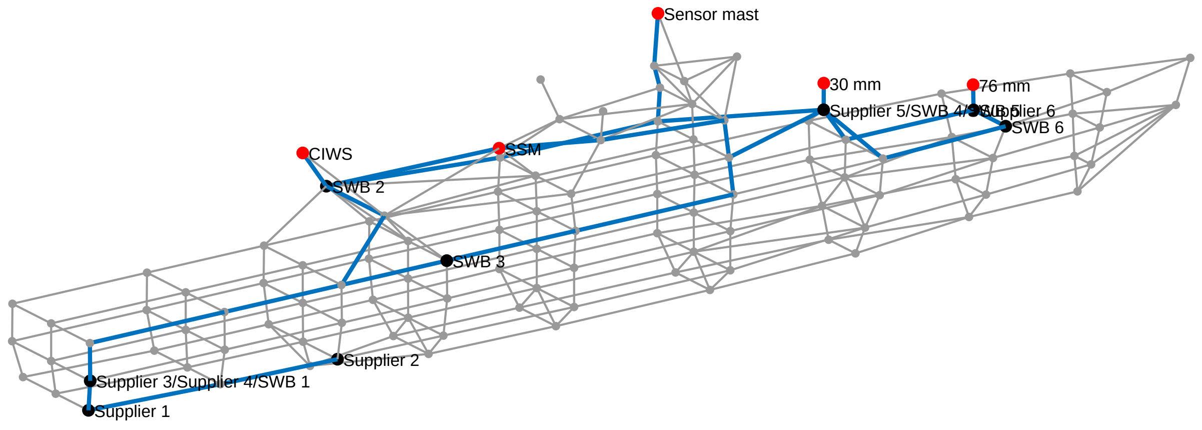


Figure D.13: Physical layout testcase 4, topology ID 2764, routing ID 399 (least vulnerable design on Pareto-front).

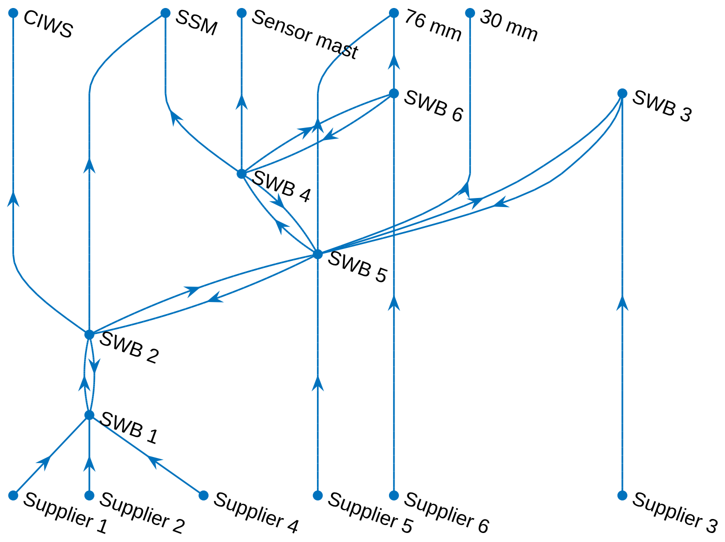


Figure D.14: Topology testcase 4, topology ID 2764, routing ID 399 (least vulnerable design on Pareto-front).

D.4.2. Least expensive design

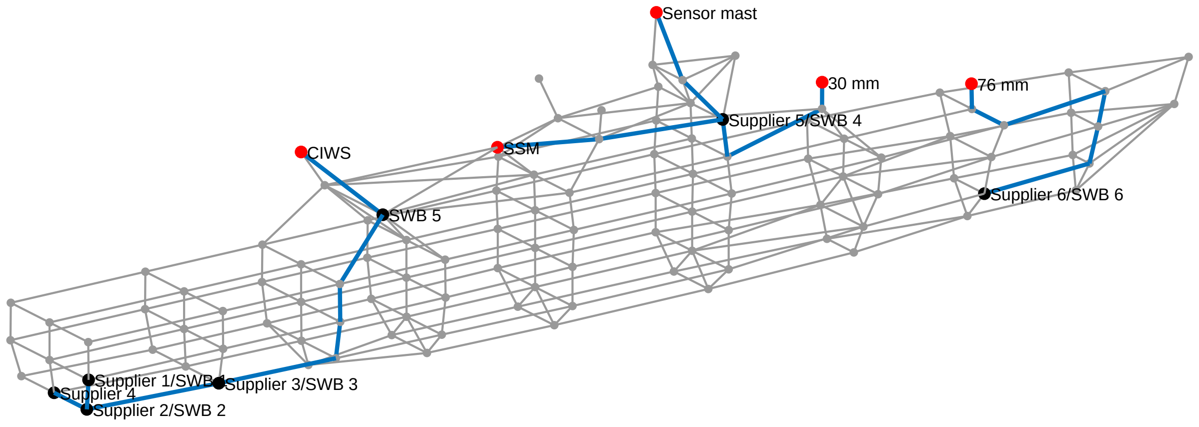


Figure D.15: Physical layout testcase 4, topology ID 2901, routing ID 233 (least expensive design on Pareto-front).

