NTNU

Norwegian University of
Science and Technology

An STPA Analysis of the ReVolt

Expanding and Improving the System-
Theoretic Process Analysis (STPA)
Framework

Christine Lovise Solberg

Master of Science in Cybernetics and Robotics
Submission date: June 2018
Supervisor: Tor Engebret Onshus, ITK
Co-supervisor: Jon Arne Glomsrud, DNV GL

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Preface

This thesis is written as a part of a collaborative project between NTNU and DNV GL, the ReVolt. The ReVolt is a small soon-to-be autonomous boat, developed by students from the cybernetics department at NTNU. The objective of thesis is to investigate the safety aspects of the project. DNV GL requested an STPA hazard analysis of the ReVolt to explore the potential of this method. The project thesis, carried out last semester, was used to become familiar with the theory behind STPA, and learning how to carry out the steps of the analysis. Instead of using the tables presented in the STPA guidelines, I used tables in Excel with a framework designed by co-supervisor, Jon Arne Glomsrud. Additionally, I spent a large amount of time understanding the ReVolt, and learning its design. Two students designed and implemented the software on the ReVolt, as a part of their master thesis last year. The documentation provided by these students is quite inadequate.

Spending a lot of time being frustrated by the lack of thorough documentation and poorly commented code, I realized that providing system documentation is essential. As a result, I proposed adding this to be the focus of the Master thesis. Suggesting that documentation of the system software, hardware and behaviour must be in place before a safety analysis can be performed. I went through the process of selecting suitable documentation tools. Since I was not given a description of the work I was expected to perform in my thesis, I presented various ideas of my own to the supervisors. Together, we agreed that it would be reasonable to limit the STPA analysis to a single

controller, instead of the entire system, since the documentation process was added to the assignment. A single controller is sufficient for the concept demonstration. An analysis of the entire system will become very large.

After meeting with co-supervisor, Glomsrud, in February I learned that he had improved the Excel framework that was used in the project thesis. We agreed that it would be interesting to perform the low-level STPA analysis of the ReVolt once more. The results are significantly improved. The scope of the thesis has therefore been modified and expanded since the beginning of the semester.

I have been given access to the ReVolt, so that I can study the hardware mounted inside and on the vessel. Some of the hardware and electronics onboard have been placed inside boxes, so that it cannot be studied closely. I have been given access to the software from the summer of 2017, and this is what has been used to document the software. I have not needed any equipment or software license to perform the work presented in this thesis, except for Microsoft Excel, which NTNU provides.

I have met with my main supervisor five times throughout the semester to briefly discuss the progress of my work. My co-supervisor has provided the STPA Excel framework, and has been available for questions regarding STPA. I would like to thank them for the help and support throughout this semester. However, I want to emphasize that I have performed the work presented in this thesis independently.

Trondheim, June 4^{th}, 2018,
Christine Lovise Solberg

Abstract

The ReVolt is an idea developed by DNV GL. It is an autonomous vessel designed for container transport between harbors. To test the concept, a co-operational project was started with NTNU. Two master students from the cybernetics department developed a smaller version of the ReVolt. DNV GL provided them with the hull itself, as well as the thruster design. The remaining components are chosen by these students, and they have implemented functionality for dynamic positioning and remote control. Because this project was successful, DNV GL have chosen new students that will continue working on the ReVolt. It is intended to become autonomous within the near future.

Safety is an important aspect of an autonomous vessel. If safety cannot be guaranteed, the vessel cannot be used. Therefore, DNV GL requested that a safety analysis is performed of the ReVolt. The method chosen to perform the analysis is called STPA (System-Theoretic Process Analysis). This analysis is the main focus of this master thesis. Using the available information, and the future plans for the ReVolt, an STPA analysis has been performed.

Due to the lack of documentation from the developers of the ReVolt, a significant amount of time was spent trying to develop the required documentation in this thesis, based on the available information. UML diagrams were used to document system behavior and functionality. Adding the step of documentation to the STPA approach has been proposed. For the STPA analysis, a framework developed by the co-supervisor

was tested and evaluated. The framework provided good support to the analyst, and helped make sure no details were ignored in the analysis.

STPA is a safety analysis tool that is well suited for a complex system as the ReVolt. Several design guidelines were extracted from the analysis results, even though time limited the level the analysis was taken to. The fact that the analysis was performed by a single person, most likely reduced the quality of the analysis. A team of people is required to ensure a satisfactory analysis. Still, the documentation that was developed greatly contributed to performing a thorough system analysis. The results from the analysis presents three main areas for system improvement for the ReVolt at this time; loss of communication, incorrect or missing sensor measurements, and time delays in the reference channels.

Sammendrag

ReVolt er en idé utviklet av DNV GL. Den er et selvstyrt fartøy designet for containertransport mellom havner. For å teste konseptet ble det startet et samarbeidsprosjekt med NTNU. To masterstudenter fra kybernetikk-avdelingen utviklet en mindre versjon av ReVolt. DNV GL ga dem selve skroget, samt thruster-designet. De resterende komponentene er valgt av studentene selv, og de har implementert funksjonalitet for dynamisk posisjonering og styring med fjernkontroll. Fordi dette prosjektet var vellykket, har DNV GL valgt nye studenter som skal fortsette å jobbe med ReVolt. Intensjonen er at ReVolt skal bli autonom innen nær fremtid.

Sikkerhet er et viktig aspekt for et selvstyrt fartøy. Hvis sikkerheten ikke kan garanteres, kan ikke fartøyet brukes. Derfor anmodet DNV GL at en sikkerhetsanalyse skulle utføres av ReVolt. Metoden som er valgt for å utføre analysen kalles STPA (System-Theoretic Process Analysis). Denne analysen har vært hovedfokuset for denne masteroppgaven. Ved å bruke tilgjengelig informasjon, og fremtidige planer for ReVolt, har en STPA-analyse blitt utført.

På grunn av mangel på dokumentasjon fra utviklerne av ReVolt, ble det brukt en betydelig mengde tid på å utvikle denne dokumentajonen i denne oppgaven, basert på tilgjengelig informasjon. UML diagrammer ble brukt til å dokumentere systemadferd og funksjonalitet. Å legge til trinnet med dokumentasjon i STPA-fremgangsmåten er foreslått. For STPA-analysen ble et rammeverk som er utviklet av medveileder testet

og evaluert. Rammeverket ga god støtte for den som utførte analysen, og sørget for at ingen detaljer ble ignorert.

STPA er et sikkerhetsanalyseverktøy som passer godt til et komplekst system som ReVolt. Flere retningslinjer for systemdesignet ble hentet fra analyseresultatene, selv om tiden begrenset nivået analysen kunne tas til. Det faktum at analysen ble utført av en enkelt person, reduserte sannsynligvis kvaliteten på analysen. Et team av mennesker er nødvendig for å sikre en fullverdig analyse. Dokumentasjonen som ble utviklet, bidro imidlertid godt til å utføre en grundig systemanalyse. Resultatene fra analysen presenterer foreløbig tre hovedområder for potensiell systemforbedring for ReVolt; tap av kommunikasjon, feil eller manglende sensormålinger og tidsforsinkelser i *referansekanaler*.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In the project assignment carried out the fall of 2017 (Solberg; 2017), an STPA analysis was performed of the ReVolt. The **ReVolt** is a concept ship, designed by DNV GL. The ReVolt is intended to become a fully autonomous ship, for transportation of containers. Last year, two students from NTNU built and developed a small model version of the ReVolt, for a test of concept. At this time, the ReVolt model is not fully autonomous, but (Alfheim and Muggerud; 2017a) developed functionality for DP (dynamic positioning) and remote control using RC (radio communication) in their Master thesis. This year, two new students have continued the development of the ReVolt. One working with sensor fusion, installing to cameras and a Lidar on the ReVolt. The other student is working with path planning and obstacle avoidance. Both working towards the goal of an autonomous ReVolt.

DNV GL requested an **STPA** analysis of the ReVolt to be performed. STPA (System-Theoretic Process Analysis) is a hazard analysis technique which is used to define safety constraints and design guidelines to ensure safe behavior of a system. STPA was developed by Prof. Nancy Leveson (MIT) in the early 2000's, and is designed for

analyzing modern, complex systems. In the project thesis, an STPA analysis of the ReVolt was started. The analysis was only taken to level 3, which is far from a complete analysis. The levels will be explained later in this report. Performing an STPA analysis is a very comprehensive process which requires much time, which is the reason why the analysis was not taken to a higher level. Also, a large amount of time was spent trying to understand the design of the ReVolt, since the existing documentation had significant shortcomings.
Co-supervisor, Jon Arne Glomsrud had designed a framework in Excel for the STPA analysis. This framework was tested in the project thesis.

1.2 Problem Description

In this thesis, an extension that may improve the quality of the STPA analysis is suggested. *The more thoroughly a system is documented, the more orderly and straightforward the process of analyzing the system becomes. By providing proper documentation, the time required to do the analysis may be significantly reduced.* A set of **UML** (Unified Modelling Language) diagrams will be used to document the system behavior and the software onboard the ReVolt. The hardware will be described, and an **RDB** (Redundancy Block Diagram) will be presented. It will be argued whether this should be adopted as a preparatory step in the STPA analysis.

After the UML and hardware diagrams are drawn for the ReVolt, a new STPA analysis is performed. Glomsrud has, since the project assignment was delivered, updated the Excel framework for the STPA analysis. This framework will therefore be described and tested. There are no definitions stating how many levels of an STPA analysis is required for the analysis to be complete. It is different for all systems. It will be attempted to complete the analysis for a small part of the system. Completing the entire STPA analysis in a semester, would require too much time for a single person to be able to complete. Based on the results of the analyses, improvements to the safety design of the ReVolt are proposed. The results of the analysis will help determine what kind of changes are needed to improve the safety of the design. It may be e.g.

hardware monitoring, changes in the ROS communication, or any other part of the system.

Summarized:

1. Decide which UML diagrams are needed to sufficiently describe a system for STPA.
2. Create these UML diagrams for the ReVolt, to document the system behavior and the software.
3. Create documentation for the hardware of the ReVolt.
4. Using the documentation, perform a low-level STPA analysis of the ReVolt.
5. Choose a small part of the system and take the analysis as far as is required to be able to define concrete safety design constraints and guidelines.
6. Based on the results of the analysis, propose changes that may improve the safety of the ReVolt.
7. Argue whether the documentation should be included in the approach of the STPA analysis, and discuss whether the new framework provides any advantages for the analysis.

1.3 Motivation

1.3.1 ReVolt

The road network across Europe is heavily congested. Trailer transportation of goods is one of the major causes of the heavy traffic. Something needs to change. The motivation behind the ReVolt has several contributors. Taking parts of the container transportation to the seas, is one of them. It would relieve the road network of some stress.
The increasing focus on autonomous vessels is another important contributing factor.

DNV GL is a leading company within technology standards. Naturally, it is important to them to be a part of this leap in technology, and to get involved early on is essential. The cybernetics department at NTNU has another motivational factor behind the interest in the ReVolt. Based on the ReVolt model, they are planning to develop an autonomous shuttle passenger ferry across *Nidelva*. This is an area of great interest to the cybernetics department, that is currently hiring six Ph.D.s to work with the autonomous ferry. There are several interesting cases in this project, such as; the autonomy challenge itself, the fact that it is planned to have all-electric power, sensor tracking of obstacles, the challenge of cyber and communications security, how to handle remote monitoring and possibly control by human operators, and lastly the challenge of safety.

1.3.2 System-Theoretic Process Analysis - STPA

DNV GL wants to investigate the potential of STPA as a tool for safety analysis. They are experts within the area of rules and standards, and studying the potential of STPA can be of great value to them. As will be discussed later in the thesis, traditional safety analysis tools are becoming outdated. When applied to new, complex systems they are not capable of identifying all potential causes of accidents. As technology is advancing, it is important for a company such as DNV GL to keep up with the development of methods for safety analysis.

1.3.3 Personal Motivation

During a part-time job last summer, I was involved in the start-up phase of a pilot project. It was related to driver-less vehicles inside production halls. The company was hiring an external company to develop the product. In the process of selecting the company, all the companies presented PowerPoint's to explain why they would be the best fit. They all promised extraordinary results with low production costs. The problem was, none of them barely provided a single technical detail on their solution, or explained how they would guarantee for the safety of their vehicles. It was frustrating to watch such a high-technological challenge being "solved" using only

words and promises. It was impossible to make a qualified choice. During this process I learned the importance of the ability to express and illustrate technical solutions. If a single one of the companies had presented a draft of their solutions with a few UML diagrams, and hardware sketch, it would have been so much easier to understand what their solution had to offer. The fact that only a few of the companies even mentioned safety, made me realize the lack of focus on this important topic. Working with UML and STPA has been rewarding and has emphasized the importance of using tools like these in industry.

1.4 Abbreviations

- ADC - Analog-to-Digital Converter
- ESC - Electronic Speed Controller
- GNSS - Global Navigation Satellite System
- IMU - Inertial Measurement Unit
- INS - Inertial Navigation System
- Lidar - Light Detection and Ranging
- OBC - On-Board Computer
- RBD - Redundancy Block Diagram
- RC - Radio Communication
- RTK - Real-Time Kinematic
- SAHRA - STPA based Hazard and Risk Analysis
- STAMP - Systems-Theoretic Accident Model and Processes
- STPA - System-Theoretic Process Analysis

- UCA - Unsafe Control Action
- UML - Unified Modelling language

1.5 Outline

The report is organized as follows

- Chapter 2 presents the conclusion of the thesis.
- Chapter 3 gives a brief introduction to the theory behind STAMP, and introduces the hazard analysis technique STPA. The theory behind UML and RBD is also presented in this chapter.
- Chapter 4 presents the detailed step-by-step approach to STPA, as it is presented by Leveson. Next, the framework designed by Glomsrud is explained in detail.
- Chapter 5 introduces the ReVolt, and its technical details are explained.
- Chapter 6 explains the version of the ReVolt that has been used in this thesis, and UML and hardware documentation is presented.
- Chapter 7 shows the results of the work that has been performed in this thesis.
- Chapter 8 presents the discussion of the results from chapter 7.
- Chapter 9 briefly discusses the possibilities for future work on this topic.

Chapter 2

Conclusions

STPA is a hazard analysis technique designed for large, complex systems, enabling detection of all possible types of flaws. As a consequence, performing the analysis is challenging and very time-consuming. For someone who are new to the method, it might seem overwhelming at first. There are two main keys to being able completing a complete an STPA analysis.

The first key is a proper framework that provides guidelines to the analyst. Even though the result of the analysis greatly depends on the knowledge and skills of the analyst, the framework can provide a supporting structure. In this thesis, the framework developed in Excel by Jon Arne Glomsrud has been presented and tested. The framework provided the analyst with great support. It helped in showing the next natural step along the process. A few details were discussed for possible improvements to the framework, but these are not very significant. Another advantage of the Excel framework is that it reduces the number of tables that the analyst needs to keep track of, and everything is kept in one document. The original STPA tables are sufficient for performing the analysis, but the Excel table provides better support.

The second key is keeping thorough documentation of the system. In this thesis,

UML documentation has been proposed to document system behavior and functionality. UML consists of a wide range of different diagrams that can be used to visualize all the qualities and specifications of a system. A use-case diagram, a state machine diagram, an activity diagram and a sequence diagram have been chosen to describe the system behavior, and a class diagram is chosen to provide the system structure with information. Although, there are probably other alternatives that might be used for the same purpose. UML was chosen here because it is a widely used standard, and it was already familiar to the author of this thesis. Hardware documentation, however was unfamiliar territory. The hardware components and their connections have been documented and an RBD diagram has been presented. These provide a good system overview that the analyst can use while performing the STPA analysis. If the UML and hardware diagrams had been designed before the development of the ReVolt was started it would have been possible to perform an STPA analysis alongside the implementation of the ReVolt. Now, assumptions have had to be made on what the system will look like, to provide a somewhat useful analysis. However, the documentation is a great supplement to the analysis, and helped keep a structured overview of the system.

The ReVolt is a large, and extremely interesting project with great potential. Without an extensive focus on safety, this project can never be realized. Launching a driver-less boat onto the ocean without being able to guarantee safety, is a risk one cannot take. Up until now, the development process has not made it easy to confirm a safe system design. The system has not been adequately planned for, before implementation is started. This way, it is more likely to encounter challenges that have not been considered, resulting in design flaws which do not enhance the system safety. Ideally, the system should be redesigned and documented, before the implementation process is continued, or maybe even started over. Then a safety analysis can be performed alongside the implementation phase, to provide a safety-guided design. Detecting possible system flaws as early as possible will make the cost of fixing it as small as possible.

Chapter 3

Background Theory

3.1 Description of STAMP (Systems-Theoretic Accident Model and Processes) and STPA

In the early 2000's Nancy Leveson developed the framework for a new accident causation model; Systems-Theoretic Accident Model and Processes (STAMP). It was based on theories from modern systems thinking and systems theory. STAMP stood out from the traditional accident causation models in the way it viewed safety. To understand Leveson's framework it is important to provide clear definitions of the following terms: safety, reliability, accidents, accident causes and hazards.

> *Reliability: "The probability that something satisfies its specified behavioral requirements over time and under given conditions; that is does not fail." (Leveson; 2011). Meaning that a component, software or a system behaves as has been specified, and does not fail at this (CambridgeDictionary; 2017a).*

Previously, it had been assumed that safety and reliability were equivalent. A system composed of only reliable components would result in a reliable and safe system. The

traditional accident causation models assume that by avoiding failure of components, a system is considered safe. Leveson does not completely agree with this. According to Leveson, safety must be handled as a control problem. By applying the correct constraints to a system, the safety can be controlled. Contradictory to the older accident causation models, Leveson claimed that the having reliable system components did not imply safety of a system. Modern systems are much more complex and software-based than what they were before, and the nature of accidents are changing.

> *Safety: A system's ability to avoid accidents and losses. "A state in which you are safe and not in danger or at risk" (CambridgeDictionary; 2017b).*

To understand Leveson's methodology, it is just as important to understand what is considered *unsafe*. In fact, the first step of the method is defining what is considered unsafe for the system, by defining accidents and hazards. Traditional safety analysis techniques had never really defined what was unsafe, other than safety being the absence of accidents and losses.

> *Unsafe: Any state where the system is at a significant risk of being prone to an accident or loss. Whether a system is unsafe or not also depends on external factors. E.g. it is only unsafe for a vessel to head towards the shore if the distance is considered too small, and/or the speed to high, relative to the weather conditions, currents and the vessel's current maneuvering capacity.*
>
> *The state where a system is prone to accidents is called a* ***hazard****. A hazardous situation is where a system is considered unsafe. In a hazardous state, the system itself is no longer in control of whether an accident will occur.*

As long as a system avoids hazards, accidents cannot happen. If an accident happens, without being in a hazard, the system description is inadequate. It is crucial that all possible accidents and related hazards are defined.

> "*Accident: In STAMP, accidents are the result of a complex process that results in the system behavior violating the safety constraints.*" Leveson (2011)

To Leveson, safety is a control problem. Safety can be assured by avoiding hazards. The defined hazards can be avoided if the correct constraints are applied to the system.

From the STAMP framework, Leveson developed System-Theoretic Process Analysis, STPA. STPA is a hazard analysis technique based on the concepts introduced in STAMP. *Safety constraints*, *hierarchical control structures* and process models are central concepts for STPA.

- **Safety constraint** - Safety constraints are enforced on the system behavior to prevent hazards and accidents from occurring. The constraints appear as *rules* or guidelines applied to the system design.
- **Hierarchical Control Structure** - The control structure of a system is arranged in a tree-like structure. It contains the controllers, actuators and sensors of a system. Each controller is in control of the components below it. In Leveson's theory, each level in the hierarchical structure can impose safety constraints on the level below.
- **Process model** - Each controller must hold a model of the process it is controlling. The process model is used by the control structure to determine which control actions are required to ensure safety constraints are not violated.

STPA is a step-by-step approach to analyze the safety of complex systems. STPA can be applied before developing a system, to ensure it will be operating safely, or it can be applied to a system after an accident has happened, to help identify the cause(s) of the accident, or even alongside the development of a system.

The STPA analysis consists of three main steps:

0. Define system-level accidents, hazards and constraints. Design the hierarchical control structure of the system.
1. Define the unsafe control actions, that lead to the defined hazards.
2. Define the causal scenarios of the unsafe control actions. What allowed the given control action to be provided?
    - Based on the identified causes, system constraints must be defined that will prevent the unsafe control actions from being provided.

- Accidents will only happen if one of these constraints are somehow violated.

To perform these steps, the two following lists can be used to identify how unsafe control signals have been applied. How to use them will be explained in greater detail in Chapter 4.
"If an accident occurs, at least one of the following must have happened:

1. The safety constraints were not enforced by the controller.
   (a) The control actions necessary to enforce the associated safety constraint at each level of the socio-technical control structure for the system were **not provided**.
   (b) The necessary control actions were provided but at the **wrong time** (too early or too late) or stopped too soon.
   (c) Unsafe control actions were **provided** that caused a violation of the safety constraints.
2. Appropriate control actions were **provided, but not followed**." (Leveson; 2011).

The following list provides four types of system flaws that can cause unsafe control actions to be provided (Leveson; 2011):
1 - Unsafe Inputs: The unsafe control is caused by incorrect or missing input from the controller above it in the control structure.
2 - Unsafe Control Algorithms: There are flaws in the control algorithm which result in the algorithm being unable to enforce the required safety constraints. Also, if the system or process is modified without the control algorithm being modified accordingly, control flaws may occur. Note that controllers may also be human. In that case, required training, as well as providing clearly stated procedures for the operators will make up the control algorithm.
3 - Inconsistent, Incomplete or Incorrect Process Models: The process models

held by the controllers must correctly represent the actual process state. If the model deviates from the actual process it may cause unsafe control actions. For a human controller, the mental model is as important. The process model may be incorrectly updated due to incorrect or missing sensor data. Resulting in an incorrect belief of the current process state.
4 - Actuators and Controlled Processes: The commands that are given are safe, but they are not put into action. There may be various causes; An actuator or another component may be broken or without power. Also, the reference channels may not be working properly. If there are multiple controllers for one process, they may provide contradictory commands.

3.1.1 Why Use STPA?

STPA is developed to go beyond the traditional safety analysis techniques, which may ignore important accident causes such as inadequate requirements, software bugs and errors, and flaws in component interactions (Thomas; 2013). Traditional methods cannot prevent all of the accident causes that may occur in modern systems. Examples of the methods referred to as *traditional* are Failure Mode and Effect Analysis (FMEA), Fault Tree Analysis (FTA), Event Tree Analysis (ETA) and Hazard and Operability Analysis (HAZOP). These were very briefly introduced in the project thesis, and will not be discussed further here. There already exists a number of articles discussing how STPA differs from these methods. Most importantly, STPA can be applied at any stage of the development process, and it can find accident causes that traditional methods are not able to find. Traditional methods mainly focus on failures that can be solved with redundancy (by keeping backup components). STPA has a broader view on the range of accident causes. STPA is designed to analyze socio-technical systems, where many types of components, humans and machines, are operating together. (Thomas; 2013)

3.2 Unified Modelling Language - UML

UML is a graphical modelling language developed by the *Object Management Group*. UML is the notation for a set of diagram types that are used to model software. In total, there are 13 types of UML diagrams. The most commonly used diagram types are class diagrams, sequence diagrams, use case diagrams, state machine diagrams and activity diagrams. These will be explained in Chapter 6. Since UML was developed in 1997, it has become a standard for describing system software and functionality. UML can be considered a kind of programming language created to visualize systems, and to serve as an assistant to system developers in the process of communicating ideas. UML is an open standard, which is continuously being developed and and improved to keep up with modern software systems. (Fowler; 2004)

There had not been developed any UML documentation before or during the development of the ReVolt, which has made it challenging to work with for those taking over the project. In this thesis, a set of UML diagrams will be presented to describe the functionality and the software of the ReVolt. A subset of the existing UML diagram types will be chosen to supplement the STPA analysis.

3.3 Redundancy Block Diagram - RBD

Because the author was not familiar with any tools for hardware documentation, the supervisor recommended that RBDs are used for this purpose. This paragraph will provide some information about RBDs.
RBDs are visual representations of the operational relationships between the elements making up the systems (Denning; 2017). The diagrams are used to evaluate the reliability of a system. Even though STPA states that reliability does not imply safety, it does not exclude that redundancy is not necessary for some components. RBD provides a way of showing which hardware components must work for the system to function properly. Which components are critical to maintain normal operation? Figure 3.1 shows a simple example of an RBD. Objects connected in parallel illustrate

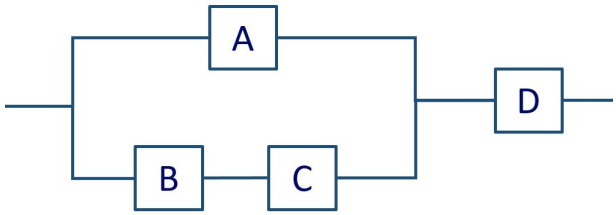

Figure 3.1: Simple example of a simple redundancy block diagram

the concept of redundancy. If block A stops working, B and C can still perform the necessary tasks. D, however, must work for the system to work properly. As of now, there are not many redundant functions on the ReVolt. Therefore, there will not be many diagrams to present at this stage.

3.4 Proposed Improvements to STPA

While working with the STPA method in the project assignment, several challenges were encountered. In the project assignment, a low-level STPA analysis was performed of the ReVolt. The ReVolt was built and developed by two students as a part of their Master thesis the spring of 2017. The focus of their thesis was to build the ReVolt, get the thrusters and the dynamic positioning algorithm to work. Safety was never considered as an important aspect of their thesis. This is often the case in most student projects, due to time being a limited resource. Finishing the product becomes the only priority, and making the product safe is forgotten. Not only students have a limited amounts of time to finish their products. This is also the case for working engineers. For a product to succeed, time to market can be crucial. While being in a hurry, it can be easy to ignore a small mistake, that can have fatal results if it is not handled properly. To be able to guarantee safer systems, it is important to have a proper framework that can help uncover these types of mistakes, that do not require more time than necessary.

