Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Selsebil A. Radi

Combining System-Theoretic Process Analysis with formal verification for safety demonstration of an all-electric actuation system

Master's thesis in Cybernetics and Robotics
Supervisor: Mary Ann Lundteigen
Co-supervisor: Ludvig Björklund
June 2023

NTNU
Norwegian University of
Science and Technology

Selsebil A. Radi

Combining System-Theoretic Process Analysis with formal verification for safety demonstration of an all-electric actuation system

Master's thesis in Cybernetics and Robotics
Supervisor: Mary Ann Lundteigen
Co-supervisor: Ludvig Björklund
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

This thesis is written as a part of the course TTK4900 for master students at the Department of Engineering Cybernetics at NTNU during the spring of 2023. The inspiration and background for this thesis comes from a previous specialization project titled "Using Safety Analysis to generate test scenarios of an All-electric Control System in Subsea Wells" [1], carried out as a part of the course TTK4550.

Emerging from a collaboration between SUBPRO - a center for research-based innovation in subsea production and processing technology led by NTNU - and my supervisor Mary Ann Lundteigen, this research contributes to ongoing research at SUBPRO for safety demonstration of the all-electric actuation system for safety valves. Employing a digital twin for this particular purpose is also a part of the project being conducted by Ludvig Björklund, a Ph.D. candidate and co-supervisor for this thesis.

The main focus of the research in this thesis is on the use of formal verification in combination with System-Theoretic Process Analysis (STPA) for safety demonstrations, especially for ensuring safety assurance of new software. Specifically, this thesis aims to explore the role of software verification in ensuring the safety of controllers that are software-dependent. To do this, the formal language Event-B is used, as well as its associated open source Eclipse-based IDE, The Rodin Platform [2].

The reader is assumed to have a concept of what safety implies, and how it differs from security. Some background in control theory is also an advantage.

Executive summary

Reliable and complex systems are becoming increasingly integral to industry solutions, necessitating more advanced safety analysis methodologies. This is particularly relevant in the exploration of novel approaches for safety valve actuation in subsea wells. The shift from traditional electro-hydraulic actuation to all-electric valve actuation introduces a new dimension of software-intensive safety systems, which require robust safety assurance mechanisms. To do so, System-Theoretic Process Analysis (STPA), known for being a suitable method for software-intensive systems, is conducted to create suitable constraints and demonstrate safety of the all-electric system. However, STPA has its limitations, such as the inability to tackle potential inconsistencies in requirements and insufficient validation of software systems.

To address these challenges, this thesis investigates an integrated approach to safety analysis by combining STPA with formal verification to strengthen the safety assurance of the all-electric actuation system. This is achieved by customizing the Security-Enhanced STPA (SE-STPA) to consider safety instead of security, resulting in a new methodology, SE-STPA-mod. SE-STPA-mod combines STPA with formal verification to validate system behavior and reduce ambiguities in STPA.

Incorporating formal verification in the safety analysis provides increased confidence in the software due to mathematical proofs of the system's safety properties. The proofs are generated by the suitable formal language to prove consistency and correctness of the model, by ensuring variables are well-defined, that constraints are not violated through the modelling and that the dynamics in the system are feasible. Additionally, formal methods allow for systematic exploration of system behavior by formally modeling every requirement. Having to model the system formally introduces a new way of identifying requirements, reducing potential software errors early in the development.

This thesis contributes to reducing risks in safety-critical systems by delineating a novel methodology - SE-STPA-mod. It elaborates on its application in constructing a formal model, as well as conducting an STPA, that effectively minimizes ambiguities. By ensuring continuous adherence to each constraint throughout the modelling process by the help of mathematical proof, a correct-by-construction software logic is created. Specifically, this approach has been employed for the all-electric actuation system, yielding a systematically verified model that underscores safety and reliability. While the safety analysis presented is a substantial step forward, it is not definitive. Complex systems are dynamic and constantly evolving, necessitating continuous improvement in safety analysis methodologies. Future research should explore additional methods, techniques, and tools to advance the safety analysis process, ensuring safer and more dependable system operation.