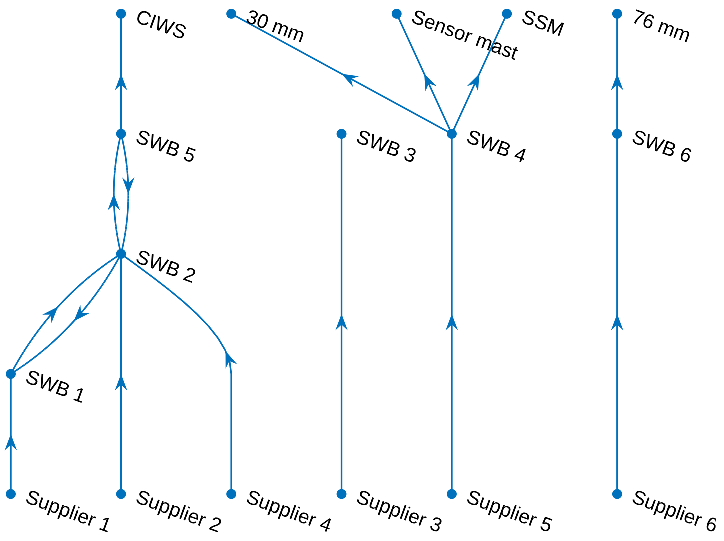


Figure D.16: Topology testcase 4, topology ID 2901, routing ID 233 (least expensive design on Pareto-front).

D.5. Testcase 5

D.5.1. Least vulnerable design

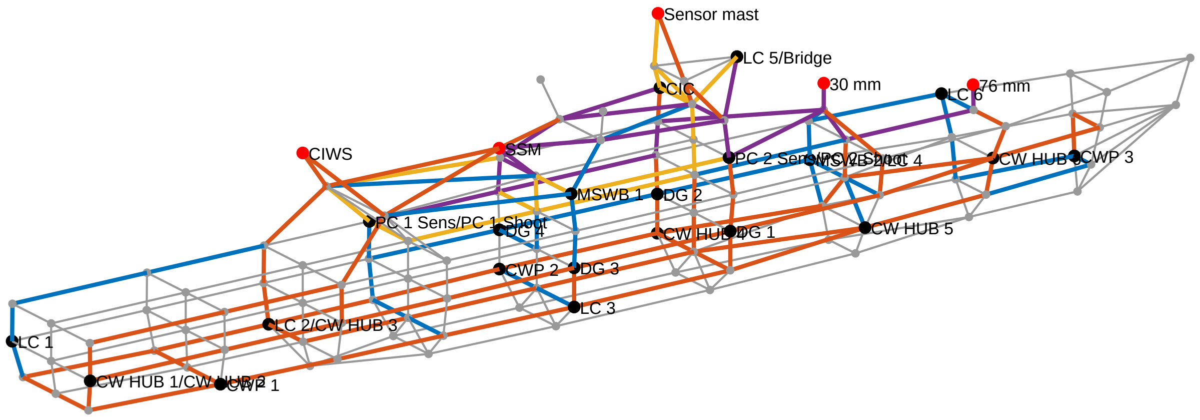


Figure D.17: Physical layout testcase 5, topology ID 610, routing ID 908 (least vulnerable design on Pareto-front).