During the project assignment, it was discovered that STPA can be an extremely

time-consuming process. The reason for this, in the ReVolt case, was essentially because of the lack of documentation. Having never seen the ReVolt before, not knowing the system specifications and details, it is difficult to predict any potential system accidents. A significant amount of time was committed to carefully reading and studying every single line of code behind the ReVolt, to learn how everything functioned.
As a result of this, it was realized of how little value STPA can be, if the system to be analyzed is not properly documented. Often, those who are making and designing the system are not the ones performing the hazard analysis. As a result of poor documentation, the analysis takes significantly more time. The time required is also depending on the complexity of the system and the quality and structure of the written code.

This Master thesis will propose adding a preparatory step to the STPA approach; **Ensure that proper documentation of software and hardware is in order before performing the analysis**. Ideally, this should be done partially before and partially during the development of a system or product. The documentation should be developed by those who are designing and developing, not by those performing the safety analysis. In fact, making proper documentation for a system that is being developed should be common practice. Yet, it is surprisingly common to cut corners here. To be able to perform a thorough analysis, that will reveal all potential hazards, one must know the system well. Not only the hardware and software of the system itself, but also all external conditions that may affect the system during operation. The objective of this Master thesis will be considering what advantages UML documentation of the software and documentation of the system hardware may provide for STPA. It may be implicit from Leveson's theory that any system should be adequately described and documented before performing STPA, but it is not explicitly mentioned, and may make a big difference to those who are beginning to study the method.
However, it should be noted that the safety analysis should ideally be performed alongside with the planning and development of a product, to reduce the cost of necessary safety measures. Removing safety issues after a product is made can be significantly more expensive than integrating it into the system design, before the product is developed. Yet, performing a safety analysis in retrospect is still quite common, and may

also be applicable while studying accidents that have happened to find its causes.

3.4.1 Maneuvering Capacity

During a discussion with co-supervisor Glomsrud, he brought up a concept he had defined during his work with STPA and autonomous vessels. The term was found very useful when describing scenarios in the STPA analysis and has therefore been used frequently. The term *maneuvering capacity* refers to a vessel's ability to maneuver in the water. A vessel with all its thrusters and components working as expected, in perfect conditions (weather, waves and currents) will have its full maneuvering capacity available. There are a number of factors that can reduce the maneuvering capacity of a vessel. Examples are provided in the list below:

- The loss of a thruster. E.g. if the bow thruster of the ReVolt stops working, the ReVolt can still maneuver, but with a reduced ability to perform sharp turns and has less total thruster power.

- Reduced functionality of a thruster, such as not being able to apply its full thrust force or being unable to move a thruster to certain angles.

- Damage to the hull of the vessel. A deformation of the hull may increase the drag force of the vessel, which may result in the process model of the controller to incorrectly modeling the state of the vessel.

- Poor weather conditions cause in the applied control signals not to result in the expected change in position.

- Significant movements of the water masses (such as waves and currents) around the vessel, cause the control signals not to have the expected outcome.

These are just a few examples of what may reduce the maneuvering capacity of the vessel.

3.5 Existing Work on STPA with UML

3.5.1 STPA based Hazard and Risk Analysis - SAHRA

Researching the topic of UML and STPA, it was found that others have developed a UML tool, designed for visualization of the STPA analysis itself. A group of students from Zurich University of Applied Sciences have developed SAHRA; STPA based Hazard and Risk Analysis (Krauss et al.; 2015a). SAHRA is a software tool developed to perform the STPA process with visual tools. SAHRA proposes a set of new types of UML diagrams specialized for the STPA process. It provides functionality to draw the hierarchical control structure, and to visualize STPA Step 1 and STPA Step 2 separately. This tool is already implemented into Sparx Systems Enterprise Architect. An STPA analysis for a complex system can become very large and may seem unorganized. These UML diagrams may help provide some overview and structure to the "clutter" of unsafe control actions and constraints. These UML diagrams may make up a good combination with the proposed change, modeling the system using various UML diagrams before performing an STPA.

Figure 3.2 shows the SAHRA toolbox. The toolbox contains all the system components that can be used in SAHRA. There are four different categories. **STPA Fundamentals** include the components that are required in Step 0 of STPA, for identification of accidents, hazards and safety constraints. **Hierarchical Control Structure** contains the components required for such a diagram; controller, controlled process, control action and feedback. **STPA Step 1** includes components to describe unsafe control actions. **STPA Step 2** contain the variables that are used to describe the causal scenarios and factors of the unsafe control actions (as in the listed accident causes shown earlier in the chapter). (Krauss et al.; 2015b) shows a few examples of what the diagrams look like, these are shown in Figure 3.3 and 3.4.

Note that in Figure 3.4 there is a box with a small key-symbol in the corner. This is a *keyword*. The keyword is not shown in the SAHRA toolbox in Figure 3.2. The keywords are corresponding to those presented in the list at the top of page 12. (Krauss et al.; 2015b)

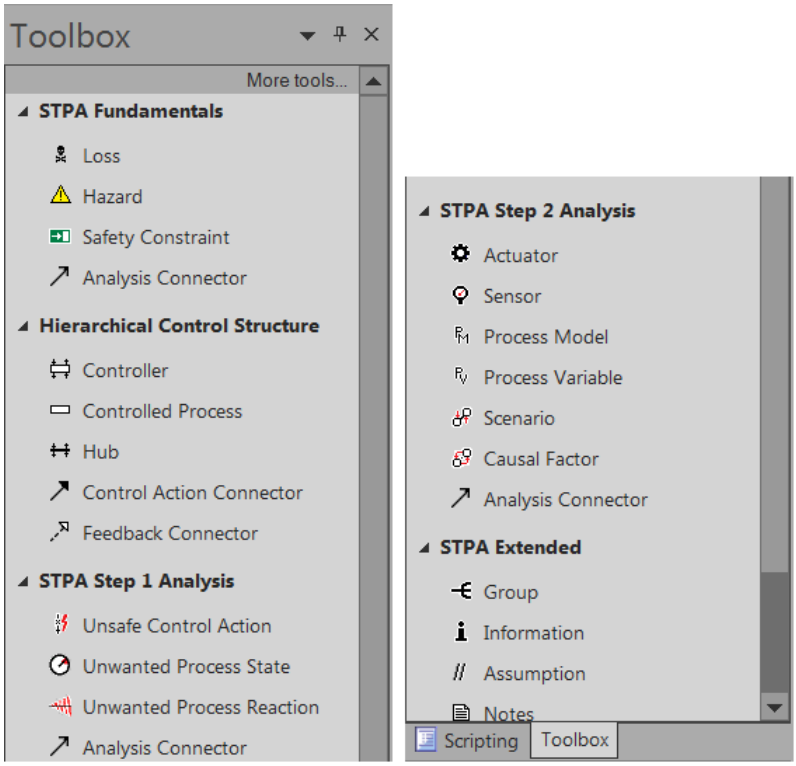

Figure 3.2: The SAHRA toolbox. Courtesy of (Safety-Critical Systems Research Lab Team of ZHAW; 2017).

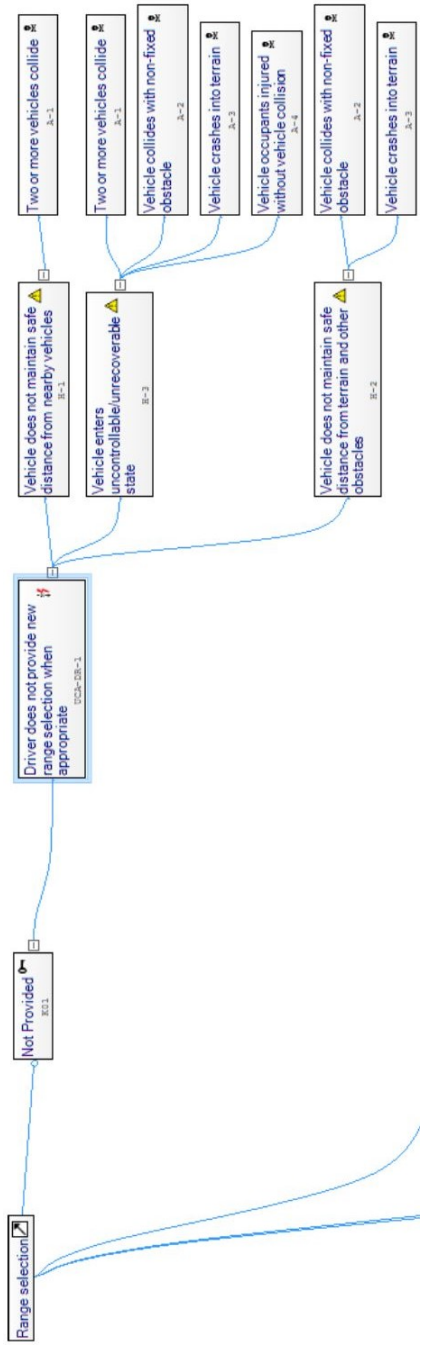

Figure 3.3: Small example showing Step 1 of STPA analysis with SAHRA. Courtesy of (Safety-Critical Systems Research Lab Team of ZHAW; 2017).

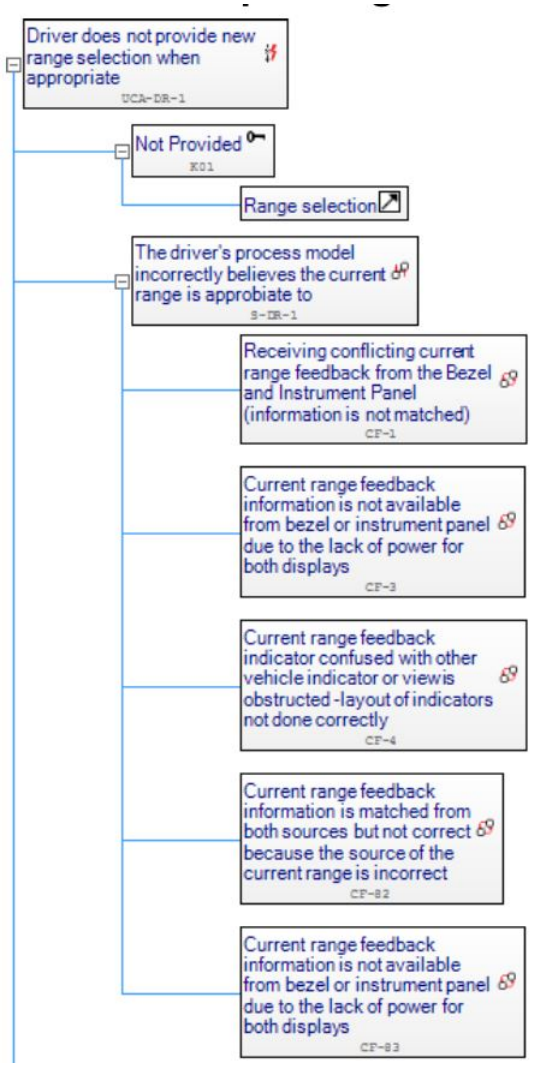

Figure 3.4: Small example showing Step 1 of STPA analysis with SAHRA. Courtesy of (Safety-Critical Systems Research Lab Team of ZHAW; 2017).

Chapter 4

Tools and Tables Used in the STPA Analysis

4.1 Improved and extended STPA analysis

The project thesis introduced an excel framework designed for performing the STPA analysis. Since the project thesis was delivered, the co-supervisor, Glomsrud, has improved this excel framework. Glomsrud's framework is more systematic and provides a better structure than the existing frameworks for STPA. It makes it easier to perform a complete STPA analysis, without accidentally omitting any details. To investigate whether the updated framework improves the analysis, the low-level STPA analysis of the ReVolt has been redone as part of this Master thesis. The improved method will be described in this chapter.
First, an example will be introduced that is often used when explaining STPA. This example will be used in this chapter when explaining the methods and principles of the STPA analysis are explained. Then, the tables that Leveson has introduced for performing an STPA analysis will be explained. Lastly, the new Excel framework will be introduced and discussed.

4.2 The Train Door Example

Please note that the information presented in this section is retrieved from (Leveson; 2011) and (Leveson and Thomas; 2015).
The example describes the controller of a train door. The door controller can perform four different control actions; open door, stop opening door, close door and stop closing door. Figure 4.1 shows the control structure of the train door.
Some possible system accidents are: a passenger falling out of the train (A1), passengers

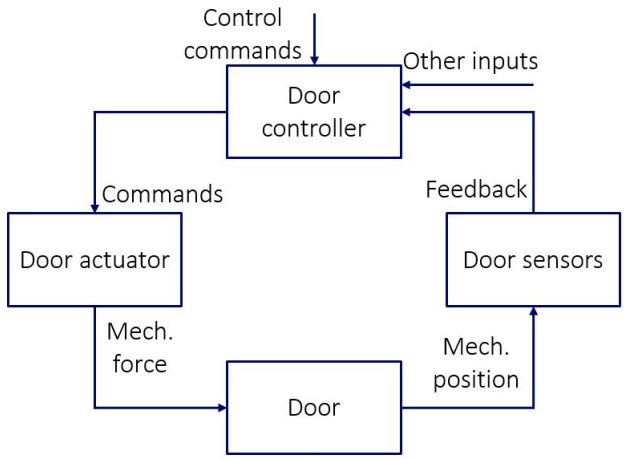

Figure 4.1: The control structure of the train door

are not let out of the train when an emergency is reported (A2), the door closes on a person resulting in an injury (A3). Some related system hazards are:

1. The door is not closed when the train starts moving.
2. The door opens while the train is in motion.
3. The door opens while the train is not aligned with a platform.
4. The door does not open when an emergency is reported.
5. The door is closed while there is someone in the doorway.

The list of hazards is not complete, but serves well as an example. From the hazards, one can observe that the door controller must receive information regarding the train motion and position, whether any emergencies are reported, in addition to the door position and state. The information regarding the door will come from the door sensors, but the other information is shown as "other inputs" in the control structure.

4.3 Leveson's STPA Tools and Framework

4.3.1 Step 0 - Defining Accidents and Hazards

The first part of the STPA analysis mainly consists of three steps:

1. List all possible system-level **accidents** and losses.
2. List the system-level **hazards** that are related to each accident.
3. List system-level **safety constraints**.

These three steps provide an overview of what can go wrong (accidents), how it may happen (hazards), and how it should be avoided (safety constraints). Hazards occur when safety constraints are violated. Note that everything is on system-level, and does not contain any technical details. The technical details will be studied further down in the steps of the analysis. When explaining these concepts, the train door controller is often used as an example. For the train, a system-level accident might be *a person falling out of a moving train.* The related system-level hazards are the situations where this accident is prone to happen; *the train door is open while the train is moving.* To avoid this hazardous situation from occurring one can apply system-level constraints, such as; *the doors must always be closed before the train can start moving, and the doors cannot open as long as the train is in motion.* There are several ways to perform and document these steps of the STPA analysis. Two alternatives are presented in (Leveson; 2011). It is common to simply list all the accidents, hazards and system constraints, and identify them as A1, A2, H1, H2, SC1, SC2 and so on. However, using the tables presented in Figure 4.2 and 4.3 makes it more orderly.

Accident	Hazard
Accident 1 (A1)	Hazard 1 (H1)
	Hazard 2 (H2)
	Hazard 3 (H3)
Accident 2 (A2)	Hazard 4 (H4)
Accident 3 (A3)	Hazard 5 (H5)
	Hazard 6 (H6)

Figure 4.2: Table used to identify system accidents and the related hazards

Hazard	Safety Design Constraint
Hazard 1 (H1)	Safety Constraint 1
	Safety Constraint 2
Hazard 3 (H2)	Safety Constraint 3
	Safety Constraint 4
Hazard 5 (H3)	Safety Constraint 5
	Safety Constraint 6

Figure 4.3: Table used to identify system-level safety constraints

Before starting the next step of the STPA-analysis one must design the safety control structure of the system. There are not many clearly stated guidelines explaining how a safety control structure should be designed. Figure 4.4 provides some guidance on the general form of the control structure. However, most systems will be significantly larger and more complex. This part of the analysis may be quite time-consuming for a first-timer. Studying existing examples is often the most effective way of understanding how to design the safety control structures.

Leveson has provided a socio-technical hierarchical control structure, which is shown in Figure 4.5. It shows a template for a typical hierarchical control structure. The box marked *operating process* contains the system itself, and the layers above and to the left show the hierarchy of controllers above it. The right side shows the system

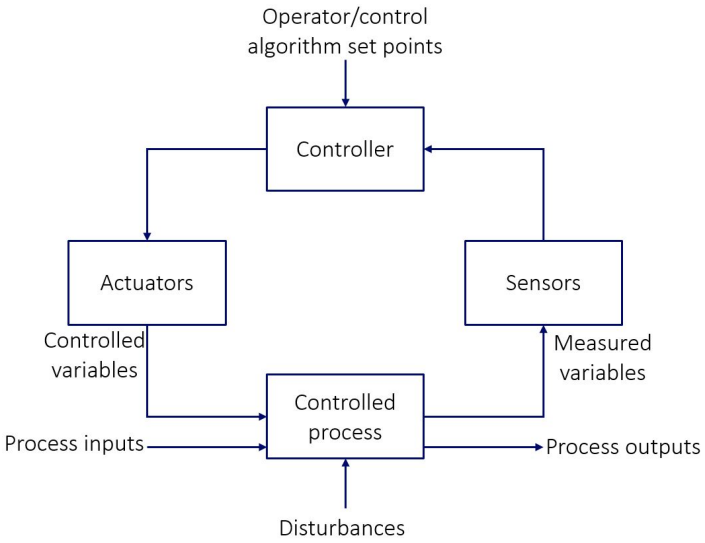

Figure 4.4: Typical control loop

development and the left side shows the system operations. Each controller imposes constraints on the layers below. The controllers can be of different types; human, automated or they can even be a set of laws or guidelines.

Definition: "Between the hierarchical levels of each safety control structure effective communication channels are needed, both a downward *reference channel* providing the information necessary to impose safety constraints on the level below and an upwards *measuring channel* to provide feedback about how effectively the constraints are satisfied" (Leveson; 2011).

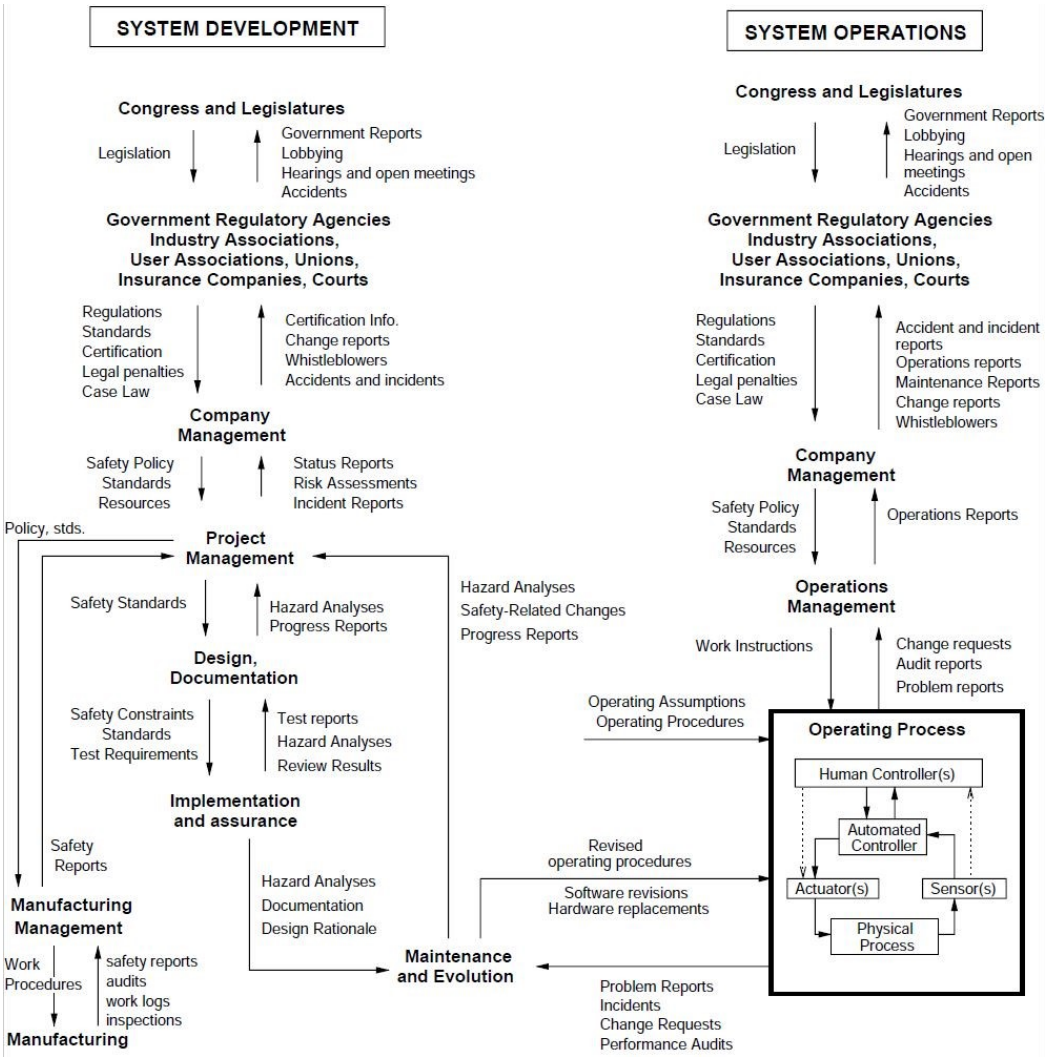

Figure 4.5: Leveson's guide to the hierarchical control structure, courtesy of (Leveson; 2011)

4.3.2 Step 1 - Defining Unsafe Control Actions

Once a detailed control structure is in place, it is time to perform step 1; identify the unsafe control actions. In her book, *Engineering a Safer World*, Leveson states that it may be beneficial to use a table to document this stage of the process. The type of table presented in the book is shown in Figure 4.6. The rightmost column identifies the control action in question. The next four columns show the four types of possibly unsafe control actions Leveson has defined:

- Providing a control action results in a hazard
- A safe control action is provided too early, too late or in the wrong order results in a hazard
- A safe control action is applied too long or stopped too soon results in a hazard
- The lack of a control action leads to a hazard

A table as this one should be created for every controller in the system. For each control action, one must list the condition or *context* that may make the control action unsafe. A control action will most likely only be hazardous under certain conditions. Such as the train door example; opening the train door will only lead to a hazard if the train is moving or if it is not aligned with the platform.

The context is sometimes referred to as a set of process model variables. They are

Control action	Not providing causes hazard	Providing causes hazard	Too early/too late, wrong order causes hazard	Stopping too soon/applying too long causes hazard
CA	(condition)	(condition)	(condition)	(condition)
CA	(condition)	(condition)	(condition)	(condition)
CA	(condition)	(condition)	(condition)	(condition)

Figure 4.6: Table for Step 1 - Identify UCAs

simply a set of variables that help describe the situation when the control action is applied. Defining which process model variables are important for each control action

can be very challenging for complex systems. It is up to the analyst to determine which factors are significant and may have an impact on the safety of the control actions, which may not always be obvious. This way, every combination is at least considered, but it is up to the one performing the analysis to determine whether a control action really is unsafe.

Figure 4.7 shows an example of a context table for the train door controller. Note

Control action	Not providing causes hazard	Providing causes hazard	Too early/too late, wrong order causes hazard	Stopping too soon/applying too long causes hazard
Open door	Door not opened when a fire is reported inside	Door is opened while the train is moving Door is opened while the train is not aligned with the platform	Door is opened too early - before train has reached a complete stop at a platform	N/A
Close door	Door not closed when train starts moving	Close door when there is a person in the doorway Close door when a fire is reported	Door is closed too late - after the train has started to move from the platform	N/A

Figure 4.7: Example of context table for a train door controller

that this table is not complete, it is just included to illustrate how the table should be used. If a control action is safe, the box can be filled with N/A or a dash, indicating that it is considered safe or not relevant. For unsafe control actions, one should fill in which context makes the control action unsafe. Note that there may be more than one context where the control action is unsafe.

A new, alternative table is presented in *An STPA Primer* (Leveson and Thomas; 2015). It is called a **context table**, and is designed by John Thomas (MIT). The context table

provides a two-step process to identify unsafe control actions. The tables are presented in Figure 4.9 and Figure 4.10. These tables are slightly different from the table from *Engineering a safer world.* There are added columns containing process model variables. These process model variables are variables that describe the context where the control action is applied. For the train door, examples of the process model variables can be the train motion, its position, reported emergency, the door state or the door position. For every control action, one must go through every possible combination of the context variables and assess whether it is safe or unsafe.

The first table examines the events where applying a control action can become hazardous. The second table examines the cases where **not** applying a control action can lead to a hazard. The tables are independent of each other and can be completed in any order, providing the same results. The results from these tables can be summarized in a table such as the one designed by Leveson (Fig. 4.6).

Finding all the process model variables for each control action is not always straightforward. To do this, one must first define the process model of the controller. In a controller's process model, there should be a list of which system states and environmental states may affect the outcome of the control action. Figure 4.8 illustrates what the train door controller's process model may look like (note that not all states are included here). For every variable, there is an option named *unknown.* This is to cover the cases where a sensor is broken, or the data from the sensor does not reach the controller.

According to (Leveson and Thomas; 2015) all the required process model variables can be found by studying the "defined hazards, the feedback in the control structure and other knowledge of the environmental and process states". There are no other guidelines to this step of the process, making it very dependent on the expertise of the analyst. Still, it is stated in (Leveson and Thomas; 2015) that this part of the analysis can easily be done by a single analyst, in contrary to the last step of the STPA process, which may require a team of people.