Sammendrag

Komplekse sikkerhetssystemer blir stadig mer synlige i industriløsninger, noe som nødvendiggjør mer avanserte metoder for sikkerhetsanalyse. Det er særlig relevant ved utforskningen av nye tilnærminger for sikkerhetsventiler i undervannsbrønner. Overgangen fra tradisjonell elektro-hydraulisk ventilaktuering til helelektrisk ventilaktuering introduserer en ny dimensjon av programvare i sikkerhetssystemer, som krever høy pålitelighet. For å oppnå dette, gjennomføres en System-Theoretic Process Analysis (STPA), som er kjent for å være egnet for programvaretunge systemer, for å demonstrere sikkerheten til det helelektriske systemet. STPA har imidlertid sine begrensinger, som manglende evne til å takle inkonsistente krav og at det tilbyr utilstrekkelig validering av programvaren.

For å takle disse utfordringene undersøker denne masteroppgaven en ny tilnærming til STPA ved å introdusere formelle verifiseringsmetoder for å styrke sikkerhetsgarantien til det helelektriske ventilsystemet. For å oppnå dette ble en ny metode utviklet ved å manipulere den eksisterende utvidelsen SE-STPA for å lage SE-STPA-mod-metoden, ved å endre på rekkefølgen og fjerne cybersikkerhetsaspektet med SE-STPA. Å integrere formell verifisering i sikkerhetsanalysen bidrar til økt tillit til programvaren. Dette er på grunn av matematiske bevis og matematiske definisjoner av systemets sikkerhetsegenskaper. Bevisene genereres basert på modellen i det egnede formelle språket for å bevise konsistens og korrekthet i modellen, ved å sikre at variabler er tydelig definert, at begrensninger til systemet ikke brytes og at systemets dynamikk er gjennomførbar. Formelle metoder tilrettelegger for en systematisk gjennomgang og utforskning av systemoppførselen ved å modellere krav formelt, som fører til en ny måte å identifisere krav på, og redusere potensielle programvarefeil tidlig i utviklingen.

Denne masteroppgaven har bidratt til å forsøke å redusere risiko i sikkerhetskritiske systemer ved å utvikle en ny metodikk, SE-STPA-mod. Bruken av denne metoden bidrar til redusere tvetydigheter gjennom konstruksjon av en formell modell, og gjennomføring av en omfattende STPA. Ved å sikre at modellen kontinuerlig verifiseres matematisk, kan systemets logikk garanteres å være korrekt og pålitelig fra start. Dette har spesifikt blitt brukt på det helelektriske ventilsystemet, og har resultert i en systematisk verifisert modell. Tilnærmingen legger grunnlag for nye strategier for risikoredusering i sikkerhetskritiske systemer. Selv om sikkerhetsanalysen som er presentert er et betydelig skritt fremover, er den ikke endelig. Komplekse systemer er dynamiske og stadig i utvikling, noe som nødvendiggjør kontinuerlig forbedring av analysemetodikker. Fremtidig forskning bør utforske flere metoder, teknikker og verktøy for å fremme sikkerhetsdemonstrasjon, og sikre tryggere og mer pålitelig systemdrift og programvare.

Acknowledgements

I would like to express my sincere gratitude to Mary Ann Lundteigen, my thesis supervisor, for her invaluable guidance, feedback, and support throughout my thesis work. Without her insight in the industry and relevant research papers, I would not have been provided with the knowledge needed for this thesis. I am also grateful to Ph. D. candidate Ludvig Björklund, which is currently researching the use of digital twins for safety demonstration, and the helping hand in providing a more technical insight for the research.

I am indebted to my friends and family members, who have provided unwavering support, encouragement, and motivation throughout my thesis work. Thank you for all the coffee breaks and unlimited fun. I would also like to thank my partner, without you, I would not have kept my spirits for the research going on throughout the semester.

I would also like to extend my gratitude to my fellow master students who are writing a report on the same topic of safety demonstrations. Without small discussions and status meetings, the motivation for the topic would not be the same.

Thank you all for your invaluable support, guidance, and encouragement throughout my thesis.