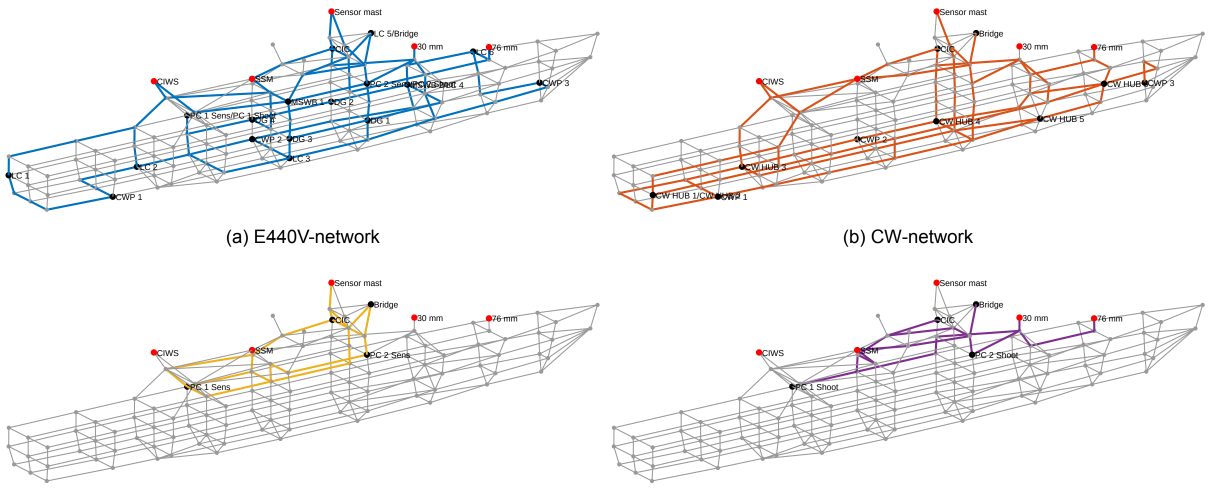


Figure D.18: Physical layout per network testcase 5, topology ID 610, routing ID 908 (least vulnerable design on Pareto-front).

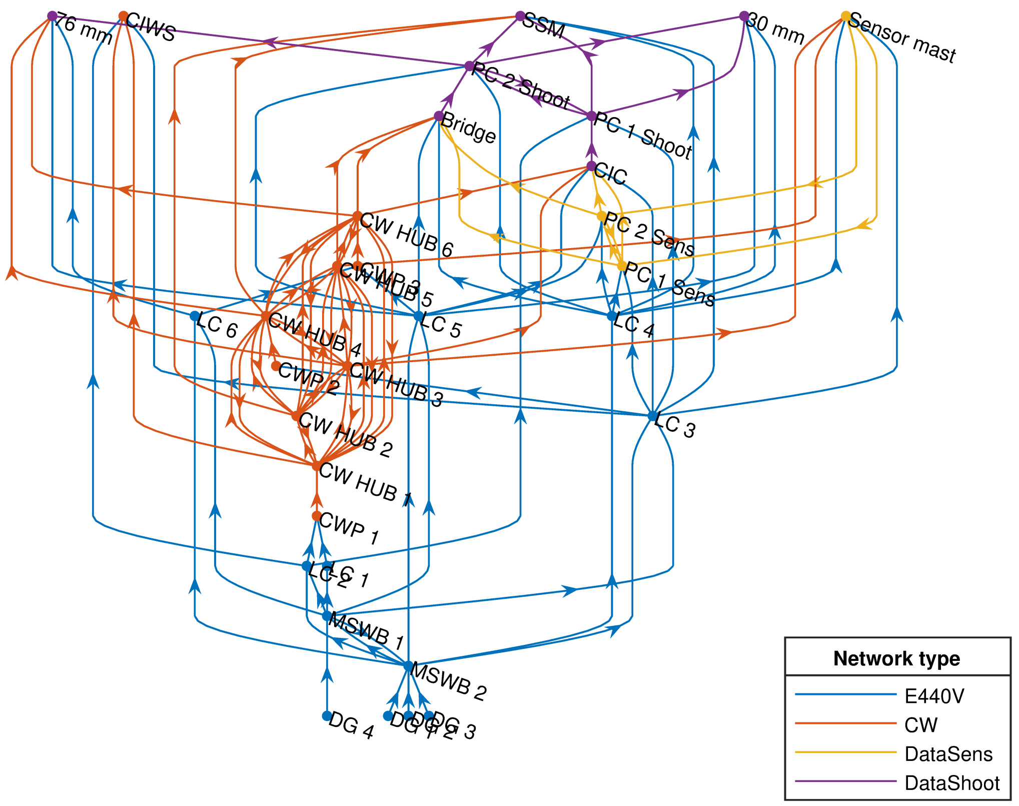
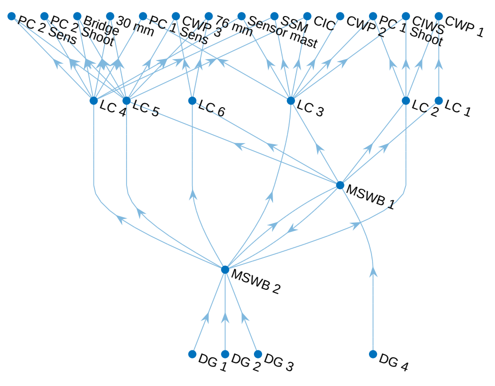
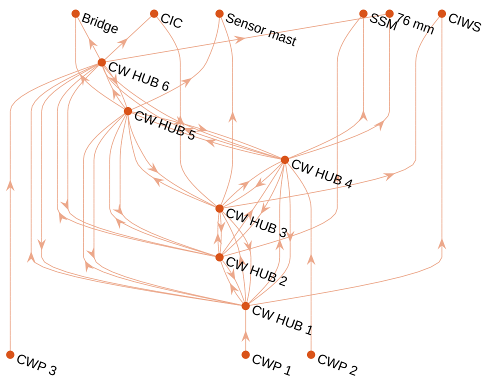


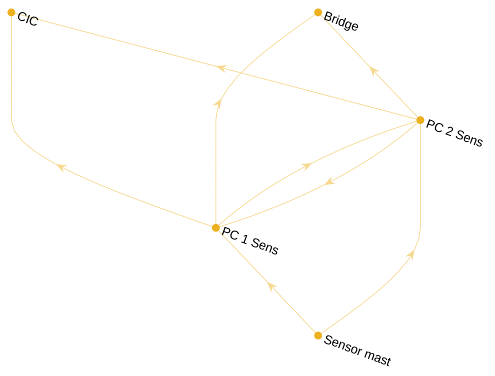
Figure D.19: Topology testcase 5, topology ID 610, routing ID 908 (least vulnerable design on Pareto-front).