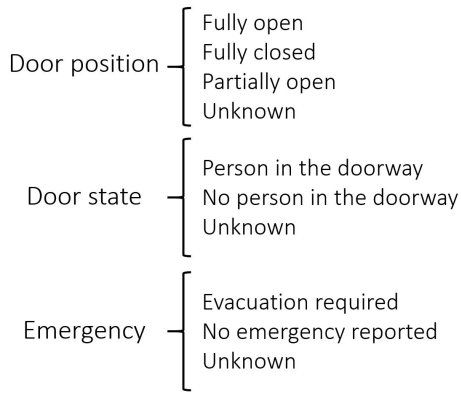

Figure 4.8: Part of the process model of the door controller.

When using Thomas' tables, it is easier to uncover potential conflicts in the sys-

Control action	Process model variable 1	Process model variable 2	...	Process model variable N	Hazardous control action in this context?		
					If provided any time	If provided too early	If provided too late
CA					Yes/No	Yes/No	Yes/No
CA					Yes/No	Yes/No	Yes/No
CA					Yes/No	Yes/No	Yes/No

Figure 4.9: Context table 1

tem. Such as for the train door controller: What should be done if the train is moving and a fire is reported? Should the passengers be trapped inside the train until it reaches a complete stop, or should the doors open allowing passengers to jump out of a moving train? Situations like these may be crucial for the design and should be carefully considered.

A table like the one shown in Figure 4.11, can be used to make a structured overview

Control action	Process model variable 1	Process model variable 2	...	Process model variable N	Hazardous control action in this context?
					If not provided
CA					Yes/No
CA					Yes/No
CA					Yes/No

Figure 4.10: Context table 2

of which accidents and hazards the unsafe control actions are linked to. In this case, each UCA that is identified in the context table should be given a number, and the number-id can be listed as in fig. 4.11.

Leveson and Thomas (2015) provides the typical structure for an unsafe control

Accident		Hazard		UCA
A1	*Accident1*	H1	*Hazard1*	3,4,12,25
		H2	*Hazard2*	1,3,5,13
A2	*Accident2*	H3	*Hazard3*	2,6
		H4	*Hazard4*	7
		H5	*Hazard5*	10,11,20
A3	*Accident3*	H6	*Hazard6*	8,9,18,19

Figure 4.11: Table used to provide an overview of which UCAs are related to which hazards and accidents - as presented in (Leveson and Thomas; 2015).

action: **Source, Type, Control Action, Context**. The *source* explains which controller provides the control action. The *type* explains if the was provided or not, and the timing of the command (too early/too late). The *control action* is the name of the command that is given. The *context* is a description of the system and/or environmental state that contributes to making the action unsafe.

4.3.3 Step 2 - Identify the Causes of and the Scenarios Leading to Unsafe Control Actions

In the second step of STPA, the scenarios for and causes of unsafe control actions are identified. Leveson states that there are two general causes of inadequate control. Either the designers have failed to identify a hazard and its accompanying safety constraints, or the control actions that are applied do not fully enforce the safety constraints. (Leveson; 2004) Leveson has provided a tool that is designed to help identify the scenarios that may lead to hazards: a detailed control loop. Figure 4.12 shows what type of failures may occur in the different parts of the control loop. These are the failures that can result in unsafe control actions. For every unsafe control action that has been identified in Step 1, one should go through the control loop and identify the possible scenarios that may allow it to happen. The results from the analysis can aid the engineers and designers in making a safe system, by indicating required changes to the existing system design. The scenarios that have been identified as causes of unsafe control actions should either be removed, controlled or mitigated to an acceptable level. (Leveson and Thomas; 2015)

Note that in the control loop in Figure 4.12 there are small circled numbers. These indicate four types of control flaws. They will be explained in this section. Note that these were already listed in Chapter three, but are repeated her for increased readability. Leveson (2011)

1 - Unsafe Inputs: The unsafe control is caused by incorrect or missing input from the controller above it in the control structure.

2 - Unsafe Control Algorithms: There are flaws in the control algorithm which result in the algorithm being unable to enforce the required safety constraints. Also, if the system or process is modified without the control algorithm being modified accordingly, control flaws may occur. Note that controllers may also be human. In that case, required training, as well as providing clearly stated procedures for the operators will make up the control algorithm.

3 - Inconsistent, Incomplete or Incorrect Process Models: The process models held by the controllers must correctly represent the actual process state. If the model

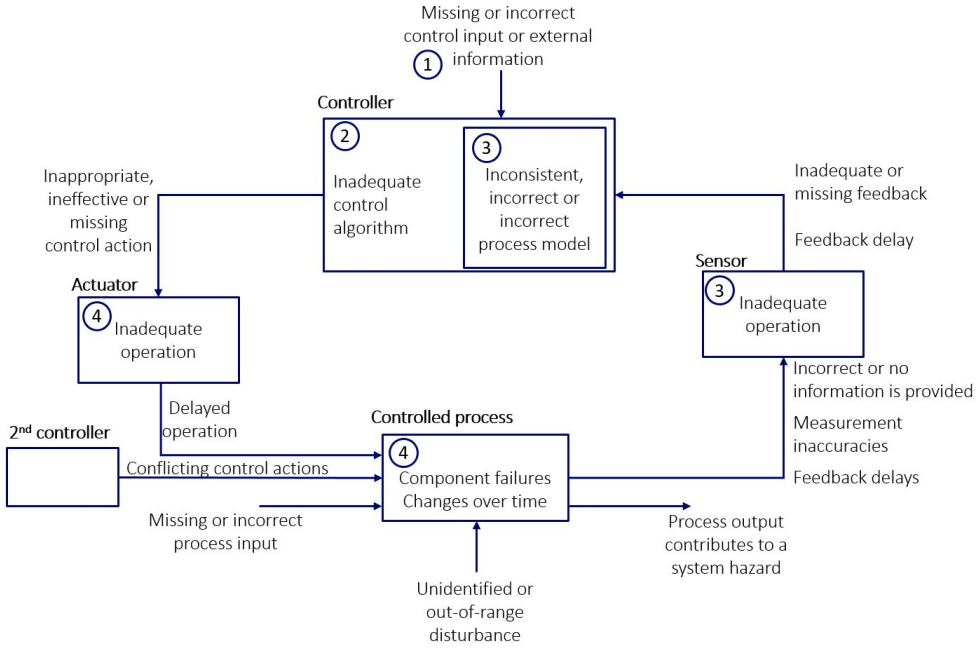

Figure 4.12: Tool that can help identify causal scenarios of unsafe control actions. As presented in (Leveson; 2011).

deviates from the actual process it may cause unsafe control actions. For a human controller, the mental model is as important. The process model may be incorrectly updated due to incorrect or missing sensor data. Resulting in an incorrect belief of the current process state.

4 - Actuators and Controlled Processes: The commands that are given are safe, but they are not put into action. There may be various causes; An actuator or another component may be broken or without power. Also, the reference channels may not be working properly. If there are multiple controllers for one process, they may provide contradictory commands.

Note that 4 includes the fourth type of unsafe control action that was defined in

Step 1: *The lack of a control action leads to a hazard.* In addition, it covers component failures, which in most other accident analysis techniques is considered the main cause of accidents. The cause of unsafe control actions being provided can typically be found in the right side of the loop (1-3). While the causes for the lack of needed control actions or inadequate execution of control actions are usually found in the left side of the control loop (4).

The final step is the most difficult and time-consuming part of the STPA-analysis. Additionally, it is the step that requires the most from those performing the analysis. It may require a lot of brainstorming, and it will most likely require a team of people working together. Prior experience with similar types of systems can be an important factor when it comes to identifying all possible causes. One of the most discussed challenges of STPA is how to know when step 2 is finished. For every scenario and safety constraint that is identified, one can define a set of sub-scenarios and sub-constraints for every scenario. *"Note that the analysis does not have to continue to find every detailed cause but can stop once enough information has been collected that an acceptable solution is identified. The analyst can stop refining causes at the point where an effective mitigation can be identified and not go down any further in detail. The analyst only has to continue refining causes if an acceptable mitigation cannot be designed."* Leveson and Thomas (2015).

4.4 Proposing a Modified Framework for STPA

The preparatory work that was done in the project thesis, (Solberg; 2017), introduced an outline for an alternative framework for performing the STPA analysis. Since then, Glomsrud has developed the excel framework even further. The new framework will be presented in the following sections. It is important to note that this is just a framework that is designed to ease the use of STPA, it will not change the STPA-analysis itself. It is intended to replace the tables that have been presented in the previous sections, designed by Leveson (and Thomas).

4.4.1 Step 0

Step 0 is the one where the least number of changes have been done. Table 4.13 illustrates the outline of step 0 of STPA in the Excel framework.

Level	Type	ID	Text	Comments ↓
			General	GROUP
0	A	01	***Accident1***	Accident
1	H	01.1	**Hazard1**	Scenario
2	C	01.1.1	*Constraint1*	Constraint or goal
3	FR	01.1.1.1	Functional requirement 1	Observer need
3	FR	01.1.1.2	Functional requirement 2	Observer need
3	FR	01.1.1.3	Functional requirement 3	Observer need
3	FR	01.1.1.4	Functional requirement 4	Controller need
3	FR	01.1.1.5	Functional requirement 5	Controller need
1	H	01.2	**Hazard2**	Scenario
2	C	01.2.1	*Constraint2*	Constraint or goal
3	FR	01.2.1.1	Functional requirement 1	Observer need
3	FR	01.2.1.2	Functional requirement 2	Observer need
3	FR	01.2.1.3	Functional requirement 3	Observer need
3	FR	01.2.1.4	Functional requirement 4	Controller need
3	FR	01.2.1.5	Functional requirement 5	Controller need

Figure 4.13: Excel table for Step 0 of STPA

At the lowest level, the system accidents are defined. Note that the leftmost column states that it is level 0, and the type is **A** - accident. Each accident is given in ID, here: 01. In the column labeled text, the accident is to be stated, such as *Passenger falling out of the train.*

One level higher, a related system-level hazard is stated. It is at level one, and is of type **H** - hazard. The ID first states which accident it is related to, and then which number of hazard it is; 01.1. In the text-column, one could write something like *Door opens while the train is moving.*

At level 2, a system-level constraint is defined, to prevent the hazard from happen-

ing. The type, **C**, indicates a constraint. The ID consists of three terms; accidentID.hazardID.constraintID - 01.1.1. Continuing with the train door example, the constraint could be *The doors must never open while the train is in motion.*
The constraint explicitly requires some functional requirements (FR), resulting in the need for observers and controllers in the system. The ID consists of four terms; accidentID.hazardID.constraintID.functionalRequirementID - 01.1.1.1. Continuing with the example, the resulting functional requirements would be: *Must know the state of the train: moving/stopped (observer), Must know the state of the door: open/closed/opening/closing (observer), Must be able to control the state of the door: open/close (controller)* In the excel table it can be stated in the rightmost column whether the FR expresses the need for an observer or controller.

The safety control structure is designed as in Leveson's description, no changes are done. All controllers and their possible control actions should be listed in a table such as the one shown in Figure 4.14. For the train door, this table would be quite simple. It only includes one controller, with 4 possible control actions: open, close, stop opening and stop closing.

Level	Type	ID	Description	Type
0	CTRL	01	***C1***	Controller
1	CA	01.1	**CA1-1**	Control Action
1	CA	01.2	**CA1-2**	Control Action
0	CTRL	02	***C2***	Controller
1	CA	02.1	**CA2-1**	Control Action
1	CA	02.2	**CA2-2**	Control Action
0	CTRL		***C3***	Controller
1	CA		**CA3-1**	Control Action
1	CA		**CA3-2**	Control Action
1	CA		**CA3-3**	Control Action

Figure 4.14: Table listing all controllers and control actions

4.4.2 Step 1

When all the information from step 0 is correctly entered into the excel tables, a table as the one shown in Figure 4.15 is generated.

Level	Type	ID	Description
H	H	(H01.1)	H01.1 Doors open while the train is in motion
0	CTRL	(H01.1)01	***Train door controller***
1	CA	(H01.1)01.1	**Open**
2	UCA	(H01.1)01.1.1	*N - Not provided when needed*
2	UCA	(H01.1)01.1.2	*U - Provided but unsafe*
2	UCA	(H01.1)01.1.3	*W - Wrong magnitude/length/timing/sequence provided*
2	UCA	(H01.1)01.1.4	*NW- Provided but not followed*
1	CA	(H01.1)01.2	**Close**
2	UCA	(H01.1)01.2.1	*N - Not provided when needed*
2	UCA	(H01.1)01.2.2	*U - Provided but unsafe*
2	UCA	(H01.1)01.2.3	*W - Wrong magnitude/length/timing/sequence provided*
2	UCA	(H01.1)01.2.4	*NW- Provided but not followed*
1	CA	(H01.1)01.3	**Stop opening**
2	UCA	(H01.1)01.3.1	*N - Not provided when needed*
2	UCA	(H01.1)01.3.2	*U - Provided but unsafe*
2	UCA	(H01.1)01.3.3	*W - Wrong magnitude/length/timing/sequence provided*
2	UCA	(H01.1)01.3.4	*NW- Provided but not followed*
1	CA	(H01.1)01.4	**Stop closing**
2	UCA	(H01.1)01.4.1	*N - Not provided when needed*
2	UCA	(H01.1)01.4.2	*U - Provided but unsafe*
2	UCA	(H01.1)01.4.3	*W - Wrong magnitude/length/timing/sequence provided*
2	UCA	(H01.1)01.4.4	*NW- Provided but not followed*
H	H	(H01.2)	H01.2 Doors open when the train is not alligned with the platform
0	CTRL	(H01.2)01	***Train door controller***
1	CA	(H01.2)01.1	**Open**
2	UCA	(H01.2)01.1.1	*N - Not provided when needed*
2	UCA	(H01.2)01.1.2	*U - Provided but unsafe*
2	UCA	(H01.2)01.1.3	*W - Wrong magnitude/length/timing/sequence provided*
2	UCA	(H01.2)01.1.4	*NW- Provided but not followed*
1	CA	(H01.2)01.2	**Close**
2	UCA	(H01.2)01.2.1	*N - Not provided when needed*
2	UCA	(H01.2)01.2.2	*U - Provided but unsafe*

Figure 4.15: Excel table for STPA-analysis

For each hazard, one has to go through every controller and its control actions. On level two, note that the four types of unsafe control actions are listed: *Not provided when needed*, *Provided, but unsafe*, *Wrong magnitude/length/timing/sequence provided* and *Provided, but not followed*. In this auto-generated table, every possible unsafe control action is listed. Now, it is up to the analyst to consider whether each one can occur, and whether it will result in the hazard in question. Note that the hazard is

listed at the top of the table.
Looking at Figure 4.15, the analyst starts with the first possibly unsafe control action: *Train door controller does **not** provide open door command (when needed) while the train is moving*. Note that this is formulated as explained at the end of section 4.3.2; Source, Type, Control Action, Context. The analyst may then conclude that the door will never need to be opened while the train is in motion, and that it is acceptable that a door open command is not provided while the train is in motion. In that case, this control action will be considered safe, and it does not have to be studied any further. This is, of course, if the analyst has decided that it is safer to keep the doors closed while the train is moving, even when an emergency is reported.
The analyst will then move on to the next line. *Train door controller (unsafely) provides a door open command while the train is moving.* This is a control action that can be considered unsafe. This is where step 2 of STPA comes in. The analyst(s) must examine how this unsafe control can be provided.

4.4.3 Step 2

The analysts must now closely study every part of the control loop in Figure 4.12. It must be determined in which ways the unsafe control action can occur. It is important to note that there may be more than one cause of an unsafe control action. These scenarios should be entered into the table that is shown in Figure 4.15. Continuing the example that was started in 4.4.2, a set of potential causes for the unsafe control action will be listed (note that accident cause is abbreviated to AC):

- AC1 - The controller above falsely indicates that the train is at a platform.
- AC2 - The control algorithm is incorrectly implemented, the door will be opened in all situations where an emergency is reported, even when the train is in motion.
- AC3 - The sensor measuring the train motion is malfunctioning, falsely indicating that the train has stopped.

- AC4 - An open door command that was provided while the train had stopped, is delayed and is put into action after the train has started moving.

For each scenario, a constraint should be created to assure that the scenario cannot occur, or that can mitigate the consequences of the control action. An example is shown in figure 4.16. Note that the analysis has reached level four. Level 0 indicates

Level	Type	ID	Description
H	H	(H01.1)	H01.1 Doors open while the train is in motion
0	CTRL	(H01.1)01	***Train door controller***
1	CA	(H01.1)01.1	**Open**
2	UCA	(H01.1)01.1.1	*N - Not provided when needed*
3	S	(H01.1)01.1.1.1	AC1 - 'The controller above falsely indicates that the train is at a platform.
4	C	(H01.1)01.1.1.1.1	The speed sensors must indicate that the train has stopped for the doors to open
3	S	(H01.1)01.1.1.2	AC2 - The control algorithm is incorrectly implemented, the door will be opened in all situations where an emergency is reported, even when the train is in motion.
4	C	(H01.1)01.1.1.2.1	The algorithm can only allow the door to open while the train has stopped
3	S	(H01.1)01.1.1.3	AC3 - The sensor measuring the train motion is malfunctioning, falsely indicating that the train has stopped.
4	C	(H01.1)01.1.1.3.1	Add more sensors, compare values Add functionality that can check for errors, frozen sensor values or detect false spikes
3	S	(H01.1)01.1.1.4	AC4 - An open door command that was provided while the train had stopped, is delayed, and is put into action after the train has started moving.
4	C	(H01.1)01.1.1.4.1	All messages from the door controller should be received and acknowledged before the train can start moving
2	UCA	(H01.1)01.1.2	*U - Provided but unsafe*
2	UCA	(H01.1)01.1.3	*W - Wrong magnitude/length/timing/sequence provided*
2	UCA	(H01.1)01.1.4	*NW- Provided but not followed*

Figure 4.16: Excel table showing UCA scenarios and constraints

the controller, level 1 indicates the control action, and level 2 indicates the (potentially) unsafe control action. At level 3, the scenarios for the unsafe control action are specified, and at level 4 the corresponding system-constraints are specified. These scenarios are quite general, and not so system specific. To get to the technical details, the analysis must be taken to higher levels. This is done by finding sub-scenarios and sub-constraints, that go further into the details of the already defined scenarios. The sub-scenarios will typically be a scenario where the defined constraint does not hold. The sub-scenarios can also be found using the control loop from Fig. 4.12. Then, sub-constraints are designed to handle the sub-scenarios. Figure 4.17 illustrates how

the sub-scenarios and sub-constraints are shown in the excel framework.

1	CA	(H01.1)01.2	**Close**
2	UCA	(H01.1)01.2.1	*N - Not provided when needed*
3	S	(H01.1)01.2.1.1	ACx - scenario
4	C	(H01.1)01.2.1.1.1	Constraint
5	S	(H01.1)01.2.1.1.1.1	ACx - Sub-scenario 1
6	C	(H01.1)01.2.1.1.1.1.1	Sub-constraint 1
7	S	(H01.1)01.2.1.1.1.1.1.1	ACx - Sub-scenario 1.1
8	C	(H01.1)01.2.1.1.1.1.1.1.1	Sub-constraint 1.1
5	S	(H01.1)01.2.1.1.1.2	ACx - Sub-scenario 2
6	C	(H01.1)01.2.1.1.1.2.1	Sub-constraint 2
2	UCA	(H01.1)01.2.2	*U - Provided but unsafe*

Figure 4.17: Illustrating how sub-scenarios and sub-constraints are shown in the Excel-table

New sub-scenarios can be defined under the sub-scenarios, taking the analysis to even higher levels. This is shown in Figure 4.17 with the sub-scenario with level 7, and the associated sub-constraint with level 8. The analysis can be taken as far as the analyst sees it necessary, by continuing to examine sub-scenarios under every scenario and sub-scenario. At the end of section 4.3, it was briefly discussed when to stop the analysis. The analyst must determine when the constraint is specific enough to prevent the unsafe control action from happening. Some sub-constraints can also be functioning as a safety net, that prevents an unsafe control action from being provided even if a lower level constraint is not applied correctly.

Chapter 5

The ReVolt

The following chapter is retrieved from the project thesis that was written in the fall of 2017, (Solberg; 2017). Some parts of the chapter are slightly rephrased, but the majority is used as it was written in the project assignment. It is confirmed with the supervisor that this is reasonable to reuse the theory that is unchanged since the project thesis was delivered.

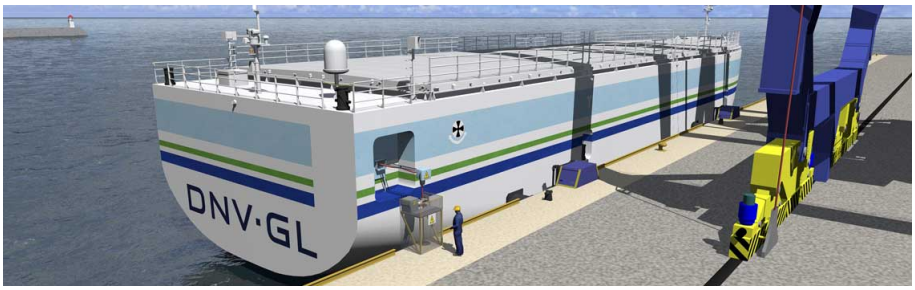

Figure 5.1: Illustration of the concept ship, ReVolt. Courtesy of DNV GL, (Tvete; n.d.).

The Revolt is a concept ship developed by DNV GL, shown in Figure 5.1. It is intended to become an unmanned ship, used for transportation of large containers. The ReVolt

will be a battery-driven ship that is designed for short-sea shipping. Its purpose is to relieve the increasingly congested road network in Europe. The increased focus on autonomous vessels has been an important motivational factor for DNV GL's ReVolt. DNV GL has not started developing the ship yet, but they have initiated a student project in cooperation with NTNU. A smaller model version of the ReVolt, 2.99 meters long, has been built to enable testing of various strategies for autonomous operation. This model ship is shown in Figure 5.2 and will be discussed further in the following sections.

5.1 The ReVolt Model

The fall of 2016, two cybernetics students started working with the ReVolt as a part of their project thesis. DNV GL provided them with the hull, and the thruster design. The two students chose and installed the remaining hardware, and began developing software. The work they did with the ReVolt model will be discussed in the following sections. Note that for simplicity, the ReVolt model will be referred to as the ReVolt from this point on.

Figure 5.2: The Revolt Model. Courtesy of Alfheim and Muggerud, (Alfheim and Muggerud; 2017a).

5.1.1 Operational Modes

During the spring of 2017, two main operational modes were developed for the Revolt: Remote control and dynamic positioning. There is also an emergency stop mode, in case an error occurs and the ReVolt needs to be stopped. By the summer of 2018, a mode offering autonomous navigation will hopefully have been implemented.

5.1.1.1 Remote controlled operation

Figure 5.3: RC Remote Control. Courtesy of Alfheim and Muggerud, (Alfheim and Muggerud; 2017b).

Using the remote control from Figure 5.3, the ReVolt can be controlled from shore. By shifting the levers on the remote, the operator can control the throttle and rudder of the thrusters, as well as lowering and retracting the bow actuator. The maximum distance between the remote and the ReVolt is limited by the coverage of the radio signals. If the ReVolt detects that communication with the remote is broken, it will shut down the thrusters.

There are three different modes of manual operation:

- Fully manual: the operator is in direct control of all thrusters
- Heading control: the operator can set the reference heading using the remote, and the heading controller software works to keep this heading.
- Manual thrust allocation: the operator sets a desired reference heading and throttle using the remote. The operator input goes through the thruster allocation

controller, instead of being applied directly to the thrusters. (The complete control structure will be explained later in Figure 5.6).

5.1.1.2 Dynamic Positioning - DP

In DP-mode, the ReVolt is controlled by a computer located on-shore. Using ROS, the computer communicates with the ReVolt over wifi. The computer sends the desired position and heading to the ReVolt. The algorithm used to keep the vessel in the desired position and heading was developed by Alfheim and Muggerud, as a part of their Maser's thesis (Alfheim and Muggerud; 2017a). The desired position cannot be located far away from the current location of the vessel; only within a few meters. There is a switch on the RC remote control, referred to as the *override switch*. If this is switched while the ReVolt is in DP-mode, it will immediately change to remote-controlled operation.

5.1.1.3 Emergency stop mode

There is a large, red emergency stop button located on top of the ReVolt. In the case where this is pushed, all thrusters are set to neutral values. The state is immediately switched to remote controlled.
This is the only safety aspect that has been added to the ReVolt at this point. An operator has to physically push the stop button located on the ReVolt for it to be activated.

5.1.1.4 Autonomous operation

If the autonomous development goes as intended, the vessel will be able to receive a destination as input, and move to the desired position autonomously. This will be accomplished by autonomously creating a safe path for the vessel to follow. As it is navigating along the planned path, it will be using cameras and a LiDAR to monitor its surroundings. These sensor measurements will be used for collision avoidance and to update the path plan. ReVolt will communicate with the on-shore operator using 4G,

and continuously feed the operator with its state information. This will be discussed further in Chapter 6.

5.1.2 The Hardware

Fig. 5.4 shows the hardware mounted inside of the ReVolt. Table 5.1 identifies each of the components. In addition to the listed components, there are a few components mounted on the outside of the ReVolt. These can be seen in fig. 5.2. There are two antennas for the GNSS receiver, one in the front and one in the back. A light beacon is used to indicate the state of the ReVolt is mounted in the back of the vessel. There are two wifi-antennas, because with the current approach, the computer on-shore uses wifi to communicate with the ReVolt. There is a red stop-button installed on top of the rear end of the vessel. Lastly, there is a radio link antenna, used for radio communication with the remote. Inside the ReVolt, there also is a water sensor intended to detect potential water leakages, designed and built by Alfheim and Muggerud.