Table of Contents

List of Tables

List of Figures

Abbreviations

Abbreviation	Description
AC	Alternating Current
APS	Abandon Process Shutdown
BMS	Battery Management System
BOP	Blowout Prevention
CIXT	Chemical Injection to X-mas tree
DC	Direct Current
DCV	Dump directional control valve
DH	Downhole
DHSV	Downhole Safety Valve
DT	Digital Twin
FOC	Field Oriented Control
IEC	International Electrotechnical Commission
ISO	Internation Organization of Standardization
IPSMS	Interior Permanent Magnetic Synchronous Machine
NOG	Norwegian Oil And Gas Association
NORSOK	The Norwegian Self's competitive position
PM	Permanent Magnet
PMV	Production Master Valve
PMSM	Permanent Magnetic Synchronous Machine
PO	Proof Obligation
PSA	Petroleum Safety Authority
PST	Partial Stroke Testing
PWM	Pulse Width Modulation
PWV	Production Wing Valve
SE-STPA	Security Enhanced System-Theoretic Process Analysis
SIS	Safety-instrumented system
SOC	State-of-Charge
SOF	State-of-Function
SOH	State-of-Health
SOV	Solenoid-operated Valve
STPA	System-Theoretic Process Analysis
SPMSM	Surface Permanent Magnetic Synchronous Machine
SVPWM	Space Vector Pulse Width Modulation
UPS	Uninterruptible Power Supply
USV	Underwater Safety Valve

1

Introduction

1.1 Background

Low environmental impact and safe production is more important than ever, and the subsea oil and gas industry has a demand for innovative solutions that ensure safer and more efficient operations. One area witnessing a significant shift is the safety valve actuation system, traditionally dominated by electro-hydraulic technology, which now aims to be replaced with all-electric valve actuation. Recognizing the potential advantages of all-electric technology, a transition toward the development and implementation of an all-electric safety valve actuation system for subsea operations. One of the notable design changes is replacing the spring-assisted actuation of hydraulically operated safety valves with a motor and gear powered by a battery.

Traditional electro-hydraulic valve actuation primarily operates on a de-energize to safe principle. This mechanism relies on hydraulic pressure for actuation, which, while effective, comes with its own set of challenges, such as potential hydraulic leaks, maintenance complexity, and a lack of real-time diagnostics [3]. To address these issues, it has been proposed to replace the traditional system with an electrically actuated, motor-driven system powered by a battery, eliminating the dependence on hydraulic actuation [4].

The transition from electro-hydraulic valves to all-electric valves is not merely a change in power source but rather a substantial upgrade in operational capabilities. The new system offers extensive diagnostics of subsea hardware, a feature lacking in the traditional set-up. The electro-hydraulic system currently in use cannot provide diagnostics for undetected failures, such as a stuck valve, worn out springs, or unwanted leakage of hydraulic fluid [3]. In contrast, the electric valve system has the potential to identify these issues, facilitating timely interventions and maintenance operations, thereby enhancing the safety and efficiency of subsea operations.

Nevertheless, challenges related to the safety aspect of the change of valves and introduction of new software need to be addressed. As the new technologies include elec-

trical/electronic/programmable electronic systems, it is necessary to apply IEC 61508 [5] as part of the safety demonstration process to ensure reliability and quality through the entire lifecycle of the safety-related system. Safety demonstrations are defined as "documentation based on evidence and reasoning that adequate safety criteria are specified and met" [4]. One difficulty in demonstrating the safety of the all-electric solution is the lack of experience and data for how the system will manage safety related scenarios. A Ph. D. research project at SFI SUBPRO, a research-based innovation center led by NTNU, to investigate the development and use of digital twins to use for safety demonstration of the control systems involved in the actuation of the valves was initiated for this purpose [6]. The digital twin is intended to be used to evaluate the system software and the system behavior before installation.

A part of the challenge for the research project is to identify all the possible ways a system may fail, especially considering that the new control system depends on the interaction of several controllers, sensors and actuated devices. A specialization project explored the use of System-Theoretic Process Analysis (STPA) as a safety analysis method to identify unsafe control actions and fault scenarios to generate controller constraints.

However, none of the constraints nor requirements are validated in an STPA, and potential inconsistencies in requirements can occur.There is also a lack of a method to verify that the desired controller behavior is executed. A way of usually verifying behavior is done by providing the model with various test cases and simulating the behavior, but for a comprehensive model this can be quite time-consuming. Additionally, a simulator must be developed as well. This is most commonly known as simulation-based testing. Hence, the use of formal methods to mathematically develop models using formal languages and implement the models using available formal tools is a proposed method to verify the requirements developed from the safety analysis.

The scope of the previously conducted specialization project involved gaining an understanding of safety analysis methods, specifically STPA, as well as evaluating how a digital twin can be beneficial for testing. Mainly, this consisted of literature reviews and perform a partial analysis to gain an understanding of the proposed system. Nonetheless, this master thesis will investigate the possibility of combining theory from STPA with formal methods to identify a complete set of requirements that addresses potential conflicting requirements, and use formal verification tools to verify that requirements addressed in the STPA are handled. The use of formal methods might solve the lack of experience and data for the all-electric solution.