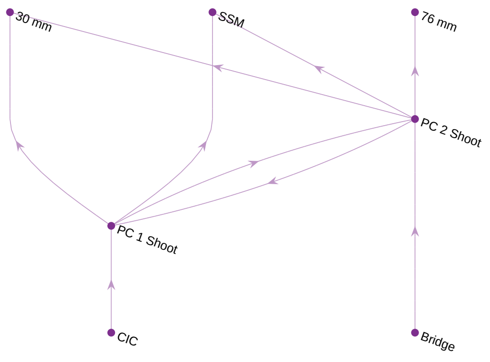
(a) E440V-network



(b) CW-network



(c) DataSens-network



(d) DataShoot-network

Figure D.20: Topology per network testcase 5, topology ID 610, routing ID 908 (least vulnerable design on Pareto-front).

D.5.2. Least expensive design

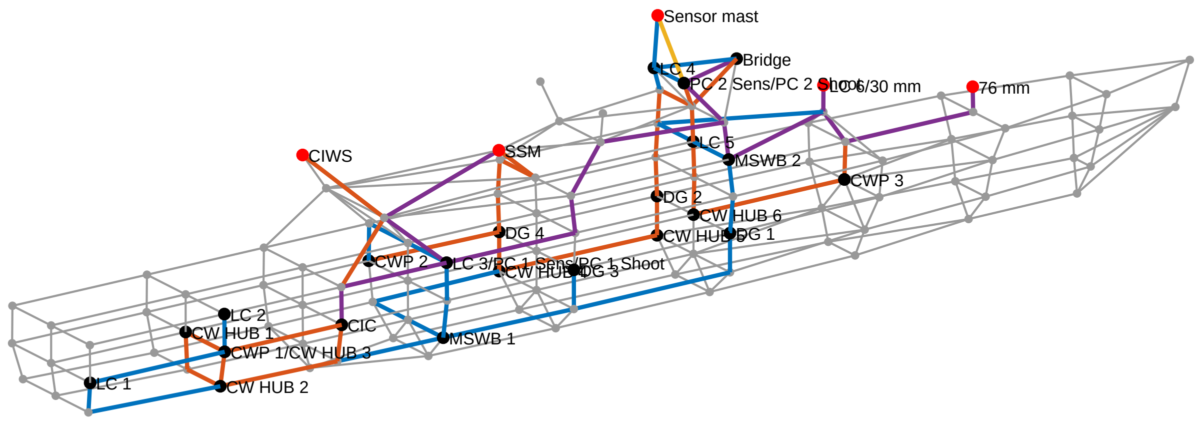


Figure D.21: Physical layout testcase 5, topology ID 2569, routing ID 603 (least expensive design on Pareto-front).

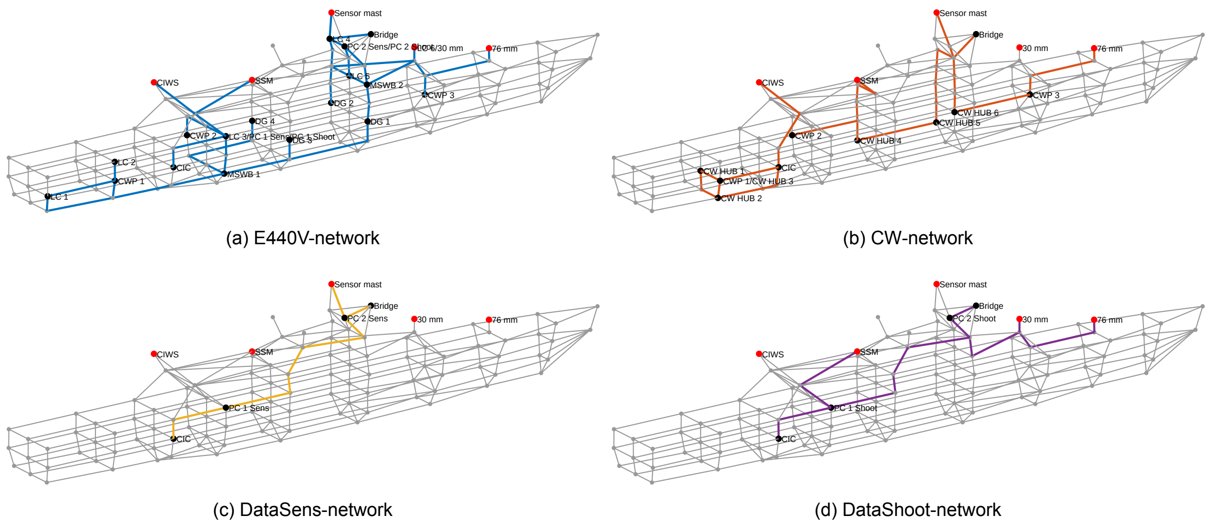


Figure D.22: Physical layout per network testcase 5, topology ID 2569, routing ID 603 (least expensive design on Pareto-front)