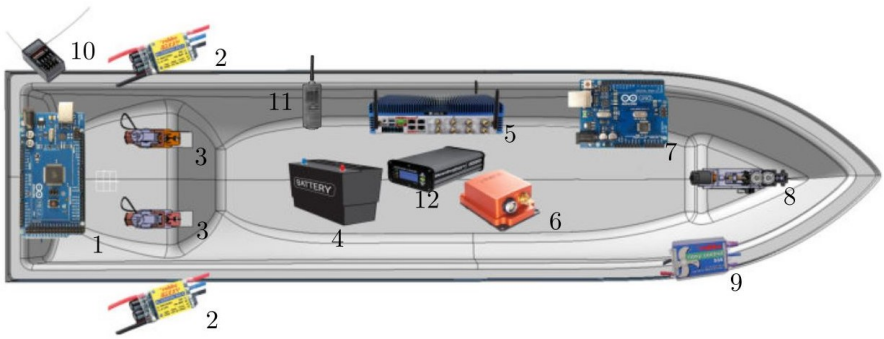

Figure 5.4: Hardware overview. Courtesy of Alfheim and Muggerud, (Alfheim and Muggerud; 2017a).

5.1.2.1 Embedded computer

The embedded computer onboard the ReVolt runs Linux Ubuntu with the ROS framework on top (ROS is briefly described in a section later in this chapter). This is where

Number	Component
1	Arduino Mega
2	Electronic Speed Controller(ESC) - AC
3	Stern Thruster
4	Battery
5	Embedded Computer
6	Xsens (Inertial Measurement Unit (IMU))
7	Arduino Uno
8	Bow Thruster
9	Electronic Speed Controller(ESC) - DC
10	RC Receiver
11	Satel Radio Link
12	GNSS Receiver

Table 5.1: Identifying the components in Figure 5.4 (Alfheim and Muggerud; 2017a)

the majority of the information processing is done. The computer is a Tank-720. The "brain" of the ReVolt runs on this embedded computer. In addition to performing the heavy calculations, it provides the Arduinos with control signals.

5.1.2.2 Arduinos

There are two Arduinos on-board the ReVolt. Arduinos are small micro-controllers which are quite intuitive to use for beginners to embedded programming. Arduino has an open-source programming language based on C++, which offers great online support. The Arduinos are used to interface with the hardware onboard. An Arduino Uno is located in the front of the vessel. It mainly applies control signals to the bow thruster. The Arduino Mega is located in the rear of the vessel. This Arduino applies control signals to the stern thrusters, in addition to operating the radio receiver, the water sensor and the stop button. The Arduino Uno and Arduino Mega offer the same functionality, but the Mega has 40 digital I/O pins and 10 analog I/O pins more than the Uno. Also, the Mega has 224kB more flash memory and 6 kB more SRAM (Static

Random-Access memory) than the Uno. The Arduino Mega is slightly larger in size than the Uno; about 3.3 cm longer. Technical details are retrieved from (*Arduino Uno*; n.d.) and (*Arduino Mega 2560*; n.d.).

5.1.2.3 Bow Thruster

The bow thruster consists of a DC-motor, an electronic speed controller (ESC), a servo motor and a linear actuator. The linear actuator can retract the thruster into the hull of the vessel. This will reduce the drag of the vessel in the water, but it makes precise maneuvering harder. The servo motor controls the direction of the thruster, which ranges from -45 degrees to 45 degrees. The thruster is capable of a rotation of ±270 degrees, but the implemented code limits the movement from -45 to +45 degrees. The electronic speed controller regulates the speed of the DC-motor, which controls the thrust by applying power to the propellers.

5.1.2.4 Stern Thrusters

There are two identical thrusters in the stern of the vessel, one referred to as the starboard thruster and the other as the port thruster, indicating the side they are installed at. Each one consists of a stepper motor, an AC-motor and an ESC. The stepper motors are used for rotation of the thrusters, controlling the direction of thrust. They can rotate 360 degrees, but are, for now, limited between ±50 degrees. Here, the ESC controls the speed of the AC-motor, which controls the effort of the propellers. The propellers can be rotated both ways, enabling the vessel to reverse.

5.1.2.5 The Xsens and the GNSS receiver

The Xsens MTi-G-710 is an advanced sensor, providing measurements for position and orientation. The Xsens functions as an inertial measurement unit (IMU), a global navigation satellite system (GNSS) and inertial navigation system (INS) (Xsens; n.d.). In addition to position and orientation, it provides estimates for angular velocity as well as acceleration. The accuracy of the position estimate is ±2 meters, and the accuracy of the orientation estimate is ±0.3 degrees, ±0.3 degrees and ±1.0 degrees for the roll,

pitch and yaw. (Alfheim and Muggerud; 2017a)

The Hemisphere vector is a GNSS receiver equipped with two antennas. The Hemisphere Vector VS330 together with the Satel radio link and RTK is capable of providing significantly more accurate position estimates than the Xsens. The position is estimated with an accuracy of ±2 **cm**. The accuracy of the orientation estimate is ±1.0 degrees, ±1.0 degrees and ±0.2 degrees for the roll, pitch and yaw. Hemisphere (n.d.) Note that for the accuracy of the yaw measurements to be ±0.2 degrees, the antennas must be placed 0.5 meters apart. The accuracy can be increased by increasing the distance between the antennas.

5.1.2.6 RC Receiver

The RC receiver is used to receive the signals from the RC remote shown in Fig. 5.3, containing maneuvering instructions. The receiver is connected to the Arduino Mega, in the rear end of the vessel.

5.1.2.7 Additional sensors

There is a water sensor installed to make sure any potential leakages are detected. The sensor is designed and built by (Alfheim and Muggerud; 2017a). There is a light beacon installed which is used to indicate the state of the ReVolt, e.g. indicating whether a water leakage is detected. Both of these components are connected to the Arduino Mega, in addition to the RC receiver and the stern thrusters. This may not be optimal. The Arduino does not have as much processing power as the embedded computer. Having this many separate tasks on a single microcontroller may create problems. If something is wrong with e.g. the light beacon, the software may get stuck in some unforeseen software loop. Then, both the RC-communication and the control of the stern motors could be lost. Ideally, these components should be implemented on a separate Arduino. This is, most likely, not done due to a limited budget.

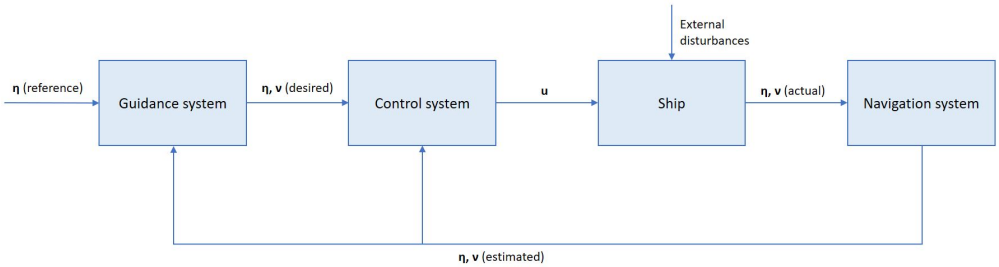

Figure 5.5: Modularized block diagram illustrating the software of the ReVolt in DP-mode, as presented by (Alfheim and Muggerud; 2017a)

5.1.3 The Software

Understanding what happens in detail in the software of the ReVolt is essential to be able to perform a safety analysis for the vessel. In their Master thesis, Alfheim and Muggerud divided the ReVolt into several modules; a guidance system, a control system and a navigation system. These modules and the communication between them is shown in Figure 5.5. Each of these modules has separate responsibilities and contain a set of software blocks. Note that η is the position and orientation vector;

$$\begin{bmatrix} x \\ y \\ \psi \end{bmatrix} = \begin{bmatrix} \text{Body-fixed position in the x-direction} \\ \text{Body-fixed position in the y-direction} \\ \text{Euler angle about the z-axis} \end{bmatrix} \tag{5.1}$$

v is the linear and angular velocity vector;

$$\begin{bmatrix} u \\ v \\ r \end{bmatrix} = \begin{bmatrix} \text{Body-fixed linear velocity in the x-direction} \\ \text{Body-fixed linear velocity in the y-direction} \\ \text{Body-fixed angular velocity about the z-axis} \end{bmatrix} \tag{5.2}$$

u indicates the total force applied by the thrusters.

- The *guidance system* receives input signals from the operator, as well as the estimated position, orientation and velocity of the vessel from the sensors. Based on the input, it calculates how to reach the next desired position and velocity, as a path or a trajectory. In DP-mode, the guidance system consists of a single software block; the reference filter. The reference filter provides a trajectory to the desired position and orientation, as long as its relatively close to the current position. As the ReVolt becomes increasingly autonomous, there will be more software components within the guidance system. The reference filter is not used while the ReVolt is manually operated.
- The *control system* receives the trajectory that takes it from the current position and velocity, to the desired position and velocity. The control system is divided into two software blocks; DP-controller and thruster allocation. The DP-controller calculates the force needed to move the vessel to the desired position and velocity, as well as the direction of the force. The thruster allocation takes in the force and the direction of the force from the DP-controller, and calculates the control signals that need to be applied to each of the three thrusters to achieve the desired force.
- As the control signals are applied to the thrusters, the position of the ship moves accordingly. External forces, that are not controlled by the vessel or the operator, will also change the position of the vessel. Among these are wind, waves and currents. The system sensors help overcome this challenge.
- The *navigation system* receives measurement data from the Xsens and Hemisphere vector sensors. It uses this data to estimate the current position, orientation and velocity. This software block is referred to as the observer from now on.

In the master thesis of Alfheim and Muggerud (Alfheim and Muggerud; 2017a), there is no complete overview of all software- and hardware-components together, showing

the communication between them. To be able to perform a safety analysis, as the one that will be presented later in this report, it is vital to have a correct, complete control structure of the system. Therefore, a significant amount of time has been spent making a complete control structure, illustrating the flow of information and control signals in the ReVolt system. The result is shown in Figure 5.6. Explanations to the figure are given in table 5.2 and the paragraphs below.

The *state controller* is the most complex software component of the ReVolt. Its functionality is a little different for each of the states. It listens for instructions from the on-shore computer and the override switch on the RC-remote, and changes states based on these instructions. The ReVolt will only change state by a command being given.

- While in *DP-mode*, the state controller provides the reference filter with DP parameter updates. The DP-parameters being the reference position and orientation. Once the on-shore computer changes the parameters, these are sent to the reference filter. Looking at Figure 5.6, only the green arrow is used while in DP-mode.

- While in *manual mode*, the input from the RC-remote is applied directly to the thruster controller (bow), thruster controller (stern) and the stepper controller. Only the purple arrows in Figure 5.6 are used in this mode. Note that the observer, reference filter and thruster allocation are not used while being in this state. The purple arrows will only be used for the manual control.

- In *manual thrust allocation mode* the user specifies a force relative to the ship. This force and direction of force are sent to the thruster allocation node, which provides the correct control input for each thruster. In Figure 5.6, this corresponds to the red arrow

- In *heading control mode*, the RC-remote is used to set a reference heading. This reference value will go into the heading controller, which provides a control effort reference for the thruster allocation. In Figure 5.6, this corresponds to the

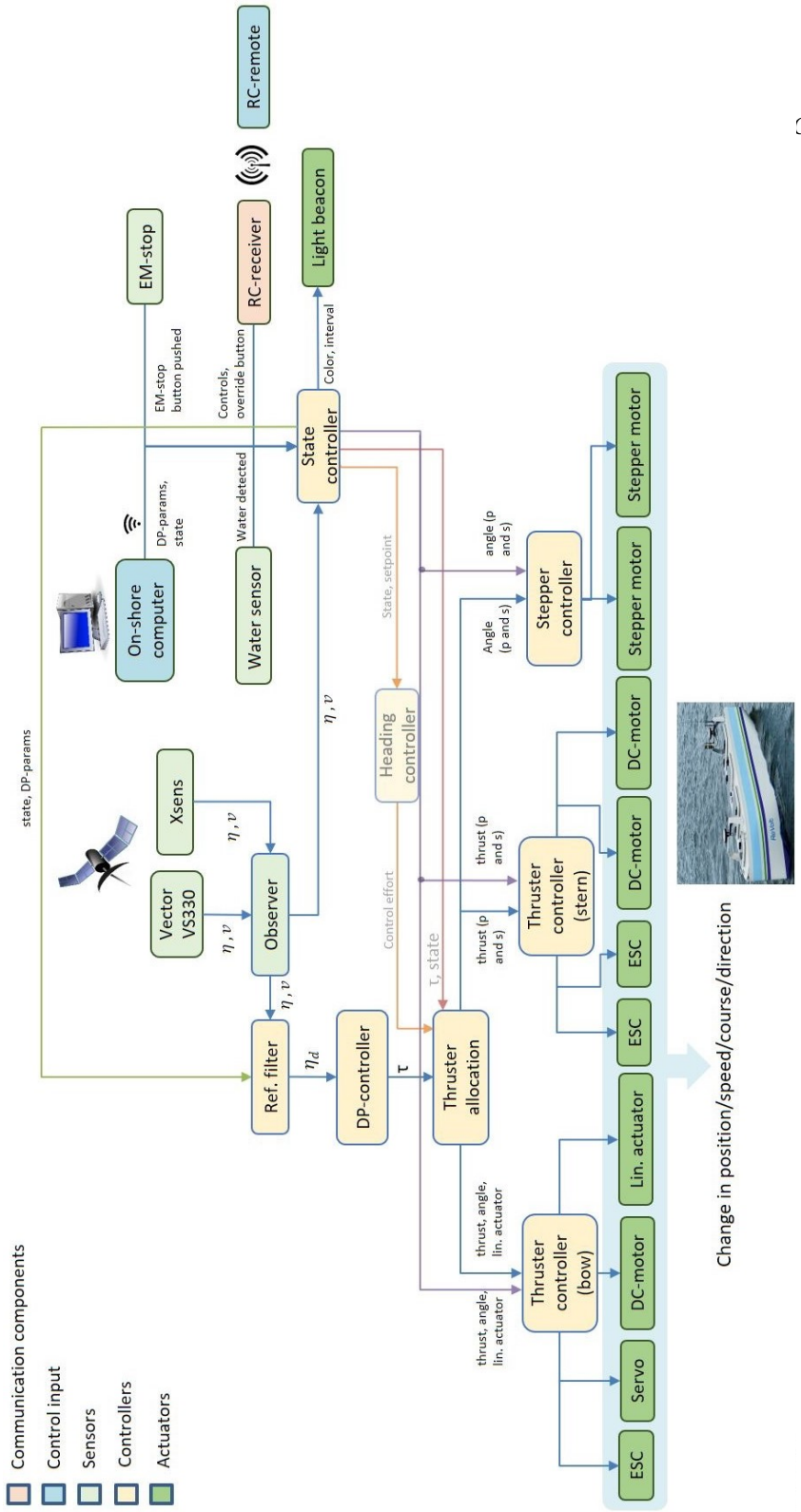

Figure 5.6: Control Structure of the ReVolt

orange arrows, in addition to the heading controller block, which is only used in this mode.

Note that the messages from the thruster controllers and stepper controller are not specified in the figure. It is omitted because of limited space, so an explanation is provided in this paragraph. The thruster controllers use low-level communication signals; pulse width modulated signals, to provide the hardware with command signals it is capable of understanding. E.g. instead of providing the electronic speed controllers with a signal between -100 and 100 percent, a number between 1100 and 2100 μs is given. 1500 μs corresponds to the neutral position of the ESC, where the propeller does not move. The percentage value is simple and intuitive to work with for the human operators, but not for the hardware components. The thruster controllers simply performs the translating work.

5.1.3.1 Robot Operating Systems (ROS)

ROS is an open-source framework developed for writing software for robots. Both Python and C++ can be used. ROS provides a standard for communication, which is used by the ReVolt. Software components are divided into *nodes*, the *nodes* communicate through *topics*. A *node* publishes messages to a specified topic, and other nodes can choose to subscribe (listen) to one or more of these topics. The publisher is unaware of the subscribers, it publishes whether or not a node is subscribing. This makes the communication easy to scale, and it can be utilized to enhance safety. A simple illustration is given in Figure 5.7 The ROS communication used in ReVolt uses both wifi, USB and RS232. The auto-generated ROS graph is shown in the appendix, Figure C.1. The oval shape indicates a node, the small rectangles are topics. The arrows indicate which nodes publish and subscribe to topics. The auto-generated graph is quite disorganized and can seem overwhelming for someone who does not know the system. It can serve as a supplement to the control structure, to get an overview of the type of information that is sent between the system components.

Figure 5.7: Illustration of ROS communication between two nodes

From	To	Message	Explanation
Vector VS330	Observer	η, ν	Explained in equations 5.1 and 5.2
Xsens	Observer	η, ν	Explained in equations 5.1 and 5.2
Observer	Ref. filter State controller	η, ν	Explained in equations 5.1 and 5.2
Ref. filter	DP-controller	η_d	Desired position and orientation
DP-controller	Thruster allocation	τ	Forces and moments vector; $\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_N \end{bmatrix} = \begin{bmatrix} \text{body-fix. force in x-dir} \\ \text{body-fix. force in y-dir} \\ \text{body-fix. moment about y-axis} \end{bmatrix}$
Thruster allocation	Thruster controller bow	thrust, angle, lin. actuator	effort for the bow thruster given as ±100% angle for the bow thruster given as ±45° position of the linear actuator (up/down)
Thruster allocation	Thruster controller stern	thrust (p and s)	effort for the stern thrusters given as ±100% s = stern, p = port
Thruster allocation	Stepper controller	angle (s and p)	angle for the stern thrusters given as ±50° s = stern, p = port
On-shore computer	State controller	DP-params, state	DP-params set the reference position and orientation for the ReVolt in DP-mode State indicates whether the ReVolt is being remote controlled or using DP
Water sensor	State Controller	Water detected	Boolean, alerts if water is detected inside the hull
RC-receiver	State Controller	Controls, override button	When being remote controlled, the thrust effort and direction is sent The override button signals whether the switch has been moved or not
EM-stop	State Controller	EM-stop button pushed	Boolean, alerts if stop-button is pushed
State controller	Light beacon	Color, interval	Indicates which color and at which frequency the light beacon should light
State controller	x	x	Complex functionality - explained below

Table 5.2: Identifying the messages in Figure 5.6

Chapter 6

System Description using UML and RBD

6.1 Description of the Modeled System

A set of assumptions have been made, in order to make a more interesting and useful case for the STPA analysis. The ReVolt model that was introduced in Chapter 5 illustrates the ReVolt as it was before the fall of 2017. As explained previously, further development has continued during this school-year. In order to be able to work continuously throughout the whole semester, it was decided to work with a static model of the ReVolt. Static meaning that the design of the model does not change throughout the semester. Following the development of the ReVolt done by other students during the semester would require much time, and performing the STPA-analysis alongside the development would be complicated. It was decided to design a ReVolt model that illustrates the expected state of the ReVolt by the end of this semester. An STPA analysis of this version may provide more help to those continuing to develop the ReVolt in the future. An analysis of the remote-controlled or the DP-state of the ReVolt, may not provide much help for the engineers in the future development. The new

ReVolt model is based on the previous design (introduced in Chapter 5), and the added functionality is built on top of the old framework. Assuming the future ReVolt will build on the original design from the fall of 2017.

The new control structure is presented in Figure 6.1. It looks a lot like the old control structure presented in Chapter 5. A few sensors have been added, and some of the old controllers have been replaced with new, more complex ones. The functionality of the controllers are explained in the next paragraphs, and the process models can be found in Appendix B. Table 6.1 provides descriptions of the communication signals that are illustrated in 6.1.
Note that the stippled arrows indicate *measuring channels* for feedback and the solid arrows indicate *reference channels*, as is defined in Chapter 4.

The following list briefly explains the functionality of the controllers in Figure 6.1:

- The *thruster allocation*, *bow controller*, *stern controller* and *stepper controller* will be kept as they were described in Chapter 5. The thruster allocation calculates the needed contribution from each controller, in terms of thruster effort and direction, based on the input from the above controller. The thruster controllers pulse width modulate the control signals and apply them to the correct output ports.

- The DP-controller will be replaced by a *force controller*. The functionality of the force controller will be similar to the DP-controller, but it will also be able to calculate necessary control signals between two points that are further distanced (where the DP could only handle small distances). The DP-functionality should also be integrated into this controller, if it is desirable to keep the DP-functionality. The DP-controller provides the thruster allocation with a vector containing the required force in the x-direction, the force in the y-direction and the moment about the z-axis.

- The *navigation controller* will be the most complex part of software. It is where

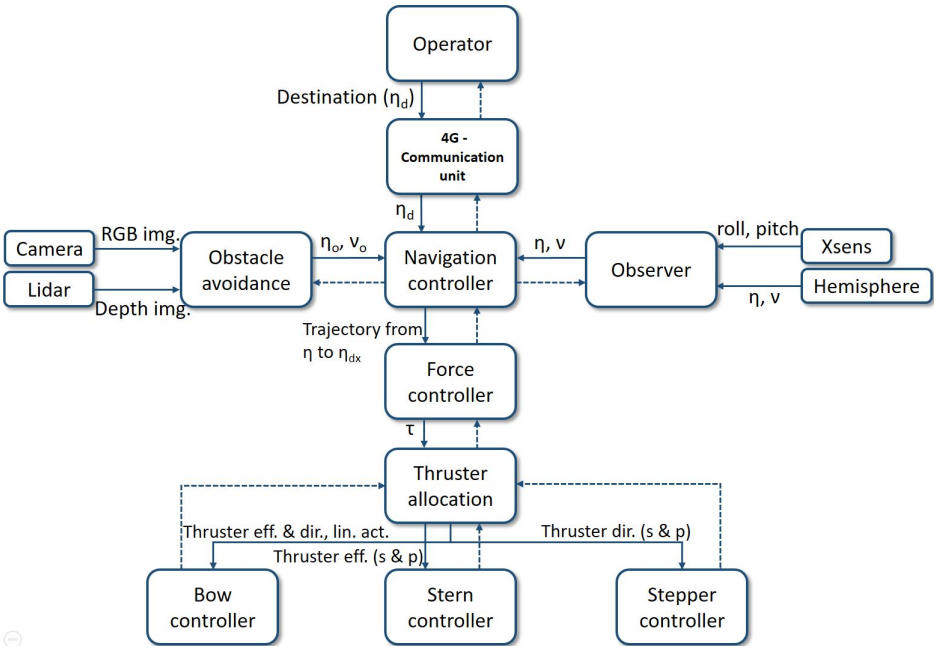

Figure 6.1: Presumed control structure of the future state of ReVolt

the majority of the functionality for autonomous navigation is implemented. The navigation controller will receive inputs from the operator, the obstacle avoidance and the observer, with information regarding the destination of the vessel, the position of any detected obstacles, and the ReVolt's current position, speed and heading. It is the navigation controller's task to provide a safe path, as a set of points, to follow in order to reach the destination provided by the operator, without causing a collision on the way. It provides the force controller with the current pose and the next desired pose (note that **pose** is a notation for position and orientation).

- *The obstacle avoidance* will receive data from the vision sensors containing information about the ReVolt's surroundings. Object detection algorithms are used to detect obstacles in the RGB images from the camera(s). The Lidar

From	To	Message	Explanation
Operator	4G-comm unit	η_d	The desired destination and heading provided by the operator in 4G message format
4G-comm unit	Navigation controller	η_d	The desired destination coordinate and heading
Camera	Obstacle avoidance	RGB img.	An array containing the RGB data in the images captured by the camera(s)
Lidar	Obstacle avoidance	Depth img.	An array containing the depth (distance) data in the images captured by the Lidar
Obstacle avoidance	Navigation controller	η_o, v_o	The estimated position, heading and velocity of a detected obstacle
Xsens	Observer	roll pitch	The measured roll and pitch angles
Hemisphere	Observer	η, v	Measured position, orientation (yaw), velocity and angular velocity
Observer	Navigation controller	η, v	Measured position, orientation (yaw), velocity and angular velocity
Navigation controller	Force controller	Trajectory from η to η_{d_x}	The trajectory from the current pose to the next point along the path
Force controller	Thruster allocation	τ	Forces and moments vector; $\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_N \end{bmatrix} = \begin{bmatrix} \text{body-fix. force in x-dir} \\ \text{body-fix. force in y-dir} \\ \text{body-fix. moment about y-axis} \end{bmatrix}$
Thruster allocation	Thruster controller bow	effort, direction, lin. act.	Thruster effort between ±100% Thruster direction between ±45° Position of the linear actuator (up/down)
Thruster allocation	Thruster controller stern	Effort (p and s)	Effort for the stern thrusters between ±100% s = stern, p = port
Thruster allocation	Stepper controller	Direction (s and p)	Direction for the stern thrusters between ±50° s = stern, p = port

Table 6.1: Identifying the messages in Figure 6.1

provides depth image of the surroundings. Fusing the information from the camera(s) and the Lidar, the position of an obstacle can be estimated. The speed

and heading can also be estimated by tracking the obstacles over time. This information is shared with the navigation controller.

- *The observer* uses data from the Hemisphere's and the Xsens's GNSS, IMU and INS to estimate the position, linear and angular velocity. This information is shared with the navigation controller.

- *The communication unit*: All communication with the on-shore operator is done over the 4G-network. The ReVolt will be able to send and receive data using 4G communication. The communication unit's function is to send and receive these messages in real-time. All received messages from the operator are distributed on to the navigation controller. The operator will mainly provide the ReVolt with its destination coordinates. The navigation controller uses the communication unit to send feedback information to the operator.

- The *operator*'s main role is to provide the ReVolt with destination coordinates. The remaining functionality is not determined yet. It may get functionality to block off areas that the vessel is restricted from operating in, due to e.g. poor weather conditions or an accident in the area. The operator should be able to request information from the ReVolt, and should be able to monitor the data captured by the cameras. The operator will have the ability to perform an emergency stop, if there is reason to believe something is wrong with the vessel. It should be discussed whether it is necessary for the operator to be able to remote control the vessel.