1.2 Objective and tasks

The previously conducted specialization project, as mentioned, primarily aimed to evaluate STPA as a safety analysis method without necessarily carrying out a complete analysis.

The objective of this master's thesis is to delve deeper into the application of STPA, specifically, its integration with formal software verification. The aim is to establish a method that can accurately validate that there are no inconsistencies in the system specification and requirements, as well as ensuring that safety constraints are not violated.

In accomplish this objective, the following tasks were defined:

- Familiarize with the safety-critical system and current solutions for safety valves
- Introduce a proposed solution for the all-electric control system
- Identify important goals and requirements to a safety demonstration process of a control logic for a safety-critical system
- Familiarize and describe existing approaches for combining STPA with software verification
- Introduce a new method for safety analysis combining STPA and formal methods
- Conduct a safety analysis for the all-electric actuation system
- Create a formal model of the system with insight from the safety analysis
- Discuss the benefits and limitations of incorporating formal modelling to the safety analysis and conclude

1.3 Approach

To gain an overview of the current state of research in the field and gain an understanding of how the different components in a subsea control system work, this thesis began with a literature review of the existing literature related to subsea actuation systems. This included research on the all-electric solution, as well as the electro-hydraulic valve actuation. In addition, research on a battery management system, suitable batteries, motor controller and a suitable motor for subsea valve actuation, to evaluate how the system works as a whole. Furthermore, a comprehensive review of standards and regulations provided by the Petroleum Safety Authority was done. This in-depth analysis serves as a foundation for understanding the safety requirements and guidelines applicable to the system that are to be studied.

In addition to the review of standards and regulations, the research also delves into the use of formal verification and Event-B. The goal of this study is to gain a deeper understanding of how these techniques can be employed to ensure that the proposed solution adheres to established safety guidelines, and to verify its overall safety and reliability. This is all done to create a solid fundament for the safety demonstration of the new all-electric actuation system. The safety demonstration approach also encompasses a Systems-Theoretic Process Analysis and examines how this method can be integrated with formal verification for a more comprehensive safety analysis, culminating in a new methodology, SE-STPA-mod.

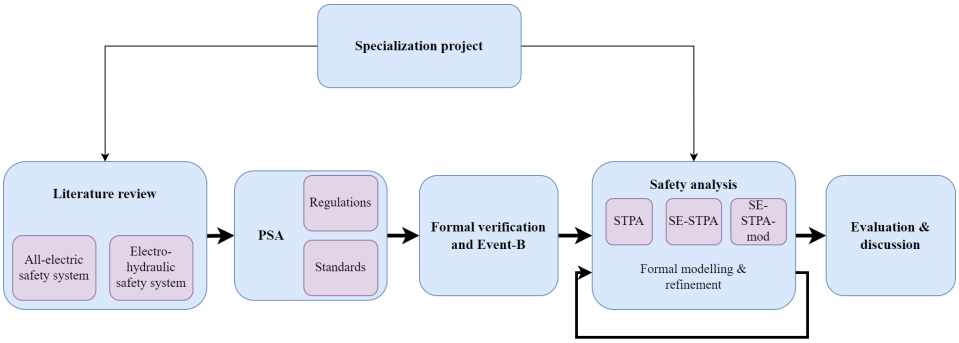


Figure 1.1: Research approach

Creating a formal model for this thesis required a combination of trial and error, as well as an extensive study of mathematical notation and mathematical proofs. This iterative approach allowed for the refinement of the model and the development of a robust representation of the system's behavior. By continuously revising and improving the formal model, this research seeks to provide a solid foundation for the safety analysis of the system under investigation, ultimately contributing to the development of a safer and more reliable solution. The analysis is then evaluated and discussed, offering insight into the relevance and limitations of this approach. The research approach is summarized visually in Figure 1.1

1.4 Assumptions and delimitations

The formal language and verification platform employed in this thesis were not selected as part of the analysis scope. Their use, namely Event-B for language and Rodin as the verification tool, was predetermined due to prior application within SE-STPA related reports. However, these were related to valve actuation systems, so it's assumed that these tools remain suitable for this safety analysis.