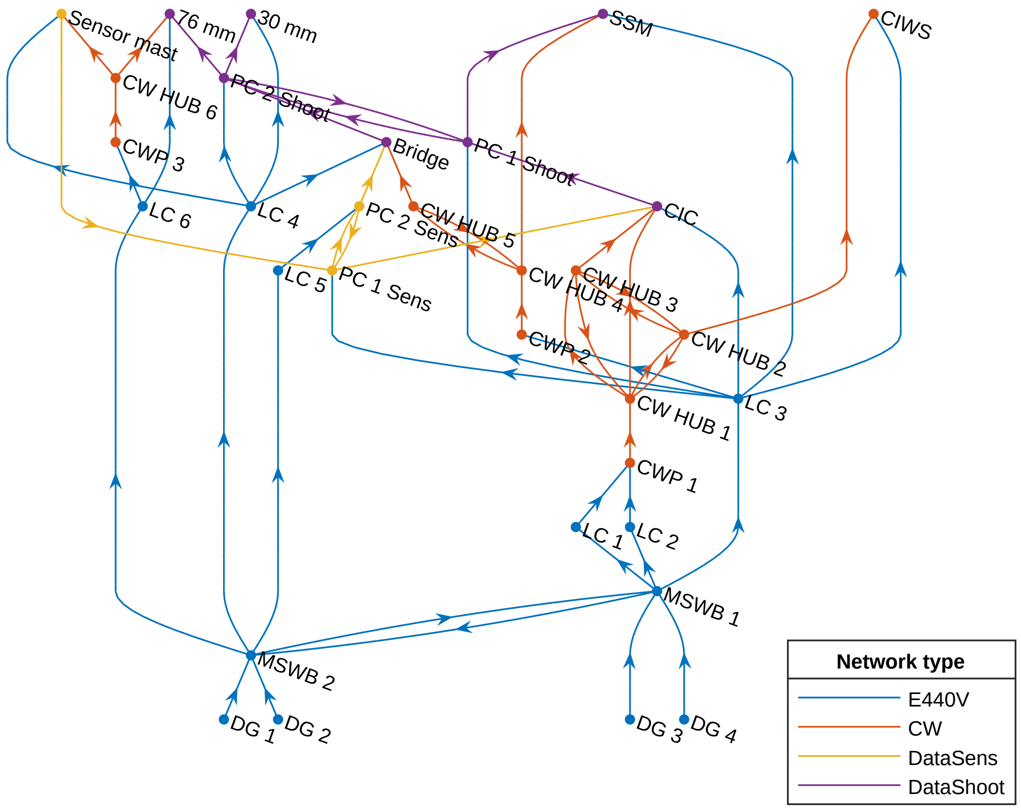
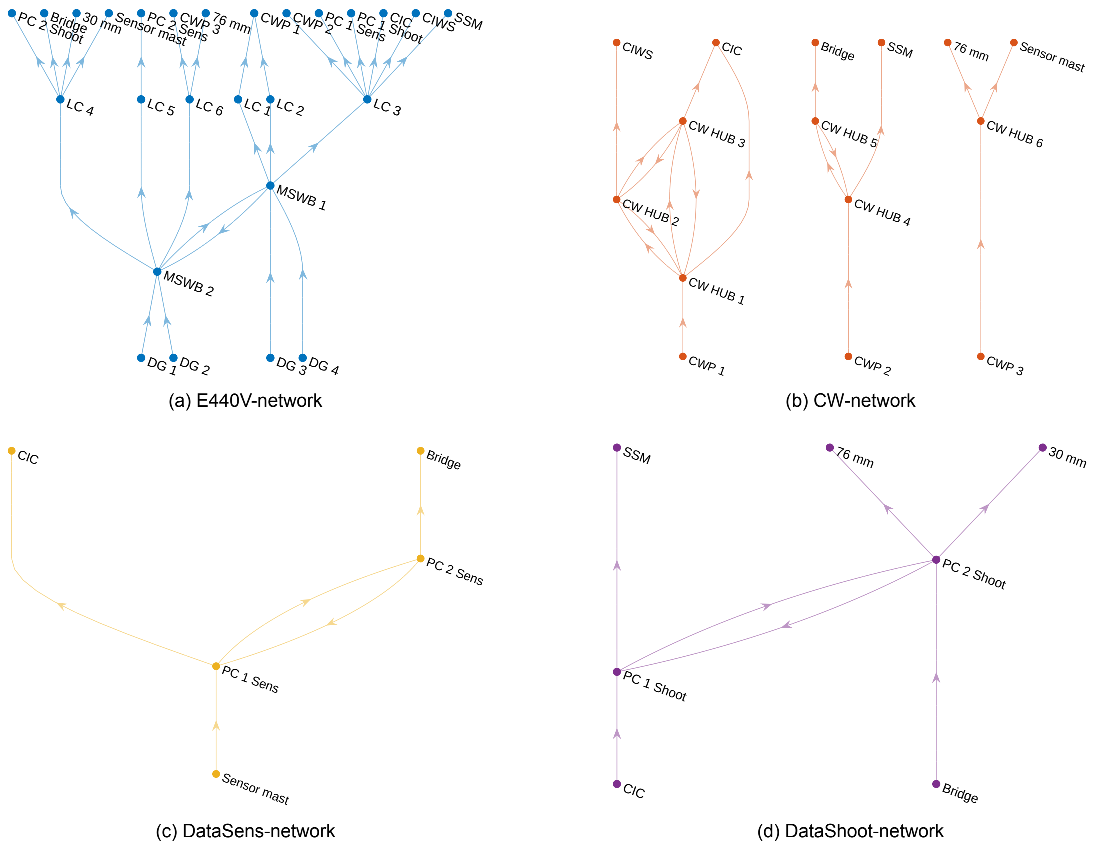