Note that there will most likely be two cameras onboard, to enable a broader angle of vision (even though there is just one camera in 6.1). It is planned to use Muvi K2 Sport cameras and a Velodyne VLP-16 Lidar.

6.2 Software and System Behavior - UML

A set of UML diagrams have been designed to describe the functionality of the ReVolt, they will be presented in the next few pages. It is assumed that those reading this

report are familiar with the UML language, and are able to read and understand the UML diagrams. Therefore, there will not be given detailed explanations to reader this report. UML Distilled (Fowler; 2004) is recommended if there are questions regarding standards and rules of the UML language.

6.2.1 Use-Case Diagram

Use-case diagrams illustrate the interaction between users and the system itself, showing how the system is used (Fowler; 2004). It offers a black-box view of the services that are provided by the system (Delligatti; 2014). Figure 6.2 shows the actions that the operator can perform, in order to control and supervise the ReVolt. The ReVolt is intended to be able to run independently of the operator the majority of the time. The most important task of the operator is to set a destination or a DP-coordinate. Without these inputs, the ReVolt will not be able to operate. The remaining control actions are intended to offer safety functionality. The operator can restrict the vessel from navigating into specific areas, by blocking off areas. This can be necessary in various cases, e.g. unfortunate weather conditions or an accident in the area. The remaining functions that are listed should be excessive once the ReVolt has become fully autonomous, but are included to let the operator override control in case of an emergency. The operator can request the video stream captured by the camera, request state data such as position and velocity, and provide an emergency stop command, if necessary.
The UML diagram is chosen to be a part of the system description in order to illustrate the context and the main functionality of the system. This diagram can be helpful in the process of defining accidents and hazardous scenarios.

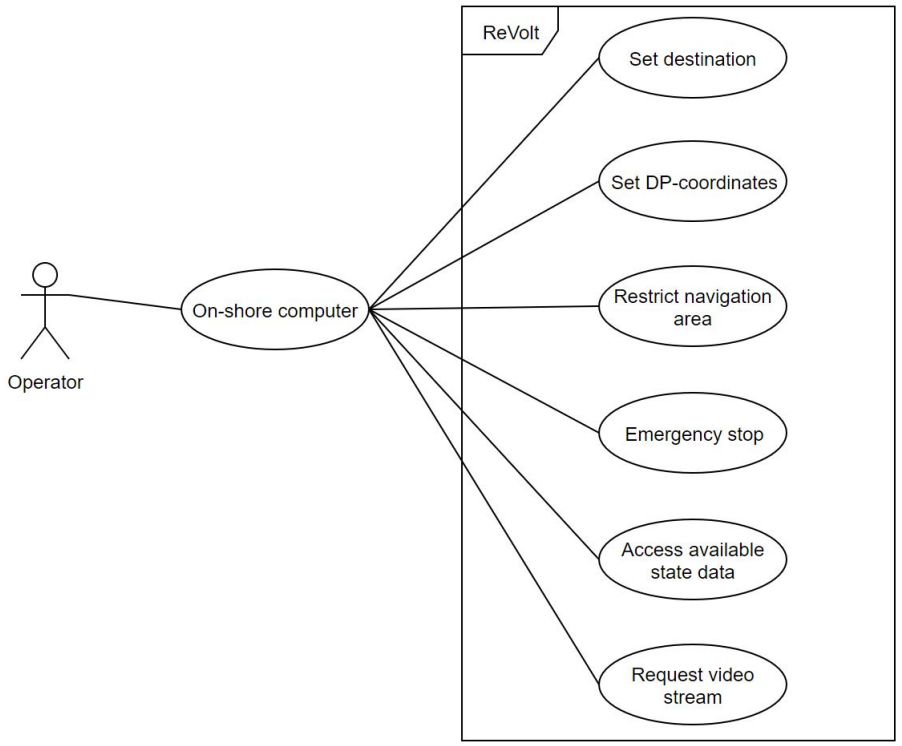

Figure 6.2: Use-case diagram of the ReVolt

6.2.2 Activity Diagram

The activity diagram is used to illustrate the dynamic behavior of the system (Delligatti; 2014), and it shows the typical workflow of the system (Fowler; 2004). Figure 6.3 presents a simple overview of the workflow of the ReVolt. Note that this diagram is not very detailed, and only contains the main functionality of the ReVolt.

Once the ReVolt has received the destination coordinates and it is given its current position estimate, it can calculate the route to the destination as a list of set-points. The system handles one point at the time, crossing off each point as it is reached. If an

obstacle is detected close to the future path of the ReVolt, the route must be modified, to avoid a collision. When the last set-point of the route is reached, the mission is completed, and the operator can provide a new destination, or a DP-setpoint.
Several activity diagrams can be designed to visualize all possible scenarios. More details can be included in the activity diagram illustrate everything that happens in the different scenarios of operation. In addition, activity diagrams can illustrate error handling and fallback strategies. This is not done here, but should ideally be done as soon as it is determined how possible malfunctions, errors or deviations should be handled with fallback strategies.

The activity diagram provides an overview of the possible system scenarios and the various chains of events that may occur. This UML diagram can also help define scenarios where fallback strategies must be designed. These are qualities that are useful to the STPA analysis.

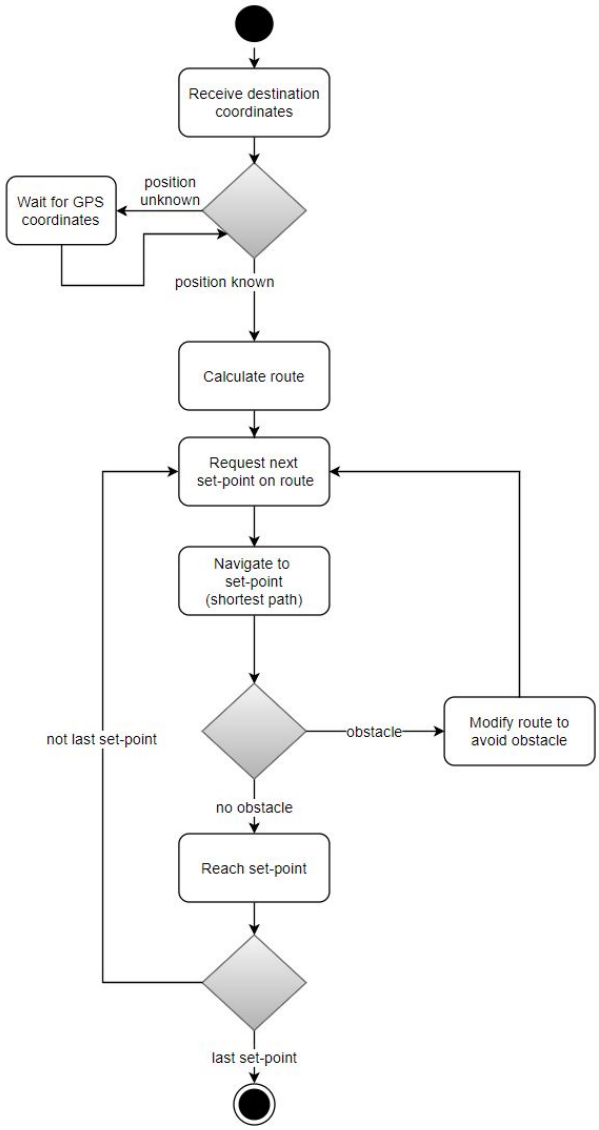

Figure 6.3: Simple activity diagram showing the behavior of the ReVolt

6.2.3 Sequence Diagram

The sequence diagram is an additional method for describing the behavior of a system. It is typically more detailed than an activity diagram. It illustrates the internal system workflow, and specifies how the various control actions are initiated by the internal controllers. Figure 6.4 shows a sequence diagram for the normal operation of the ReVolt.

1. The operator enters a destination, and the position sensors provide the current position of the vessel.
2. The navigation controller calculates the route to the destination as a list of points, referred to as set-points.
3. As long as a route exists, the navigation controller follows the list of set points.
4. The thruster allocation and control provides the control signals to the thrusters that results in the desired force and direction of the vessel.
5. The vision sensor data is continuously read by the obstacle avoidance.
6. When an obstacle is detected its position is estimated, and the navigation control is warned.
7. The route is adjusted accordingly, and normal operation continues until the final set-point is reached.

Note that not all controllers are shown here, to keep the diagram simple and easy to follow. The force controller's functionality is included in the navigation control. Thruster allocation and control are merged into one participant in the sequence diagram.

This sequence diagram only provides a brief overview of the normal operation of the ReVolt. One can go into greater detail showing the detailed flow of information and commands. An example is included in the Appendix in Figure A.1. It shows how control signals are continuously calculated and applied to the thrusters, including all the controllers.

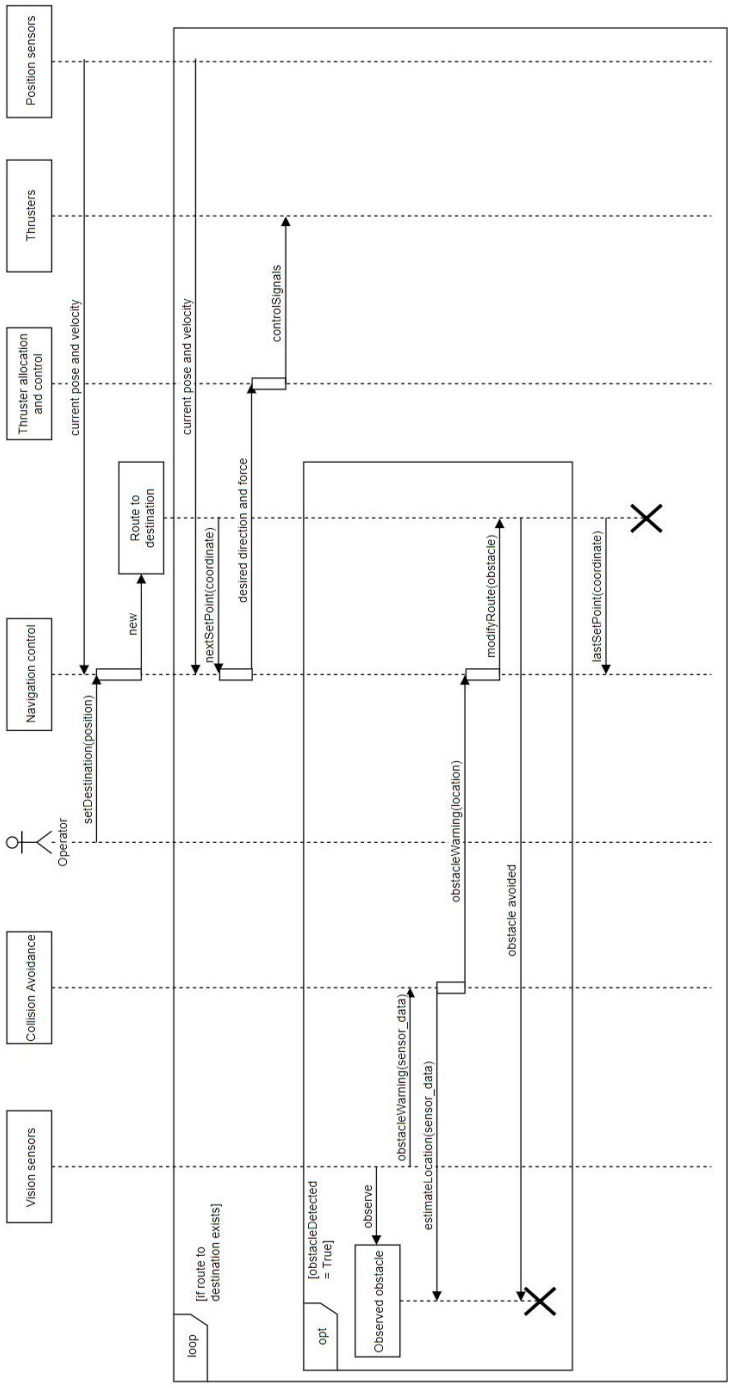

Figure 6.4: Sequence diagram

6.2.4 State Machine Diagram

The state diagram is another behavioral diagram. It describes the different system states and the transitions between the states (Fowler; 2004). The nodes illustrate the different system states, and the triggers on how to change between states are described along the transition lines. It is suited to describe the behavior of an object over many use cases (Fowler; 2004). The state diagram helps the STPA analyst, in finding the various functionality obtained by each software component. It can be a helpful tool when identifying potential fallback strategies. The state machine diagram that is presented in Figure 6.5 is a general, overall structure only containing three states. State diagrams that describe the sub-states of the state machine can be seen in the Appendix in Figure A.5 and A.6. More details are provided in these diagrams.

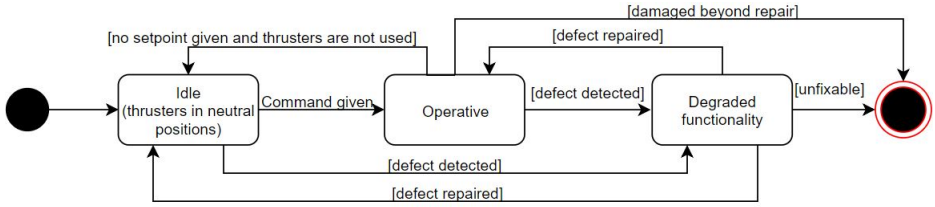

Figure 6.5: State machine diagram of the ReVolt

6.2.5 Class Diagram

The class diagram is a way of describing the structure of a system. It helps provide a description of the types of objects a system consists of and the relationships among these objects. Additionally, it describes the properties and operations of the classes Fowler (2004).

The class diagram of the ReVolt is quite large and does not fit in a single page. Therefore, it has been divided into multiple figures. Figure 6.6 shows a simplified version of the class diagram. The ReVolt, consisting of a set of controllers, actuators and sensors.

For more detailed figures illustrating the various sensors, actuators and controllers, check Figures A.2, A.3, A.4 in Appendix A.

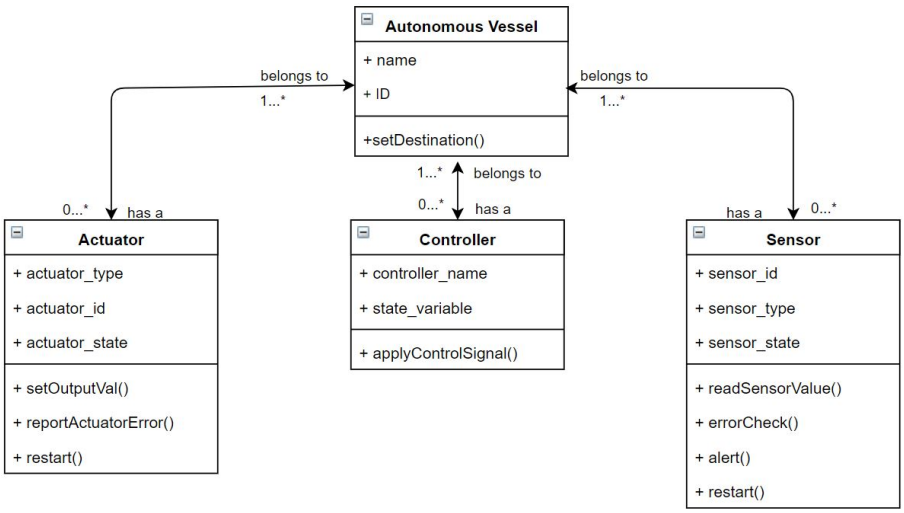

Figure 6.6: Top level of the ReVolt class diagram

6.2.6 UML contributions to the analysis

In the project assignment (Solberg; 2017), an STPA analysis was performed without having any documentation describing the ReVolt. The process was difficult, and the mental model of the ReVolt kept changing along with the analysis. Without keeping an exact model of the system the is being analyzed, the analysis is guaranteed to fail. Keeping a consistent process model is vital. This can easily be done with a set of UML diagrams and hardware documentation. Creating the documentation requires time, but it most definitely makes up for it. With a few diagrams, the STPA analysis can much more easily be completed, and is more likely to produce suitable system constraints and design guidelines. Thousands of lines of (poorly) written code can be replaced with a few comprehensible visual diagrams. It makes the STPA process much more efficient. The selection of UML diagrams may have to be reconsidered.

Those presented in the previous sections served well for a low-level STPA analysis. Once the analysis is taken to higher levels, the need for additional qualities may be revealed. UML is a great tool, which is easy to learn and use. The diagrams most certainly provide the STPA analyst with great advantages. During the STPA analysis presented in this thesis, the UML diagrams saved the analyst a significant amount of time.

6.3 Hardware

In this section, the hardware components onboard the ReVolt will be shown and briefly discussed. Figures have been drawn to show how the hardware on the ReVolt is connected. For more details on the hardware that has been used, see Figures C.2, C.3 and C.4 in Appendix C. These figures show images of the actual components that are used, but the same information is given in the next figures in this chapter.

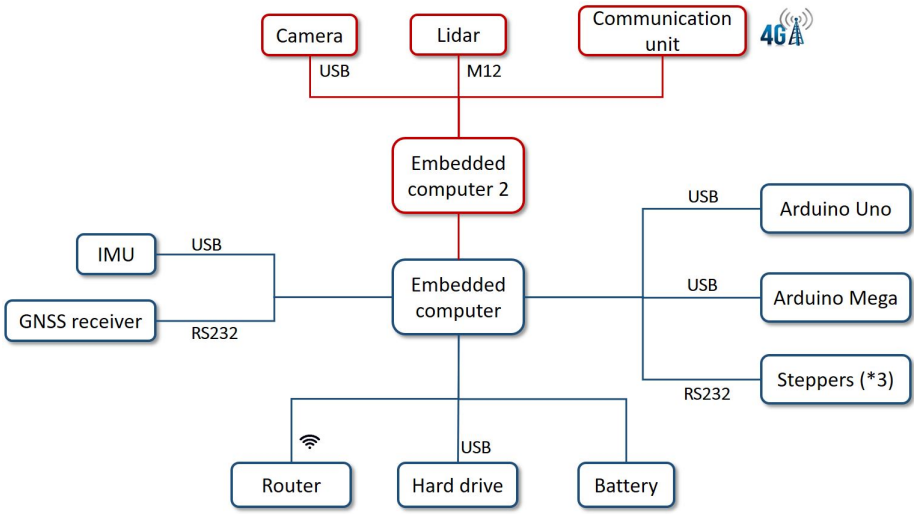

Figure 6.7: Showing the hardware that is connected to the embedded computer

Figure 6.7 shows the hardware that is connected to the embedded computer onboard the ReVolt. The red lines indicate the parts that will be installed onboard the ReVolt this semester. The type of communication media is indicated along the lines connecting the components. A few of them are missing a label, this is because it is not yet determined. Note that the Xsens sensor is labeled IMU, because during normal operation the Xsens serves as an inertial measurement unit. The Hemisphere sensor is the one labeled GNSS receiver. The router will most likely be removed once the 4G communication is up and running. The wifi is currently used for ROS communication with the on-shore computer. The remaining hardware on-board is connected to the Arduino micro-controllers. The details are shown in Figure 6.8 and 6.9.

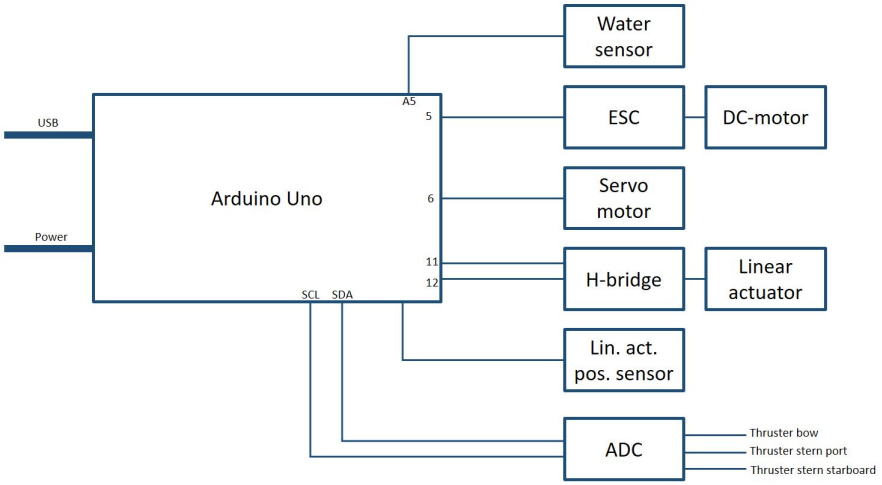

Figure 6.8: Showing the hardware that is connected to the Arduino Uno

Figure 6.8 shows the Arduino Uno and the hardware connected to it. It is mainly connected to the components of the bow thruster, the ESC, DC-motor, servo motor, linear actuator, and the sensor for the position of the linear actuator. A water sensor is connected to alert the system if water is detected inside the vessel. The last component shown in Figure 6.8 is an analog to digital converter (ADC). This component is not listed

in the previous documentation for the ReVolt, but is used in the code implementation. The Arduinos are placed inside boxes onboard the ReVolt, so it cannot be visually confirmed that they are there. According to the code that is written for the Arduino Uno, the ADC receive inputs from the bow thruster and the two stern thrusters, measuring the motor currents to detect if a thruster stops working.

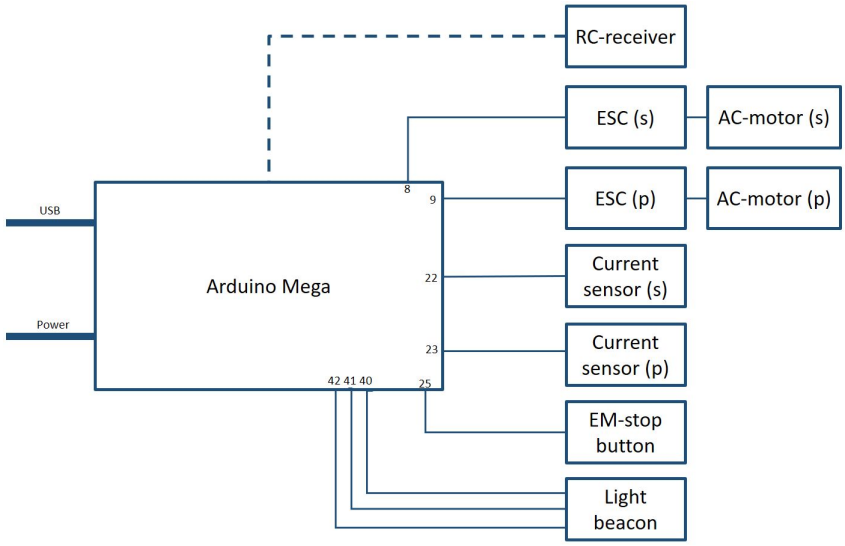

Figure 6.9: Showing the hardware that is connected to the Arduino Mega

Figure 6.9 shows the Arduino Mega and the hardware connected to it. The Arduino's main function is to control the effort of the stern thrusters. The emergency stop-button and light beacon are also connected to it. The RC-receiver is also connected to the Arduino Mega. It is shown with a stipulated line in the figure, because it will be removed as soon as remote-controlled operation is no longer necessary. The *current sensor (s)* and *current sensor (p)* are labeled `sensor_s` and `sensor_p` in the Arduino code. It is not explained further what these are in the previous documentation. In this thesis, it has been assumed that they are sensors measuring the thruster currents,

providing the ADC (located in the bow) with inputs.

6.3.1 RBD

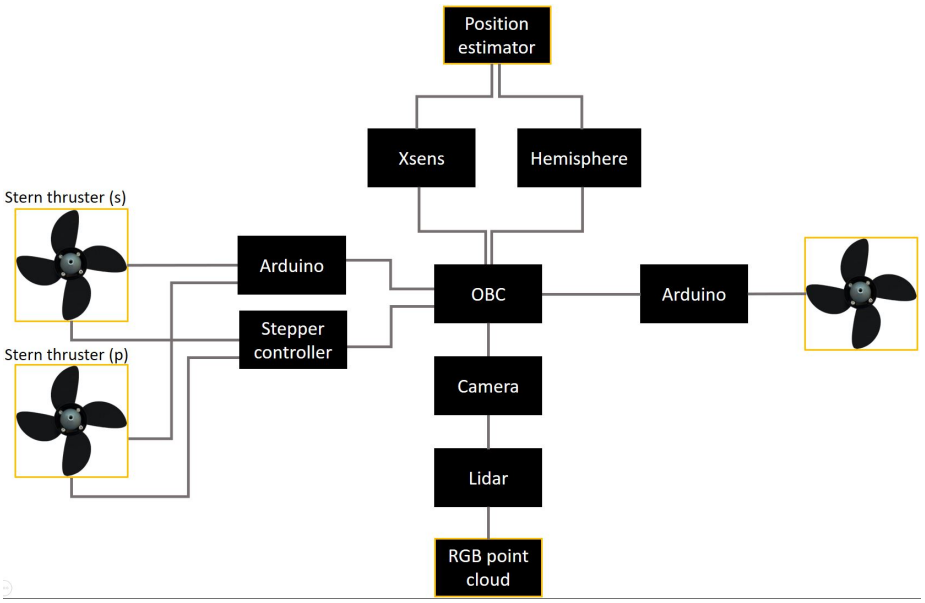

Figure 6.10: Simple redundancy block diagram

Figure 6.10 is a sketch of a potential RBD for the ReVolt. The OBC (on-board computer) is the embedded computer. Note how the Xsens and Hemisphere are connected in parallel to illustrate that one can work as a back-up for the other. The camera and the Lidar are connected in series, to show that both are required to work, in order for the RGB point cloud to exist. The RGB point cloud is a 3D model of the ReVolt's surroundings with RGB and depth information.
If the Arduino in the bow is lost, one cannot control the direction or the force of the bow thruster. The stern thrusters have two separate controllers for force and direction. If the stepper controller is lost, the stern Arduino can still be used to set the force of the thrusters.

Chapter 7

STPA Results

A low-level STPA analysis of the ReVolt has been carried out, and the results from the analysis will be presented in this chapter. Note that the ReVolt-model described in Chapter 6 is used. The results from the analysis can serve as design and implementation guidelines for those who will continue to develop and work with the ReVolt. The analysis is only taken to level four, which means that the analysis cannot be considered complete. However, it can provide indicators on where it is important to focus in order to increase the level of safety. Also, it is performed by a single analyst, which is not ideal. As was mentioned in Chapter 4, Step 2 of the STPA analysis may require a team of analysts, to find all accident causes and the required constraints. As a result, there is no guarantee that the results from the analysis can give a fail-safe system. Still, the analysis will contribute to a safety guided design strategy for the engineers. In addition, a higher level STPA analysis has been carried out using a part one of the ReVolt's controllers. The force controller was chosen to serve as an example. The purpose of this is showing that STPA can provide more detailed and specific safety constraints when it is taken to higher levels. The results from this analysis will be presented at the end of this chapter.

7.1 Overall System

To reduce the size of the analysis the number of accidents and hazards have been limited. This was agreed upon with the co-supervisor. Including all accidents and hazards in the analysis would make the analysis too time-consuming and too extensive for the scope of this thesis. However, a complete STPA analysis of the ReVolt should be performed in the near future to be ensure a safe system design. The accident that will be examined in this analysis is a collision, either with another vessel, the shore or another type of obstacle. This may be considered the most comprehensive type of accident the ReVolt can be prone to, and is central to the autonomous behaviour. This accident very comprehensive, and will lead to numerous system constraints and design guidelines for the system. Step 0 from the STPA analysis is shown in Figure 7.1.

Level	Type	ID	Text
			General
0	A	01	***Collision with other vessel/object***
1	H	01.1	**Vessel has unsafe speed/course in relation to other vessel/object/land, risking collision**
2	C	01.1.1	*Vessel must keep a safe speed/course in relation to other vessel/object to avoid close contact*
3	FR	01.1.1.1	Vessel must decide its safe speed/course in relation to other vessel/object
3	FR	01.1.1.2	Vessel must monitor/estimate present/future position of other vessel/object
3	FR	01.1.1.3	Vessel must monitor speed, course, heading
3	FR	01.1.1.4	Vessel must monitor/estimate present/future position
3	FR	01.1.1.5	Vessel must safely control the speed and course
1	H	01.2	**Vessel is too close to other vessel/object/land, risking collision**
2	C	01.2.1	*Vessel must keep a safe distance to other vessels/objects to avoid close contact*
3	FR	01.2.1.1	Vessel must decide its safe position/distance and heading in relation to other vessel/object
3	FR	01.2.1.2	Vessel must monitor positon of other vessel/object
3	FR	01.2.1.3	Vessel must monitor its positon/distance and heading in relation to other vessel/object
3	FR	01.2.1.4	Vessel must monitor its maneuvering capacity
3	FR	01.2.1.5	Vessel must safely control its positon/distance and heading in relation to other vessel/object

Figure 7.1: STPA Step 0, defining system-level accidents, hazards and constraints

For the chosen accident, there are two defined hazards; *The vessel has an unsafe speed and course relative to an obstacle, (heading towards an obstacle, risking collision)* and *The distance to an obstacle is too short to avoid a collision.* The functional requirements provide a brief overview of which parts of the system needs to be observed and controlled, in order to avoid the listed hazards.

Level	Type	ID	Description	Type
0	CTRL	01	***Operator***	Controller
1	CA	01.1	**Provide destination**	Control Action
1	CA	01.2	**Provide emergency stop command**	Control Action
0	CTRL	02	***Obstacle avoidance***	Controller
1	CA	02.1	**Share obstacle information (position, speed, course)**	Control Action
1	CA	02.1	**Collision warning**	Control Action
0	CTRL	03	***Navigation controller***	Controller
1	CA	03.1	**Provide a safe path from current position to destination (set of points, x_1 to x_n)**	Control Action
1	CA	03.2	**Provide a safe maximum speed**	Control Action
1	CA	03.3	**Provide safe courses**	Control Action
1	CA	03.4	**Recalculate path to avoid obstacle**	Control Action
1	CA	03.5	**Calculate maneuvering capacity**	Control Action
1	CA	03.6	**Estimate future position and heading (with the current control signals and conditions)**	Control Action
0	CTRL	04	***Force controller***	Controller
1	CA	04.1	**Provide the required force and moment that takes the vessel between to executive points**	Control Action
0	CTRL	05	***Thruster allocation***	Controller
1	CA	05.1	**Provide safe commands to bow controller**	Control Action
1	CA	05.2	**Provide safe commands to stern controller**	Control Action
1	CA	05.3	**Provide safe commands to stepper controller**	Control Action
0	CTRL	06	***Bow controller***	Controller
1	CA	06.1	**Set throttle**	Control Action
1	CA	06.2	**Set angle**	Control Action
1	CA	06.3	**Set linear actuator position**	Control Action
0	CTRL	07	***Stern controller***	Controller
1	CA	07.1	**Set throttle starboard**	Control Action
1	CA	07.2	**Set throttle port**	Control Action
0	CTRL	08	***Stepper controller***	Controller
1	CA	08.1	**Set angle starboard**	Control Action
1	CA	08.2	**Set angle port**	Control Action

Figure 7.2: The controllers of the ReVolt and their control actions.

Figure 7.2 shows the controllers of the ReVolt and their control actions. The control actions for the operator is reduced to a minimum, since it is not yet determined what

functionality will be required in the end.

Figure 7.3 shows a sketch of the hierarchical control structure of the ReVolt. It is slightly different from the template in Figure 4.5. The ReVolt does not have as many layers of controllers in the hierarchy of system development. For the ReVolt, the project management, the design and assurance are merged into one *controller*, consisting of only a few people; the students developing the ReVolt software and their supervisors. The *safety control and documentation* consists of the work presented in this Master thesis. Maintenance and evolution is also included in the figure, although at the moment this also is included in the design and development controller. However, at a later stage this will most likely become a separate controller in the hierarchical safety control structure. The operating process will be explained in greater detail in a separate control structure.

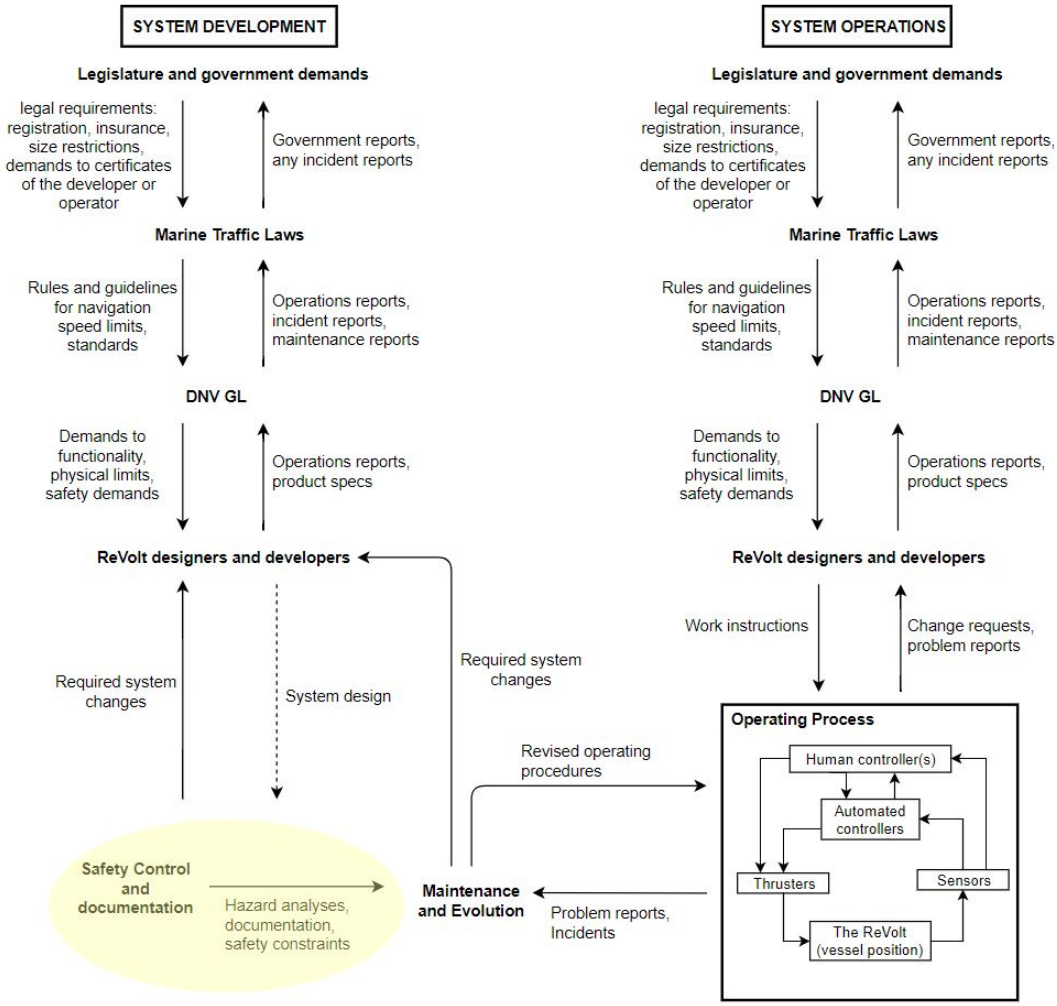

Figure 7.3: Hierarchical safety control structure for the ReVolt.

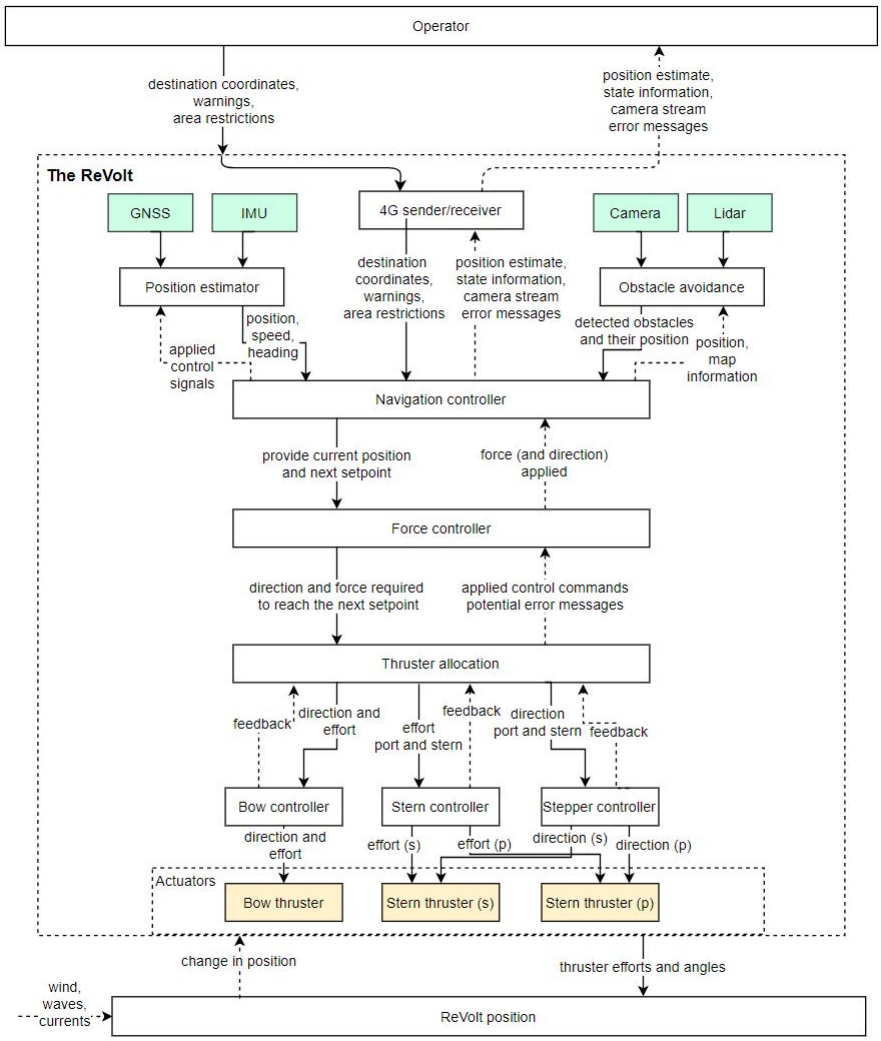

Figure 7.4: The detailed operating process of the hierarchical safety system control structure

Figure 7.4 shows the detailed operating process of the ReVolt. The ReVolt itself consists of a set of automated controllers, along with a set of sensors (shaded in green), and a set of actuators (in yellow). The operator communicates with the ReVolt using 4G communication. The controlled process here is the ReVolt's position. The sensors measure the position, linear and angular velocity of the vessel, and the controllers change it as desired by applying control signals to the thrusters.

Level	Type	ID	Description
H	H	(H01.1)	H01.1 Vessel has unsafe speed/course in relation to other vessel/object/land, risking collision
0	CTRL	(H01.1)01	***Operator***
1	CA	(H01.1)01.1	**Provide destination**
2	UCA	(H01.1)01.1.1	*N - Not provided when needed*
3	S	(H01.1)01.1.1.1	AC1 - The operator is not notified by the navigation controller that the vessel needs a new destination input
4	C	(H01.1)01.1.1.1.1	As soon as the ReVolt is started, or has reached a destination (completed an order) the operator should be notified that a new destination should be provided
3	S	(H01.1)01.1.1.2	AC2 - The operator is notified that a new destination is required, but ignores it
4	C	(H01.1)01.1.1.2.1	The operator must have complete overview of the situation before pushing EM-stop - a direct stop command from the operator should always be followed
3	S	(H01.1)01.1.1.3	AC3 -
4	C	(H01.1)01.1.1.3.1	no contraint required
3	S	(H01.1)01.1.1.4	AC4 - The communication channel between the ReVolt and the operator is flawed, and the cmd does not reach the ReVolt or is corrupted along the way
4	C	(H01.1)01.1.1.4.1	There should be a strict communication protocol between the operator and the ReVolt - lost messages must be resent until received, ckeck for corrupted messages (such as checksum)
2	UCA	(H01.1)01.1.2	*U - Provided but unsafe*
3	S	(H01.1)01.1.2.1	'AC1 - The operator provides an unsafe destination coodinate due to errouneous map information
4	C	(H01.1)01.1.2.1.1	The operator must always be provided with an updated map of the area. All map information must be correct, and taking the tide and other factors into account
3	S	(H01.1)01.1.2.2	AC2 - An invalid/unsafe input destination is not discarded by the control algorithm (the operator)
4	C	(H01.1)01.1.2.2.1	All destinations entered by the operator must be approved; are they within the operating range and is it valid
3	S	(H01.1)01.1.2.3	AC3 - The operator enters a desination whithout understanding it is potentially unsafe
4	C	(H01.1)01.1.2.3.1	The operator's instructions must clearly state which destinations are considered safe
3	S	(H01.1)01.1.2.4	AC4 -
4	C	(H01.1)01.1.2.4.1	no contraint required

Figure 7.5: The first 22 lines of Step 1 and 2 of the STPA analysis of the ReVolt

Using the excel framework developed by Glomsrud, step 1 and 2 of STPA are presented in one table. Figure 7.5 shows the a small selection from the table. The complete table is delivered along with the master thesis in the zip-file. The STPA-analysis for the first hazard, to level four, reaches 803 excel lines, and is too large to be included in the report. Therefore, a summary of the most important results will be presented.

Although the the analysis is only taken to level 4, it resulted a set of important indicators and design guidelines for the engineers. A number of safety constraints were produced, which can be found in the Excel table. Discussing each individual constraint is to extensive to be presented here. However, it has been attempted to make a summary of the most important result of the analysis. Among the most critical causes of the listed hazard are *loss of communication among system components*, *incorrect or missing sensor measurements* and *time delays in the reference channels*. The details of these causes will be discussed in the next pages, and represent the most important results of the analysis that has been performed. Taking the analysis to higher levels would have resulted in even more design guidelines and constraints.

Loss of communication

Losing the 4G-communication between the on-shore computer (used by the operator) and the ReVolt is critical. If the operator cannot provide new instructions, there is no way for the operator to control the ReVolt. It should be a priority to make the 4G-communication as fail-safe as possible. However, a fallback strategy must be designed. An alternative is to provide the ReVolt with a set of predetermined "safe-coordinates". In the scenario where the ReVolt detects communication is lost, it should start navigating towards the closest safe-coordinate. Implicit in this suggestion is that the connection between the operator and the ReVolt must be monitored. This can be solved by sending *pings* at a set time interval, and implementing a *watchdog* that times out and alerts the ReVolt if no ping is received. Pings are simply messages sent at a set time-interval to inform other systems it is *alive*. However, this fallback strategy is risky. Having an autonomous vessel running, with no possibility for controlling or monitoring is a big risk.

Onboard the ReVolt there are two Arduinos connected to an embedded computer. The thruster hardware is connected to the Arduinos, which applies them with control signals. If the Arduinos do not receive messages from the embedded computer, it is not possible to provide the thrusters with new control signals. Without being in control of the thrusters, there is nothing that can be done in order to avoid a collision. The Arduinos receive their instructions from the onboard computer through a USB cable.

Rosserial, a ROS library, which is used to handle this communication. It is vital that the communication between these components is working properly. There does not exist a backup solution for the communication between the Arduinos and the onboard computer. The connection between these components must be monitored.
A fallback strategy should be implemented that can handle the situation where the communication is lost. It is impossible to define a state for the thrusters that can always guarantee that the hazards will be avoided. The safest solution may be shutting off the thrusters completely. However, this is no way to guarantee that hazards will be prevented. The vessel may drift into an obstacle, causing an accident. Note that the Arduino located in the bow controls the bow thruster. The Arduino located in the stern controls the DC-motors in the stern. The stepper controller controls the angles of the stern thrusters. The connections are illustrated in Figure 7.6. It is not likely that communication with all three components is lost at the same time. Therefore, it is important to consider what should be done if *one* of the three components are lost.

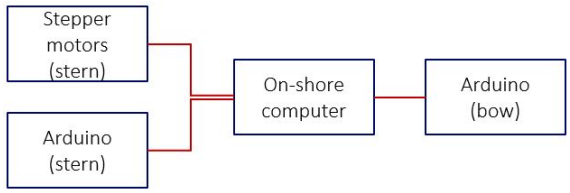

Figure 7.6: On-board computer and Arduinos

Is it possible to control the ReVolt if communication is lost with the stern Arduino or the stepper motors? If the bow thruster alone is strong enough to steer the vessel, an alternative mode must be defined for the navigation controller, force controller, and thruster allocation, based on the maneuvering capacity of the vessel in this state. If it is possible, this mode can be used to avoid hazards until the error is repaired.
Next, it must be considered: *Is it possible to control the ReVolt if communication is lost with the bow Arduino?* It must be examined whether the stern thrusters alone are able to precisely steer the heading of the vessel. If so, an alternative mode must be designed for the the navigation controller, force controller, and thruster allocation, based on the

maneuvering capacity of the vessel in this state. Note that the safety buffers, in terms of distance to obstacles and maximum speed, will have to be adjusted to compensate for the reduction in maneuvering capacity.
As soon as either of the Arduinos detect that the communication with the onboard computer is lost, the thruster should be set to neutral positions. If the stepper motors are not functioning, the stern thrusters should not be used.

The two embedded computers onboard the ReVolt will communicate with each other. The specifics of the type of communication that will be used, is not determined. However, it is crucial that the chosen type of communication provides a stable connection between the two computers. A brief description of the functionality of the computers is listed in Figure 7.7.

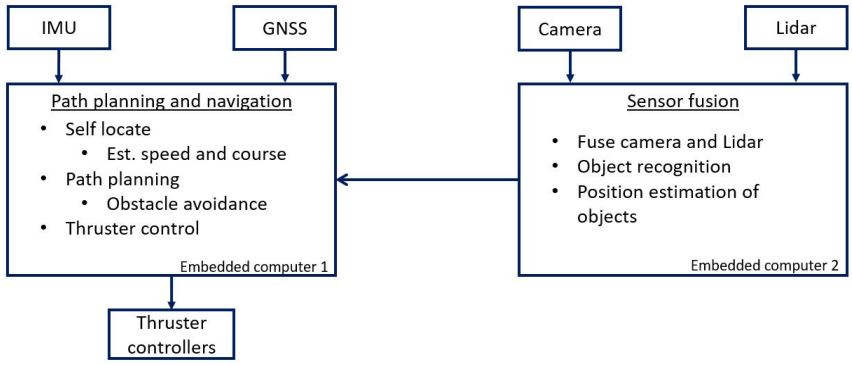

Figure 7.7: Illustration of the functionality of the embedded computers onboard the ReVolt

The embedded computer that performs the *path planning* is reliant on the data input from the *sensor fusion* computer, to avoid collisions. Without being able to sense its surroundings, the ReVolt cannot anticipate any potential collisions. There is no way to ensure that collisions are avoided without the data from the vision sensors. Thus, it is crucial that the communication is flawless. Finding a sufficient fallback strategy for the scenario where this communication is lost is difficult. The option that is closest to

guaranteeing safety is: setting the thrusters in neutral positions and cast the anchor. This means that it might be necessary to equip the ReVolt with an anchor. If this is not done, the ReVolt will either be navigating blind, or drifting in the water waiting for help, if the communication breaks down.

Incorrect or missing sensor measurements

The ReVolt has four different sensors that are used for navigational purposes; a Hemisphere GNSS receiver, an Xsens sensor (combined IMU, INS and GNSS), a Lidar and two RGB-cameras. The information from each sensor is crucial for the ReVolt to function properly, and to avoid hazards.

- Without data from the Hemisphere GNSS receiver, the ReVolt cannot measure its current position accurately. If the Hemisphere fails, the Xsens sensor can be used as a backup GNSS receiver. The accuracy of the Xsens' position estimate is ± 2 meters, and for the orientation it is ± 0.3 ° (roll), ± 0.3 ° (pitch), ± 1.0 ° (yaw). In comparison, the accuracy of the Hemisphere's position measurements is ± 2 **cm** (with RTK correction data). If the ReVolt has to navigate using the Xsens measurements, the safety buffers must be adjusted accordingly. Note that the safety buffers refer to the minimum accepted distance to an object, the maximum safe speed and so on, this was discussed in the project thesis (Solberg; 2017). The measurements from the Xsens should be used for navigation while the system attempts to restart the Hemisphere. If a restart of the sensor does not help, the Xsens should be used to navigate to a predefined safe-place, where the Hemisphere can be repaired.

- The only data that is normally used from the Xsens is from the inertial measurement unit (IMU); the roll and pitch. The Xsens provides a little higher accuracy than the Hemisphere does for these values. The accuracy for the Hemisphere is: ± 1.0 ° (roll), ± 1.0 ° (pitch), ± 0.1 ° (yaw). Note that the yaw-measurements are more accurate for the Hemisphere, and is normally used. If data from the Xsens is no longer available, it is possible to use the roll and pitch measured by the Hemisphere. As a consequence the safety buffers, such as maximum speed

or safe course, may have to be slightly expanded. Losing the Xsens is not very critical, as long as the Hemisphere is functioning properly. However, without the Xsens, there is no backup that can take over in case the Hemisphere loses functionality.

- The cameras are used to continuously capture the surroundings of the ReVolt as RGB images. Object recognition is used to detect obstacles on the water, islets, harbours and other obstacles. By combining the images from the cameras with the data from the Lidar, it is possible to estimate the position of detected obstacles. The video stream can be sent to the on-shore operator in real-time for supervision. If the cameras stop working, it is difficult to detect obstacles. It is possible to perform object recognition using the data from the Lidar alone, but this will require more advanced algorithms. It has to be considered which functionality is sufficient to implement on the ReVolt. Object recognition from depth data are among the qualities that may have to be implemented for safety assurance.

- The Lidar is used to map the ReVolt's surroundings as a 3D point cloud model. It can estimate the distance to everything around it. The Lidar that is planned to use has a range of 100 meters. Using this data, the obstacle avoidance can estimate the position of objects that are discovered by the object recognition algorithm. Without the data from the Lidar, the ReVolt cannot estimate the distance to obstacles around it. This information is necessary to perform path planning with collision avoidance. The safest fallback strategy for the scenario where the Lidar data is lost, may be going into DP-mode until the error is fixed.

It is important that the loss of a sensor is discovered as early as possible. The sooner the error is detected, the sooner the ReVolt can perform the planned fallback strategy. Continuing normal operation without one of these sensors can allow hazards to happen. If hazards occur, accidents and losses may occur.
A sensor may lose power, and will stop providing sensor data. Providing the most vital sensors with a backup power source may increase the reliability of the sensors. Malfunction of the sensors is difficult to predict, and the easiest way to handle sensor

failures is by redundancy. However, redundancy can also be expensive. The Hemisphere VS330 RTK Heading Receiver Kit alone, costs almost 100 000 NOK. Keeping an extra Hemisphere on the ReVolt as back-up in case the other one stops working, is an expensive solution.
Detect incorrect sensor data must also be considered. Sensors can "freeze", have spike values, offset values and so on. Using data like this can be dangerous. Logical checks must be set up for the sensor inputs, and any erroneous data must be rejected. However, it is important that good sensor data is not rejected. A real-time system needs all its sensor inputs to guarantee safety.