Figure D.23: Topology testcase 5, topology ID 2569, routing ID 603 (least expensive design on Pareto-front).



(a) E440V-network

(b) CW-network

(c) DataSens-network

(d) DataShoot-network

Figure D.24: Topology per network testcase 5, topology ID 2569, routing ID 603 (least expensive design on Pareto-front)

E

Directions for users

This appendix contains recommendation on how input should be formulated.

E.1. Physical input

- Check the subdivision and routing adjacency network for unintended SPF. The SPF may not obvious (e.g. the node connected directly to the node with degree 1), but is related to the damage size and number of hits.
- When possible, define spaces with the same resolution in a particular direction. When automatically determining the affected spaces following a hit with a large damage extent, the model considers the neighbouring spaces damaged as well. If the dimensions of the spaces are not similar, this will result in significantly different damage extents resulting from the same threat which is unrealistic. There are 2 solutions to avoid this: as said, when possible try to give all spaces the same dimensions. If for example all spaces are approximately 10 meters in X-direction, and 5 meters in Z-direction, spaces which significantly differ from these dimensions need to be further subdivided. The second possibility (as used for the hangar in the testcases) is to manually correct the damage case lists for large damages.

E.2. System input

- Check the system input for unintended SPF. When the generated solutions are performing significantly worse than expected, the most likely reason is the existence of a SPF in the system configuration.
- Components which occur in multiple networks in the same role (e.g. hubs in data networks) need to be modelled as separate components. The positions of the components need to be coupled when the positions of the component is variable. This coupling might also be considered for the topological connections when the type of connection is non-directional (e.g. for data connections).
- Definition of component names and capabilities should be exactly the same in all input to avoid errors.

Bibliography

Andrews, D.J. A Comprehensive Methodology for the Design of Ships (and Other Complex Systems). *Proceedings: Mathematical, Physical and Engineering Sciences*, 454(1968):187–211, 1998.

Andrews, D.J. et al. IMDC State of the Art Report - Design for layout, 2012.

Belcher, M. Survivability Primer: An Introduction to Naval Combat Survivability, 2008.

Blum, C. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353–373, 2005.

Brefort, D. et al. An architectural framework for distributed naval ship systems. *Ocean Engineering*, 147:375–385, 2018.

Brown, A.J. Application of Operational Effectiveness Models in Naval Ship concept exploration and design. *Ciencia y tecnología de buques*, 7(13):9, 2013.

Brown, A.J. *Ship and Marine Engineering Systems Concept Design*. 2018.

Brown, A.J. and Kerns, C. Multi-objective optimization in naval ship concept design. In *Marine Systems and Technology (MAST) 2010 Conference, Rome, Italy*, pages 9–11, 2010.

Brown, A.J. and Waltham-Sajdak, J. Still Reengineering the Naval Ship Concept Design Process. *Naval Engineers Journal*, 127(1), 2015.

Brown, D.K. No In Harm's Way: Warship Vulnerability 1939-98, Experience and Statistics, 1997.

Chondrogiannis, T. et al. *Alternative Routing: k-Shortest Paths with Limited Overlap*. nov 2015.

Chondrogiannis, T. et al. Finding k-shortest paths with limited overlap. *The VLDB Journal*, 2020.

de Vos, P. *On early-stage design of vital distribution systems on board ships*. PhD Thesis, Delft University of Technology, 2018.

de Vos, P. and Stapersma, D. Automatic topology generation for early design of on-board energy distribution systems. *Ocean Engineering*, 170:55–73, 2018.