Time delays in the reference channels
It is important that the system is only working on the real-time data. Working on old data, believing it is real-time data, can have fatal consequences. All collected sensor data should be marked with a time-stamp. Before the data is used, their time-stamp must be checked and verified. Old data must be rejected. If the position estimator uses old data, the ReVolt's position belief will lag. An incorrect position belief will cause problems for the navigation controller. The navigation controller will attempt to control the ReVolt's position, while having an incorrect position estimate. The applied control actions will appear to have a different effect than what is expected. This may cause the system to believe there is something wrong with the Hemisphere, and choose to use the less precise Xsens data instead.
Working on old data from the vision sensors can be just as critical. Objects may not be detected until it is too late to avoid a collision. Also, the system may falsely believe that detected objects are further away than they really are. The surroundings of the ReVolt may have changed significantly since the vision data was recorded.
There are various reasons that may cause data to be delayed.
It might be a flaw or a failure in the transmission of the data in the reference channel. Using the ROS-communication protocol, it is possible to set up buffers at the receiving end of a topic, where incoming messages are stored until the system wishes to access them. It must be carefully considered which topics (communication channels) require a buffer and the consequences it might have. The data will be read from the topic in

the order it was received, meaning the most recent sensor data will be read last. All possible causes of delayed data should be studied, in order to avoid this from happening. These causes may be discovered in a complete STPA analysis.

7.2 The Force Controller

This is mainly aimed to be a study which investigates the potential of using STPA as the hazard analysis technique for a system like the ReVolt. It is too extensive for a single person to perform a complete analysis of the ReVolt. Therefore, as explained previously, the STPA analysis was stopped at level four. It provided some useful information for the system engineers. To fully investigate the method's full potential, it was chosen to attempt carrying out an analysis of a chosen controller to a higher level. For this purpose, the force controller was chosen. This decision was made together with with the supervisors.

Choosing an arbitrary controller turned out to be a poor strategy. One of the most important aspects of STPA is the hierarchical system structure; each controller enforces constraints on the controllers further down in the structure. Without having completed the analysis of the controller above the force controller, not all the necessary constraints have been enforced on the force controller yet. This may have resulted in more safety constraints being enforced on the force controller than what was necessary. Since the results would not be optimal, it was decided to choose only one of the UCAs of the force controller, which will still suffice for concept demonstration. The chosen scenario is:

Controller: *Force Controller*
Control Action: *Apply force command*
Unsafe Control Action: *Command is provided, but is unsafe.*

Figure 7.8 shows the selected part of the analysis at level 4. Figure 7.9 shows the expanded version of the analysis, when it is taken to higher levels. In this case, it is

2	UCA	(H01.1)04.1.2	*U - Provided but unsafe*
3	S	(H01.1)04.1.2.1	AC1 - The force controller provides an unsafe command because the trajectory from the navigation controller is unsafe
4	C	(H01.1)04.1.2.1.1	It must be verified that the information sent from the navigation controller is correct and safe
3	S	(H01.1)04.1.2.2	AC2 - The force command provided by the force controller is unsafe due to algorithm failure
4	C	(H01.1)04.1.2.2.1	It must be verified that all commands given by the force controller will not put the vessel in an unsafe situation
3	S	(H01.1)04.1.2.3	AC3 - An unsafe control signal is given, due to an incorrect process model held by the force controller
4	C	(H01.1)04.1.2.3.1	The process model must be correct and precise, it must be updated as new sensor data is available
3	S	(H01.1)04.1.2.4	AC4 -
4	C	(H01.1)04.1.2.4.1	no contraint required

Figure 7.8: STPA analysis to level 4

taken to level 8. The eight lines in excel are expanded to 31 lines with the higher level analysis. The analysis is almost four times as large. This provides a rough overview of the potential size of the complete analysis. The analysis to level four, for one hazard, exceeded 800 lines of code, which means that the completed analysis for the same hazard will most likely exceed at least 3200 lines. The analysis for one accident with two hazards may then exceed 6400 lines. This seems overwhelming for an inexperienced STPA analyst, who may not even know all the details of the ReVolt that well, due to the lack of documentation. It is also important to note that the analysis presented in Figure 7.9 may need to be taken to even higher levels. As was discussed in Chapter 4, it is difficult to know how many levels of the STPA are required to consider the analysis complete. Here, it seemed like the defined constraints gave restrictions that are specific enough for the engineers to work with.

2	UCA	(H01.1)04.1.2	*U - Provided but unsafe*
3	S	(H01.1)04.1.2.1	AC1 - The force controller provides an unsafe command because the trajectory from the navigation controller is unsafe
4	C	(H01.1)04.1.2.1.1	It must be verified that the information sent from the navigation controller is correct and safe
5	S	(H01.1)04.1.2.1.1.1	AC1 - The trajectory from the navigation controller takes the ReVolt too close to a detected obstacle with the
6	C	(H01.1)04.1.2.1.1.1.1	The necessary speed restrictions must be provided to the force controller before giving a trajectory that takes the vessel closer to an obstacle
5	S	(H01.1)04.1.2.1.1.2	AC3 - An unsafe force command is given because it is not detected that the starting point of the trajectory does not correspond to the current position (missing sensor measurement)
6	C	(H01.1)04.1.2.1.1.2.1	If starting point of a trajectory is not equal to the current position (within a set buffer), the force controller should request a new trajectory from the navigation controller
7	S	(H01.1)04.1.2.1.1.2.1.1	AC3 - The trajectory from the navigation controller has an incorrect starting position because the sensor values have "frozen" (do not changes)
8	C	(H01.1)04.1.2.1.1.2.1.1.1	The GNSS receiver must continously be checked for errors, such as frozen values, spike measurements, a stationary deviation (offset), lost power and so on
8	C	(H01.1)04.1.2.1.1.2.1.1.2	If sensor data from the GNSS receiver (Hemisphere) is missing or an error is detected, position data should be requested from the Xsens sensor
3	S	(H01.1)04.1.2.2	AC2 - A force command provided by the force controller is unsafe due to algorithm failure
4	C	(H01.1)04.1.2.2.1	It must be verified that all commands given by the force controller will not put the vessel in an unsafe situation
5	S	(H01.1)04.1.2.2.1.1	AC2 - The control algorithm provides a force command that violates the speed restricitons because it does not take the maximum speed restricition into account when calculating the required force and moment
6	C	(H01.1)04.1.2.2.1.1.1	The control algorithm must never provide a force command that leads to the speed restrictions being
7	S	(H01.1)04.1.2.2.1.1.1.1	AC2 - The controller provides a force command that violates the speed restrictions because the manuevering capacity estimate is not used correctly when calculating the force and moment
8	C	(H01.1)04.1.2.2.1.1.1.1.1	The control algorithm must correctly interpret the manuevering capacity estimate and take this into consideration when calculating the requires force and moment
3	S	(H01.1)04.1.2.3	AC3 - An unsafe control signal is given, due to an incorrect process model held by the force controller
4	C	(H01.1)04.1.2.3.1	The process model must be correct and precise, it must be updated as new sensor data is available
5	S	(H01.1)04.1.2.3.1.1	AC3 - The process model is not updated with GNSS sensor data, because the Hemisphere data does not reach the force controller.
6	C	(H01.1)04.1.2.3.1.1.1	If sensor data from the GNSS receiver (Hemisphere) is missing or an error is detected, position data should be requested from the Xsens sensor
5	S	(H01.1)04.1.2.3.1.2	AC1 - The force command is unsafe due to the manuevering capacity not being updated correctly
6	C	(H01.1)04.1.2.3.1.2.1	The force controller should be provided with the current manuevering capacities at frequently (at a set time interval). If no estimate is received when expected, a fallback strategy must exist.
7	S	(H01.1)04.1.2.3.1.2.1.1	AC3 - The manuevering capacity is incorrectly estimated due to flawed measurements on the ReVolt's position, linear- or angular velicity
8	C	(H01.1)04.1.2.3.1.2.1.1.1	There must be several logical checks for the maneuvering capacity to guarantee it is correct
5	S	(H01.1)04.1.2.3.1.3	AC3 - An unsafe control signal is given because the ReVolt's current linear and angular velocity estimates are incorrect
6	C	(H01.1)04.1.2.3.1.3.1	Malfunctioning sensors must be detected immediately and their data discarded
7	S	(H01.1)04.1.2.3.1.3.1.1	AC3 - Force controller provides a control signal that will increase the speed higher that the max. limit, because the sensors provide incorrect speed measurements
8	C	(H01.1)04.1.2.3.1.3.1.1.1	measurements
5	S	(H01.1)04.1.2.3.1.4	AC3 - An unsafe control signal is given due to sensor data on the vessel speed is missing
6	C	(H01.1)04.1.2.3.1.4.1	The GNSS data from the Xsens should be used if the Hemisphere stops working, in spite of worse accuracy
3	S	(H01.1)04.1.2.4	AC4 -
4	C	(H01.1)04.1.2.4.1	no contraint required

Figure 7.9: Partial STPA analysis to level 8

Chapter 8

Discussion

8.1 STPA Framework

8.1.1 Advantages and Disadvantages

The STPA framework designed by Glomsrud offers a set of advantages to the analyst.

- The Excel framework gathers all the information related to the analysis in one place. There are three separate tables, but they are all tied together in the end. The information from the first two tables; the accident and hazard overview and the controller overview (Step 0), is used to generate the framework for the table used in STPA Step 1 and 2. When the Step 0 tables are completed, one can simply start "filling out the gaps" in the auto-generated STPA-table. This is more intuitive to work with, and provides a well-organized structure. Compared to using the five separate tables that is originally designed for STPA, Glomsrud's framework is easier to work with.

- If any changes are done in the first two tables (Step 0), the changes are automatically updated in the final table. This way, it is easy to confirm that all tables are based on the same version of the system.

- The auto-generated table for Step 1 and 2 ensures that all potentially unsafe control actions are considered. This way, no UCAs can be forgotten or ignored. The only way a UCA can be left out of the analysis, is that the analyst incorrectly decides a control action is safe.
- The number-ID makes it easier to keep track of which hazards and controllers the scenarios and constraints are derived from.

Also, a few challenges have been encountered while working with the Excel framework.

- When performing Step 1 and 2 of the analysis to higher levels, the scenarios and constraints at all levels look more or less the same. It makes it hard to grasp the depth-perspective of the analysis. In the analysis shown in Figure 7.9, indentation is used to separate the higher levels from the low ones. The left-most column indicates the level of each scenario and constraint, but the indentation improves the readability of the tables.
- As explained in Chapter 4, each UCA must be analyzed using the control loop presented in figure 4.12, where the four types of accident causes are listed. *Accident cause* may be a misleading notation. They are causes of unsafe control. Unsafe control actions **might** lead to an accident, but will not always lead to an accident. However, it should be considered to add these to the auto-generated table. So that for each potentially unsafe control action, the table requires that all possible scenarios and causes of an unsafe control action are considered. In some cases, an unsafe control action may have more than one cause of the same category (AC1, AC2, AC3, or AC4). Meaning it can have two causes of category AC1. Where this is the case, it must be up to the operator to add the extra rows that might be necessary.
- With the STPA table design, it is difficult to add rows in the middle of the table for Step 1 and 2. If a small mistake is made when inserting a new row, the part of the table below the inserted row may become flawed. Some columns may become offset relative to the other columns. If possible, it can be helpful to develop a small set of functions for the framework, such as those shown in

Figure 8.1. The first two *buttons* add a single line, the third adds eight rows (4 scenarios and 4 constraints), the fourth one deletes a specified number of rows.

Figure 8.1: Proposed added functionality to STPA table for Step 1 and 2

8.1.2 Evaluation

The Excel framework has been easy to adapt to and work with. The changes that have been done to the Excel framework, since the project thesis (Solberg; 2017) was written, have improved the framework greatly. The results from the analysis performed in Solberg (2017) could not be used to create any guidelines for the future design of the ReVolt. The analysis performed as a part of this Master thesis is significantly larger in size, but it also contributes to several design guidelines for the ReVolt. This is a result of the changes done to the Excel framework. It assists the analyst in finding the natural next step of the analysis process. This framework is more intuitive and easy to use than Leveson's tables, of course depending on the preferences of the analyst.

8.2 UML Documentation

The UML diagrams presented in Chapter 6 were designed before the STPA analysis was performed. These diagrams were actively used as the analysis was performed. The diagrams served as support for the analyst, providing details of the system while the system was analyzed. The visual language made it easier to examine details of the system quickly, rather than having to find the correct part of the software or looking it up in (Alfheim and Muggerud; 2017a).
The reason why UML was chosen for this purpose is that it is considered an (unofficial) universal standard for software documentation. There are most likely other options for SW documentation that could serve the same purpose. However, most software engineers are at least acquainted with UML, which is very often the preferred tool

for software documentation and describing system behaviour. It is possible to create UML diagrams without knowing the entire UML language, when only choosing a few of the diagram types. It is quite common choosing to only work with a subset of the diagram types. Most of the UML diagrams are quite intuitive to understand for readers, and there is plenty of supplementary literature available online. Also, there are many options of open-source software that can be used to create the diagrams. For this project, draw.io has been used. It might be beneficial to use software that offers more functionality than this option, but it works for illustrative purposes.
The 13 types of UML diagrams are grouped into two categories; structural and behaviour diagrams. The STPA analysis is mainly analyzing the behaviour of a system. Hence, most of the UML diagrams chosen to advance the analysis are behavioral diagrams. The use-case, state machine, activity and sequence diagrams help describe the overall system behavior, as well as the detailed internal system behavior. The class diagram was chosen to identify all the components of the system and providing a brief overview of their functionality. The five chosen diagram types are among the most commonly used types of UML diagrams.

So, which advantages can the UML diagrams offer to the STPA analysis?
The analyst is usually not the one designing and implementing the system software. Understanding the complete functionality and all the details of a system one has not seen before may require a significant amount of time. Having to go through the process of describing a system's behavior can help the analyst get to know and understand the system better. This way, it reduces the number of times the analyst needs to go back to study details of the system he/she might have ignored or forgotten. Having a well-structured, visual representation of the system is advantageous. It makes it easier to find the answers to questions the analyst may have while performing the analysis. It is also easier to present the system to others, who are not familiar with the system, when UML diagrams are available.

STPA is a very young method, which still lacks a strong, solid framework. An STPA analysis will typically be quite large, and will require the analyst(s) to be very struc-

tured to guarantee that the analysis is performed correctly. By having the system properly documented with UML diagrams, it is easier to perform the STPA analysis, because system information is easily accessible. The easier it is to access required information about the system, the more effectively the analysis can be done.

Developing UML documentation for a system that is not yet finished can help strength the design. Using STPA along with the UML documentation, one can develop a system with a safety-guided design. In fact, UML diagrams should be created before a system is developed. It makes it easier to create well organized, structured system software.

By combining the UML documentation with the SAHRA framework, introduced in Chapter 3, UML could serve as a universal language for STPA. UML already being a universal standard for software documentation, it can be advantageous to base STPA on UML, by using an alternative such as SAHRA. By making UML documentation a mandatory, preparatory step of STPA and using e.g. the SAHRA framework, UML might become the universal language for STPA.

Is it really worth spending the time it takes to design and draw the diagrams? And was it easier to perform the analysis with the documentation in place? Creating UML documentation for a system you do not know can be challenging and time-consuming. Still, without it, the STPA analysis becomes even more challenging and time-consuming. When performing an analysis with only a mental model of the system, faults are doomed to happen. Keeping a consistent and complete mental model of such a complex system as the ReVolt is almost impossible. By keeping structured behavioural diagrams, as well as hardware documentation, the risk of introducing analysis failures can be significantly reduced.

8.3 Hardware Documentation

Having never worked with much hardware documentation before, it was challenging to know where to start when studying which types of documentation can be used.

Supervisor Onshus recommended using redundancy block diagrams. RBDs are diagrams which models the reliability of single components and how reliable a system is based on the composition of its components. This can be useful using alongside the STPA analysis when considering the need for backup solutions for critical hardware components. There has not been spent a lot of time designing RBDs, due to the lack of redundant components onboard at this time. In Chapter 7 it was discussed which sensors and actuators it can be critical to lose. It was concluded that it might be necessary to add backup components, which will increase the level of system redundancy. In this case, a redundancy diagram would provide more information.

The main motivation for including hardware documentation was simply to provide an overview of the hardware components and how they are connected. The designers of the ReVolt did not provide any wiring diagrams. This should ideally have been drawn before installing all the hardware onboard. Now that many components are hidden inside boxes onboard the vessel, it is difficult to find all the details. It has been attempted to create some approximate sketches, which were presented in Chapter 6. These sketches do not contain all details, but shows all the hardware components of the ReVolt and roughly explains how they are connected. Some information is missing from these diagrams, such as power sources for the individual components are not shown, because it is not possible to find without unscrewing the boxes, and taking them apart.

It is important to know which hardware components are used in the system and how they are connected while performing an STPA analysis. It is important that the hardware that is added or modified by others in the future is documented, if it is desirable to analyze the system safety. It is important to consider which components are critical for sufficient system operation.

8.4 Challenges

Even though working with both STPA and the ReVolt has been very interesting and educational, it has been slightly frustrating and challenging at times. The most significant challenges will be briefly discussed here.

- It has been emphasized several times that the lack of documentation has complicated the work in this thesis and in the project thesis (Solberg; 2017). Not knowing the system completely makes it difficult to perform a sufficient safety analysis. Every line of code and the work done in (Alfheim and Muggerud; 2017a) has been carefully studied. Based on the available information, the hardware was documented. An updated ReVolt version was designed for the purpose of making behavioral diagrams that can be useful for further development of the ReVolt. The updated model is based on the old control structure of the ReVolt, and is intended to show what the ReVolt is expected to look like. The actual ReVolt model will most likely differ from the model designed in this thesis. Hopefully the structures will be similar, so that the STPA analysis performed here can be used for future development.

- An STPA analysis of the ReVolt is an excessive task. An analysis like this should be performed by experts, who are familiar with the technique and the type of system. It has at times felt like a race with time, when not being completely aware of the extent of the analysis.

- Working alone on such a complex analysis is difficult. Having never worked with hardware and/or software systems this extensive before, it is a significant transition working with and understanding the ReVolt. It is stated that it is beneficial to perform Step 2 of the STPA with a team of analysts. It is challenging to come up with all potential accidents. Even with the improved Excel framework it is easy to ignore details that may have fatal consequences.

- Since STPA is a relatively new method it is difficult to find good examples. The method has been tested within a wide range of subject areas, such as autonomous

vehicles, aerospace, airplanes and more. However, the documentation of the STPA process itself is limited in most cases. Therefore, the main source of inspiration for how the STPA procedure is performed is found in Leveson's *train door example*. The train door is a basic example, only containing one controller with only discrete variables (e.g. train moving/train stopped, door open/door closed). The train door is not always sufficient for illustrating the challenges that can be encountered while working with STPA. Other examples include systems that are too complex to understand for someone who are not familiar with the type of technology.

To summarize, STPA is a difficult method, but it has great potential with the right aids and tools.

Chapter 9

Future work

There are endless possibilities when it comes to further work with both STPA and the ReVolt. Some of them will be discussed in this section.
Before the ReVolt is developed further, a concrete plan should be made for its goal state. All details should be planned and documented thoroughly. The hardware components must be planned for, and wiring diagrams must be drawn. The functionality can be visualized with a set of UML diagrams. A complete control structure should be designed to provide guidelines for the software design. The communication flow must be clearly stated to provide a well-structured ROS framework, this can be done with UML.

Once a satisfactory system description is in place, the continued system development can take place. If the chosen control system is very different from the one that has been used until now, the development process may have to start from scratch. Readability and structure must be the main focus of a project such as this one. Where the project is sent from one developer to the next without all the necessary information sharing.

With a well-documented, complete project plan, the STPA safety analysis can be

performed alongside the development. This way, the STPA analysis can provide useful inputs for the system developers. If it is decided to make any changes to the planned system design, it is important that all of the developers and the safety analysts are informed.
If the development process continues as it has until now, without providing detailed plans for development, it is almost impossible to guarantee safety.

In fact, it is planned to develop an autonomous shuttle ferry in Trondheim (Brekke; 2017). This ferry will most likely be based on the ReVolt. The ferry is planned to go from Ravnkloa to Fosenkaia, and to board up to 12 passengers at the time. It is an area where it is quite likely to meet other vessels, and will have to be able to navigate around them. The developers must be able to guarantee a safe system, that do not cause collisions. As has been demonstrated in this thesis, it is a complicated task. However, it is an extremely important task. It is important that the autonomous ferry project keeps safety as its main focus.

Appendix A

SW and HW documentation

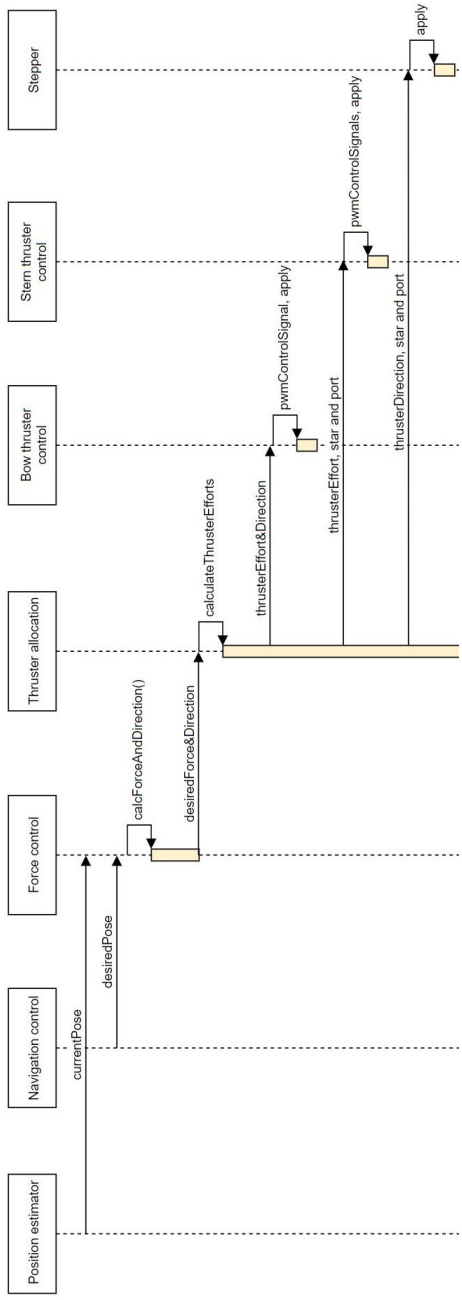

Figure A.1: Sequence diagram showing the detailed operation of control signals being calculated and applied to the thrusters

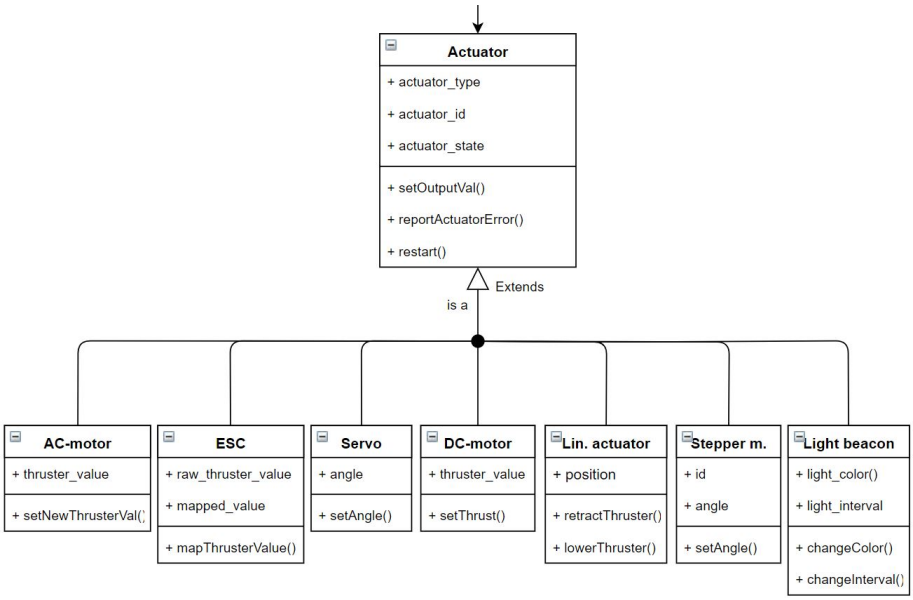

Figure A.2: Part of the class diagram, showing details of the system actuators

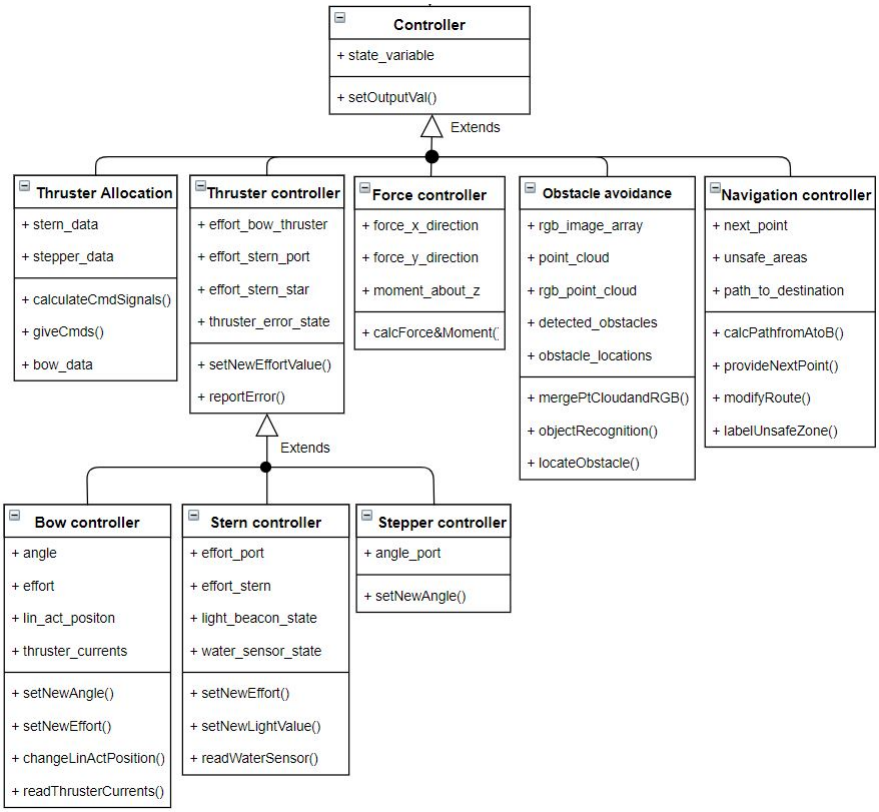

Figure A.3: Part of the class diagram, showing details of the system controllers

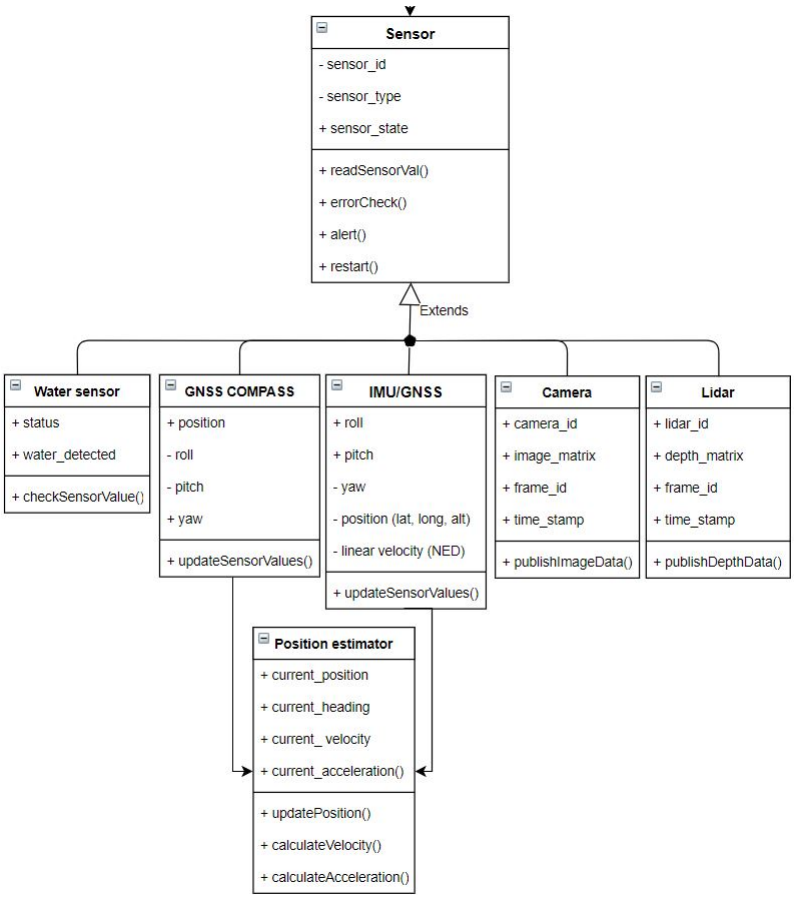

Figure A.4: Part of the class diagram, showing details of the system sensors

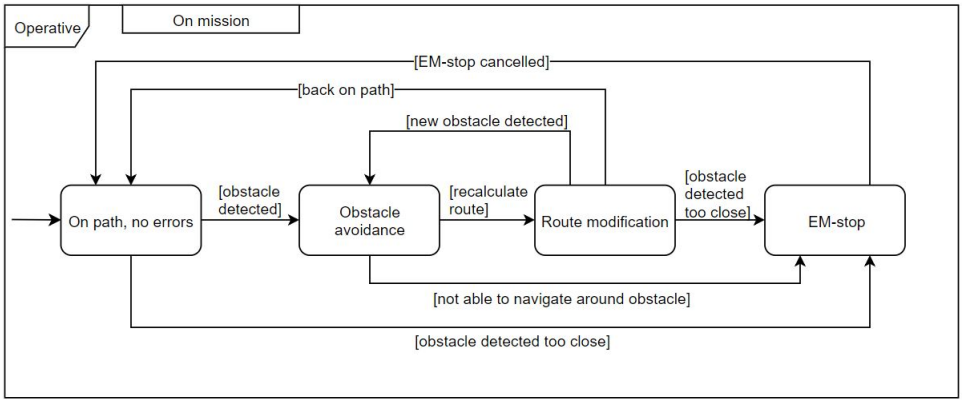

Figure A.5: Sub-states of the operative state

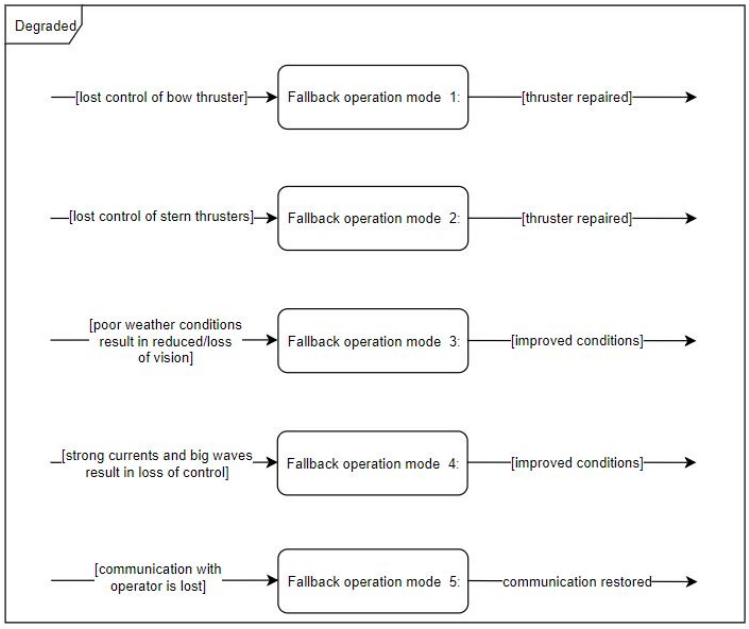

Figure A.6: Sub-states of the degraded state

Appendix B

Process models

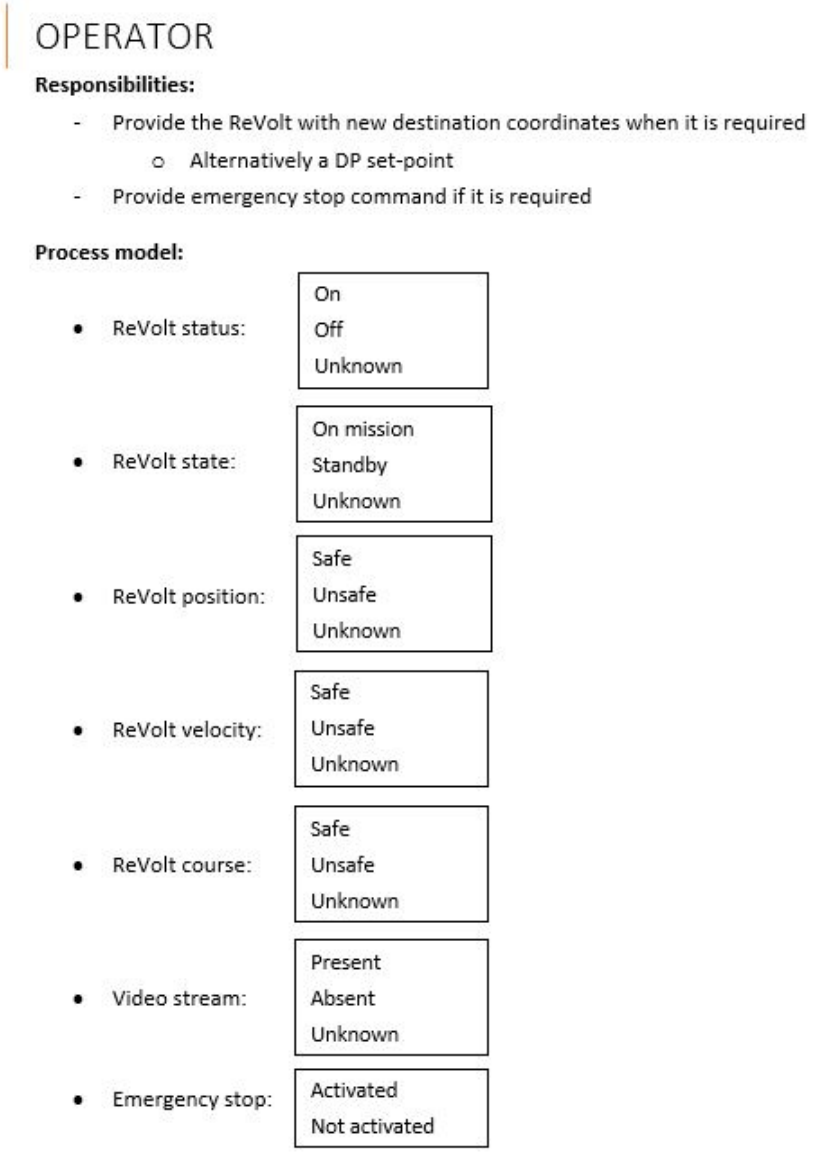

Figure B.1: Process model for the operator

OBSTACLE AVOIDANCE

Responsibilities:

- Create RGB point cloud of the ReVolt's surroundings
- Analyze data from cameras and Lidar
- Detect obstacles
    - Estimate obstacle position, velocity and course

Process model:

- Camera data: Present / Absent
- Lidar data: Present / Absent
- Obstacle detection: Obstacle present / No obstacle / Unknown
- ReVolt position: Safe (coordinates) / Unsafe (coordinates) / Unknown
- Obstacle position: Safe / Unsafe / Unknown
- Obstacle velocity: Safe / Unsafe / Unknown
- Obstacle heading: Safe / Unsafe / Unknown
- Collision: Potential collision / No nearby obst. / Unknown

Figure B.2: Process model for the obstacle avoidance

NAVIGATION CONTROLLER

Responsibilities:

- Calculate path to destination as a list of set-points
  - Modify the path accordingly based on the input from obstacle avoidance
- Provide limits for safe speed and course
- Calculate the current maneuvering capacity
- Estimate the future pose and velocity based on the control signals and the maneuvering capacity

Process model:

- ReVolt state: On mission / Standby / Unknown
- ReVolt position: Safe / Unsafe / Missing data / Unknown
- ReVolt velocity: Safe / Unsafe / Unknown
- ReVolt course: Safe / Unsafe / Unknown
- Obstacle detection: Obstacle present / No obstacle / Unknown
- Obstacle position: Coordinates / Not defined / Unknown
- Obstacle velocity: Safe / Unsafe / Unknown
- Obstacle heading: Safe / Unsafe / Unknown
- ReVolt path: Exists / Non-existent / Unknown
- Maneuvering capacity: 100 % / Reduced / Critically low / Unknown
- Bow thruster (error): Functioning / Fault present / Unknown
- Stern thrusters (error): Functioning / Fault present / Unknown
- Bow thruster: Effort (0-100 %) / Direction (±45°) / Lin. act. Pos. (up/down) / Unknown
- Stern thrusters: Effort (± 100 %) / Direction (±50°) / Unknown

Figure B.3: Process model for the navigation controller

FORCE CONTROLLER

Responsibilities:

- Calculate the total force and moment that takes the ReVolt from its current pose to the next desired pose

Process model:

- ReVolt position:
  - Known (coordinates)
  - Unknown
- ReVolt heading:
  - Known
  - Unknown
- Desired position:
  - Valid coord.
  - Invalid coord.
  - Unknown
- Desired heading:
  - Valid
  - Invalid
  - Unknown
- Current force and moment:
  - Force
  - Moment (about z)
  - Unknown
- Maneuvering capacity:
  - 100 %
  - Reduced
  - Critically low
  - Unknown

Figure B.4: Process model for the force controller

THRUSTER ALLOCATION

Responsibilities:

- Calculate the efforts required by each individual thruster to achieve the force and moment provided by the force controller

Process model:

- Bow thruster (error): Functioning / Fault present / Unknown
- Stern thrusters (error): Functioning / Fault present / Unknown
- Bow thruster: Effort (0-100 %) / Direction (±45°) / Lin. act. Pos. (up/down) / Unknown
- Stern thrusters: Effort (± 100 %) / Direction (±50°) / Unknown
- Maneuvering capacity: 100 % / Reduced / Critically low / Unknown
- Desired force and moment: Force value / Moment value / Unknown

Figure B.5: Process model for the thruster allocation

BOW CONTROLLER

Responsibilities:

- Pulse-width modulate the signals received from the thruster allocation
- Apply these signals to the correct output ports

Process model:

- Thruster effort:
  - Valid value
  - Invalid value
  - Unknown
- Thruster direction:
  - Valid value
  - Invalid value
  - Unknown
- Prev. thruster effort:
  - Valid value
  - Invalid value
  - Unknown
- Prev. thruster direction:
  - Valid value
  - Invalid value
  - Unknown
- Linear actuator state:
  - Retracted
  - Lowered
  - Unknown
- Linear actuator cmd:
  - Retract
  - Lower
  - No command
  - Unknown

Figure B.6: Process model for the bow controller

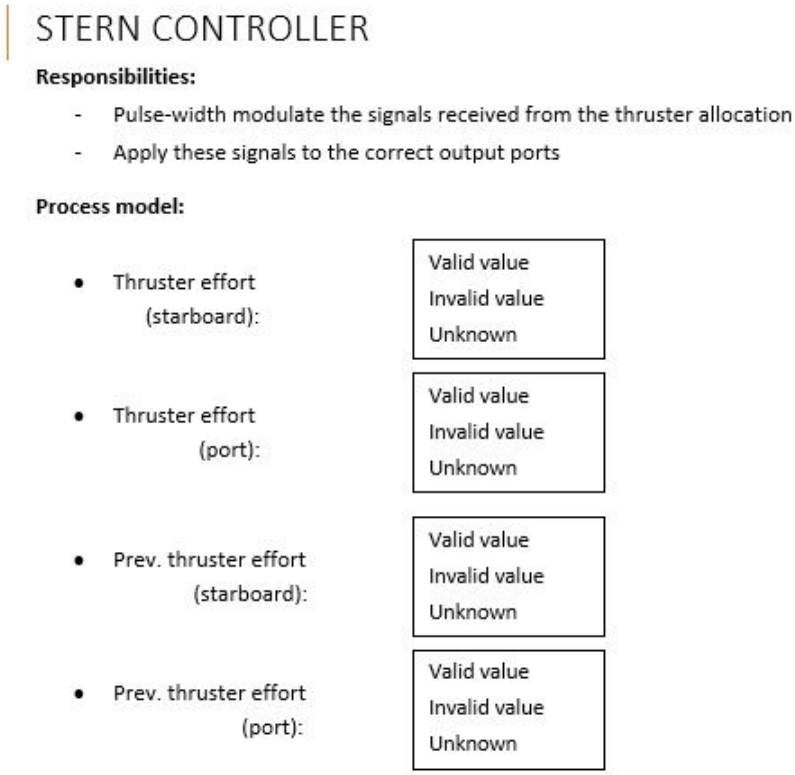

Figure B.7: Process model for the stern controller

STEPPER CONTROLLER

Responsibilities:

- Pulse-width modulate the signals received from the thruster allocation
- Apply these signals to the correct output ports

Process model:

- Thruster direction (starboard):
  - Valid value
  - Invalid value
  - Unknown
- Thruster direction (port):
  - Valid value
  - Invalid value
  - Unknown
- Prev. thruster direction (starboard):
  - Valid value
  - Invalid value
  - Unknown
- Prev. thruster direction (port):
  - Valid value
  - Invalid value
  - Unknown

Figure B.8: Process model for the stepper controller

Appendix C

Miscellaneous

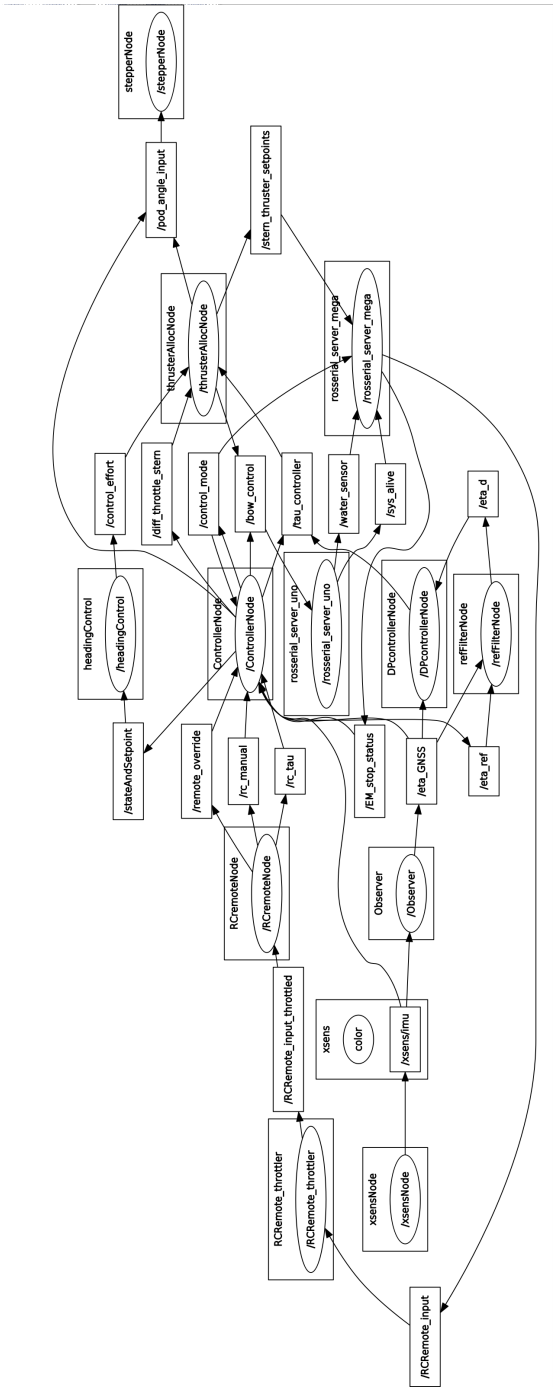

Figure C.1: Auto-generated ROS graph illustrating all ROS nodes and ROS topics - note that the Hemisphere is not shown in this graph

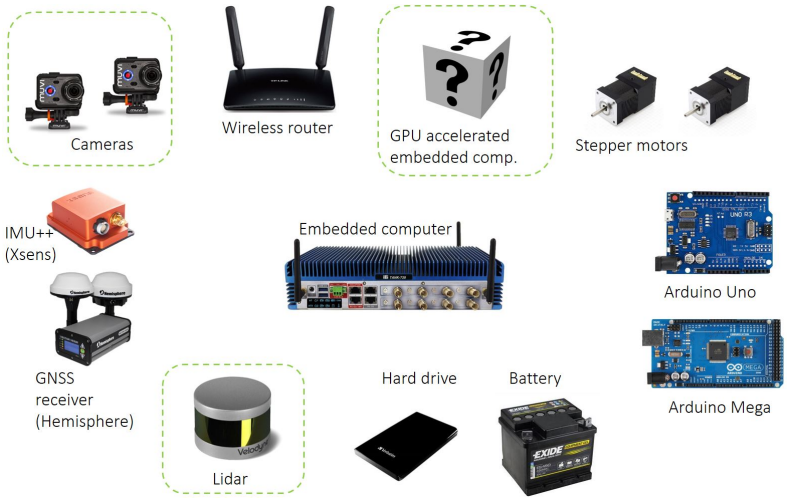

Figure C.2: Shows the components connected to the OBC

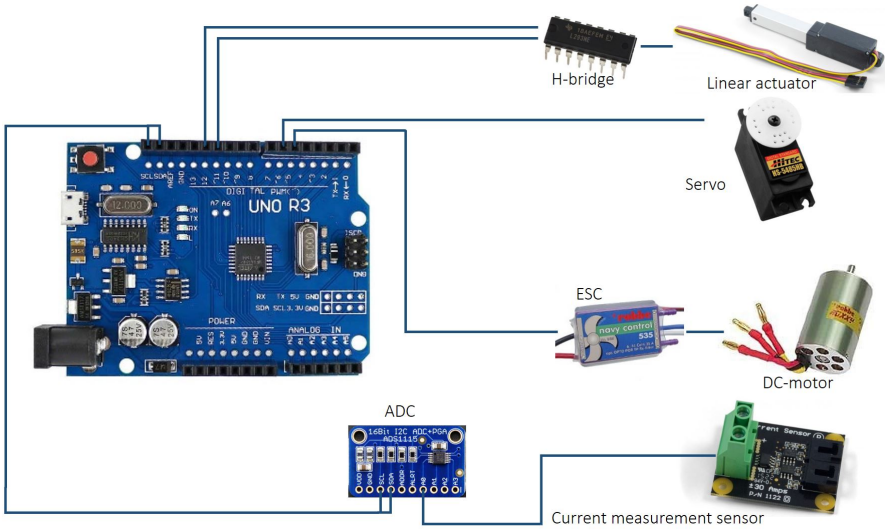

Figure C.3: Shows components connected to the Arduino Uno

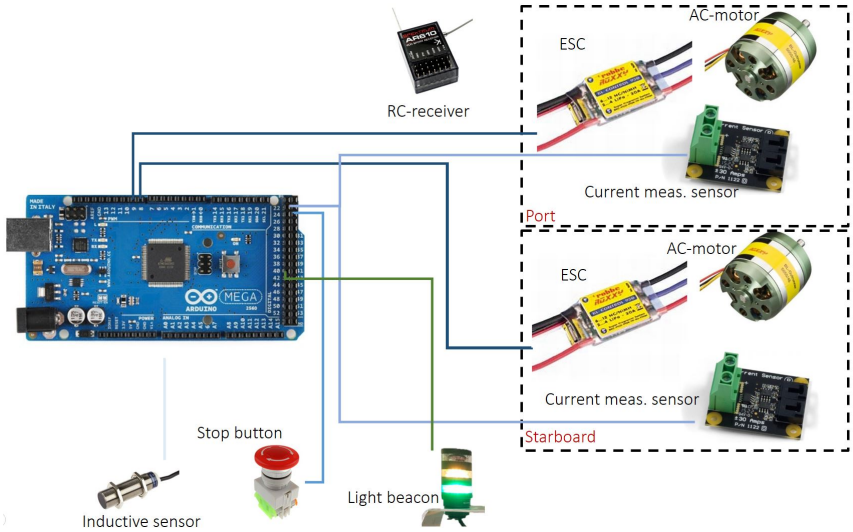

Figure C.4: Shows components connected to the Arduino Mega

References

Alfheim, H. and Muggerud, K. (2017a). *Development of a Dynamic Positioning System for the ReVolt Model Ship*, Master's thesis, Norwegian University of Science and Technology.

Alfheim, H. and Muggerud, K. (2017b). Revolt user manual.

Arduino Mega 2560 (n.d.). [Online; retrieved 26-May-2018].
URL: *https://store.arduino.cc/arduino-mega-2560-rev3*

Arduino Uno (n.d.). [Online; retrieved 26-May-2018].
URL: *https://store.arduino.cc/arduino-uno-rev3*

Brekke, E. F. (2017). Autonomous shuttle ferry in trondheim, "`https://www.sintef.no/globalassets/project/hfc/sarepta/4-ntnu-autonomus-ferry-efb.pdf`. [Online; accessed 31-May-2018].

CambridgeDictionary (2017a). Cambridge dictionary: definition reliable, "`https://dictionary.cambridge.org/dictionary/english/reliable`. [Online; accessed 5-December-2017].

CambridgeDictionary (2017b). Cambridge dictionary: definition reliable, "`https://dictionary.cambridge.org/dictionary/english/safety`. [Online; accessed 5-December-2017].

Delligatti, L. (2014). *SysML Distilled: A Brief Guide to the Systems Modeling Language*, Addison-Wesley Professional.

Denning, R. (2017). *Applied R and M Manual for Defence Systems, Part C Techniques. Reliability Block Diagrams Contents the Nature of a Reliability Block Diagram*, UK SARS.

Fowler, M. (2004). *UML Distilled, Third edition - A brief guide to the standard object modelling language*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Hemisphere (n.d.). Vector vs330 gnss compass, "https://hemispheregnss.com/Products/Products/Position-Heading/vector-vs330e284a2-gnss-compass-94. [Online; accessed 2-December-2017].

Krauss, S. S., Rejzek, M. and Hilbes, C. (2015a). Tool Qualification Considerations for Tools Supporting STPA, *Procedia Engineering* **128**: 15–24.
URL: *http://dx.doi.org/10.1016/j.proeng.2015.11.500*

Krauss, S. S., Rejzek, M. and Hilbes, C. (2015b). Tool Qualification Considerations for Tools Supporting STPA, *Procedia Engineering* pp. 15–24.
URL: *http://dx.doi.org/10.1016/j.proeng.2015.11.500*

Leveson, N. (2004). A new accident model for engineering safer systems, *Safety Science* **42**(4): 237–270.

Leveson, N. G. (2011). *Engineering a Safer World: Systems Thinking Applied to Safety*, The MIT Press.
URL: *http://medcontent.metapress.com/index/A65RM03P4874243N.pdf*

Leveson, N. and Thomas, J. (2015). *An STPA Primer.*

Safety-Critical Systems Research Lab Team of ZHAW, Z. U. o. A. S. (2017). The new enterprise architect extension for stpa sahra. [Online; accessed 1-April-2018].
URL: *http://www.sahra.ch/*

Solberg, C. L. (2017). Design and Verification of Safety Critical Systems, STAMP and STPA analysis of the ReVolt.

Thomas, J. (2013). *Extending and Automating a Systems-Theoretic Hazard Analysis for Requirements Generation and Analysis*, PhD thesis, Massachusetts Institute of Technology.

Tvete, H. A. (n.d.). The revolt, a new inspirational ship concept, "https://www.dnvgl.com/technology-innovation/revolt/index.html. [Online; accessed 20-November-2017].

Xsens (n.d.). Mti-g-710, "https://www.xsens.com/products/mti-g-710/. [Online; accessed 2-December-2017].