Deb, K. and Agrawal, R.B. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

Deb, K. et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

Dekking, F.M. et al. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media, 2005. ISBN 1852338962.

Dijkstra, E.W. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

Droste, K., Kana, A.A. and Hopman, J.J. Process-based analysis of arrangement aspects for configuration-driven ships. In *13th International Marine Design Conference*, pages 327–337. CRC Press, 2018.

Duchateau, E.A.E. *Interactive evolutionary concept exploration in preliminary ship design*. PhD Thesis, Delft University of Technology, 2016.

Duchateau, E.A.E., de Vos, P. and van Leeuwen, S.P. Early stage routing of distributed ship service systems for vulnerability reduction. In *13th International Marine Design Conference*, Helsinki, 2018.

Gates, P.J. *Surface warships : an introduction to design principles*. London ; Washington : Brassey's Defence Publishers, 1st english edition, 1987. ISBN 0080347533.

Goodfriend, D.B. *Exploration of System Vulnerability in Naval Ship Concept Design*. MSc Thesis, Virginia Polytechnic Institute and State University, 2015.

Goodrum, C.J., Shields, C.P.F. and Singer, D.J. Understanding cascading failures through a vulnerability analysis of interdependent ship-centric distributed systems using networks. *Ocean Engineering*, 150:36–47, 2018.

Habben Jansen, A.C., Kana, A.A. and Hopman, J.J. An approach for an operational vulnerability assessment for naval ships using a Markov model. In *13th International Marine Design Conference*, volume 2, pages 1073–1082. CRC Press, 2018. ISBN 9781138541870.

Habben Jansen, A.C. et al. A Framework for Vulnerability Reduction in Early Stage Design of Naval Ship Systems. In *Intelligent Ship Symposium*, 2019a.

Habben Jansen, A.C. et al. Assessing complex failure scenarios of on-board distributed systems using a Markov chain. *Journal of Marine Engineering & Technology*, pages 1–17, 2019b.

Habben Jansen, A.C., Kana, A.A. and Hopman, J.J. A Markov-based vulnerability assessment for the design of on-board distributed systems in the concept phase. *Ocean Engineering*, 190, 2019c. ISSN 00298018.

Hillier, F.S. and Lieberman, G.J. *Introduction to Operations Research*. McGraw-Hill Education, New York, US, 10th inter edition, 2015.

Kerns, C., Brown, A.J. and Woodward, D. Application of a DoDAF total-ship system architecture in building a design reference mission for assessing naval ship operational effectiveness. In *ASNE Global Deterrence and Defense Symposium*, pages 13–14. Citeseer, 2011.

Koç, Y. *On Robustness of Power Grids*. PhD thesis, jan 2015.

Mattsson, L.G. and Jenelius, E. Vulnerability and resilience of transport systems – A discussion of recent research. *Transportation Research Part A: Policy and Practice*, 81:16–34, 2015.

NL MoD. *Fundamentals of Maritime Operations*. Royal Netherlands Navy, 2014.

Ouyang, M. A mathematical framework to optimize resilience of interdependent critical infrastructure systems under spatially localized attacks. *European Journal of Operational Research*, 262(3):1072–1084, 2017.

Ouyang, M. et al. Vulnerability Mitigation of Multiple Spatially Localized Attacks on Critical Infrastructure Systems. *Computer-Aided Civil and Infrastructure Engineering*, 33(7):585–601, jul 2018.

Parsons, M.G. et al. Intelligent Ship Arrangements: A New Approach to General Arrangement. *Naval Engineers Journal*, 120(3):51–65, dec 2008.

Pedersen, K. et al. The"Validation Square"-Validating Design Methods. (January), 2000.

Piperakis, A.S. *An integrated approach to naval ship survivability in preliminary ship design*. PhD Thesis, University College London, 2013.

Said, M.O. Theory and practice of total ship survivability for ship design. *Naval Engineers Journal*, 107 (4):191–203, 1995.

Seyler, S.L. et al. Path Similarity Analysis: A Method for Quantifying Macromolecular Pathways. *PLOS Computational Biology*, 11(10), oct 2015.

Shields, C. et al. *Design space exploration for shipboard distributed systems in concept design*. jul 2018.

Shiflet, A.B. and Shiflet, G.W. *Introduction to Computational Science: Modeling and Simulation for the Sciences - Second Edition*. Princeton University Press, 2006. ISBN 978-0691125657.

Singer, D.J., Doerry, N. and Buckley, M.E. What Is Set-Based Design? *Naval Engineers Journal*, 121 (4):31–43, 2009.

Singhal, N. CS273: Algorithms for Structure and Motion in Biology, 2004.

Spruit, J.P. et al. Survivable naval platform distribution systems and their automation. In *Proc. of Engine as a Weapon III*, London, 2009. IMarest UK.

Stark, S.A. *Definition of Damage Volumes for the Rapid Prediction of Ship Vulnerability to AIREX Weapon Effects*. MSc Thesis, Virginia Polytechnic Institute and State University, 2016.

Streppel, G. *Optimalisatie Restcapaciteit: Ontwikkeling van een Kwantificeringsmethode en Ontwerpregels*. MSc Thesis, Delft University of Technology, 2004.

US DoD. Universal Naval Task List (UNTL) - OPNAVINST 3500.38B/MCO3500.26A/USCG COMDTINST 3500.1B version 3.0, 2007.

US DoD. *DoD Dictionary of Military and Associated Terms (as of November 2019)*. 2019.

van Leeuwen, S.P. *Estimating the vulnerability of ship distributed system topologies*. MSc Thesis, Delft University of Technology, 2017.

van Meurs, R. Integrating Vulnerability in Ship Design, 2019.

Van Oers, B.J. *A Packing Approach for the Early Stage Design of Service Vessels*. PhD Thesis, Delft University of Technology, 2011.

van Oers, B.J. et al. Warship Concept Exploration and Definition at The Netherlands Defence Materiel Organisation. *Naval Engineers Journal*, 130(2):61–82, 2018.

Wagner, M. Nested Multi- and Many-Objective Optimization of Team Track Pursuit Cycling, 2016.

Yen, J.Y. Finding the K Shortest Loopless Paths in a Network. *Management Science*, 17(11):712–716, may 1971.

Zimmerman, R. Decision-making and the vulnerability of interdependent critical infrastructure. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 5, pages 4059–4063 vol.5, 2004. ISBN 1062-922X VO - 5.

Zitzler, E., Laumanns, M. and Thiele, L. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report*, 103, 2001.

Acronyms

AAW

Anti-Air Warfare.

ACO

Ant Colony Optimization.

AHP

Analytical Hierarchy Process.

ASuW

Anti-Surface Warfare.

ASW

Anti-Submarine Warfare.

ATG

Automatic Topology Generation.

CIC

Combat Information Centre.

CIWS

Close-In Weapon System.

CODLAD

Combined Diesel-Electric and Diesel Engine.

CODLOG

Combined Diesel-Electric or Gas Turbine.

COG

Centre of Gravity.

CW

Chilled Water.

CWP

Chilled Water Plant.

DBB

Design Building Block.

DC

Damage Control.

DE

Diesel Engine.

DG
Diesel Generator.

DMO
Defence Materiel Organisation.

DRM
Design Reference Mission.

DUT
Delft University of Technology.

DV
Design Variable.

FC
Fuel Cell.

FCS
Fire Control Solution.

FIDES
Functional Integrated Design Exploration of Ships.

GA
Genetic Algorithm.

ISA
Intelligent Ship Arrangements.

kSPwLO
k-Shortest Path with Limited Overlap.

LC
Load Centre.

MOD
Ministry of Defence.

MOE
Measure of Effectiveness.

MOGO
Multi-Objective Genetic Optimization.

MOP
Measure of Performance.

MOTISS
Measure of Total Integrated Ship Survivability.

MSWB
Main Switch Board.

NAVSEA
Naval Sea Systems Command.

NMET
Naval Mission Essential Task.

NSGA-II
Non-dominated Sorting Genetic Algorithm II.

OEM
Operational Effectiveness Model.

OMOE
Overall Measure of Effectiveness.

OMOV
Overall Measure of Vulnerability.

OpSit
Operational Situation.

OPV
Ocean Going Patrol vessel.

PDF
Probability Density Function.

PSA
Path Similarity Analysis.

RCM
Residual Capability Matrix.

RPG
Rocket Propelled Grenade.

SDB
Subdivision Block.

SPEA-2
Strength Pareto Evolutionary Algorithm-2.

SPF
Single Point of Failure.

SR
Short Range.

SSM
Surface-to-Surface Missile.

SURMA
Survivability Management Application.

SWB

Switch Board.

UCL

University College London.

UNTL

Universal Naval Task List.

VLS

Vertical Launch System.

VT

Virginia Polytechnic Institute and State University.

Glossary

Compartment

A collection of all spaces in a zone-deck..

Connection

A link between two components in the logical layer.

Effector

A component in a distributed system which ultimately brings an operational effect, such as sensor and weapon systems.

Path

A physical implementation/solution of a connection.

Routing

The process of generating a physical implementation of a connection.

Survivability

The capability of a ship and its shipboard systems to avoid and withstand a weapons effects environment without sustaining impairment of their ability to accomplish designated missions.

Susceptibility

The inability of a ship to avoid being damaged in the pursuit of its mission and to its probability of being hit.

Vulnerability

1) The inability of a ship to withstand damage after a sustained hit.

2) The extent to which the operational performance degrades due to the inability of the ship or system to deal with damage after being hit.