As for the all-electric actuation system, descriptions and specifications focus on one isolated subsea well. Any requirement or limitation beyond the safety analysis's system boundary is not considered. System function details that are irrelevant to the safety analysis are also omitted.

Normally, STPA is carried out in a workshop setting with field experts. However, for this thesis, the STPA analysis was performed by the author, guided by thesis supervisors, and not in the typical workshop setting. Additionally, the STPA was not completed to its fullest possible extent, as the main goal of this thesis was to explore the possibility of combining safety analysis with formal verification. Therefore, the analysis primarily focused on uncovering a sufficient amount of specifications required to create a formal model.

1.5 Structure of the report

Chapter 2 outlines the issues with the existing actuation system in subsea wells and justifies the rationale for transitioning to an all-electric system. It also introduces the current x-mas tree configuration and presents the intended architecture for the new system. Furthermore, this chapter provides insights into subsea control systems, including relevant standards in the Oil and Gas industry. In addition, the proposed solution for all-electric valve actuation, along with descriptions of the Battery Management System (BMS) and motor controller, are introduced as part of Chapter 2.

Chapter 3 is influenced by the previous conducted specialization project [1], and presents the STPA approach for safety analysis.

Chapter 4 introduces the concepts of formal verification and formal methods, with a particular focus on the formal language, Event-B. This chapter explains the foundational principles and common concepts of Event-B, to provide a comprehensive understanding of how this formal language can be applied. Furthermore, this chapter will present an enhanced version of the STPA method, called SE-STPA. A new analytical approach, specifically developed by the author for the purposes of this thesis, will also be outlined. This new method, referred to as SE-STPA-mod, has been tailored to fit the specific type of analysis conducted in this thesis.

Chapter 5 delves into a detailed safety analysis of the all-electric actuation system using the SE-STPA-mod method, which was introduced in Chapter 4. This comprehensive analysis encompasses a review of all safety aspects of the all-electric actuation system. Additionally, it includes the development of a formal model that outlines the expected behavior of the system under consideration.

Chapter 6 discusses the benefits of integration of formal verification into the STPA analysis, as well as limitations of the SE-STPA-mod and the use of Event-B as a suitable language for this thesis.

Chapter 7 concludes this thesis by summarizing the findings from the discussion in chapter 6. Suggestions for further work are also presented.

2

Safety demonstration of the all-electric control system

This chapter outlines the key standards that subsea control systems are required to adhere to, and describes the safety demonstration framework provided by IEC 61508 [7] and the Safety 4.0 project [4]. It also provides detailed descriptions of the various system components that make up an electro-hydraulic valve actuation. Finally, it presents the proposed solution for all-electric valve actuation, influenced from the previously conducted specialization project [1].

2.1 Background

The offshore oil industry is continuously growing in a time when environmental sustainability is more important than ever. Consequently, a need for pollution-free and safe production is essential. The current solution for safety valve actuation is electro-hydraulic, and even though they are accepted in current standards and regulations, they come with some challenges [8]. One challenge is the fact that valves based on electro-hydraulic actuation do not provide any diagnostics. When the valves are installed, the industry has to rely on them being maintenance free. Hence, without diagnostics, there is no way to tell if a valve stem is stuck or not available during an emergency shutdown, and the operators have to rely on occasional testing.

An all-electric valve actuation system for subsea safety valves has been introduced for SUBPROs research project because of the need to reduce impact on the environment, reduce risks and gain a high level of safety. The level of safety and reliability of the valves are in conjunction with the available diagnostics from the electrical system, which will also play a part in reducing maintenance cost [4]. Nevertheless, the strict requirements for the oil industry require the change of control system to be completely safe, pass strict testing before use and have a comprehensive safety demonstration. A lot of requirements

on testing and development follow criteria from standards, such as IEC 61511 [9], IEC 61508 [7], NORSOK S-001 [10] and NOG 070 [11].

2.2 Relevant standards

In Norway, the Petroleum Safety Authority (PSA) manages petroleum activities, particularly those offshore, with an emphasis on safety. The regulations set by PSA include risk- and performance-based requirements, which specify all critical activities in every phase of oil and gas operations. In order to ensure compliance with the PSA framework for the new proposed system, some relevant standards will be presented here, and further elaborated throughout the report.

Standards provided by the PSA for offshore activities play a crucial role in enhancing the efficiency and effectiveness of various operations and functions, because of clear safety guidelines. They also provide operational consistency, and eases the coordination and co-operation of different offshore departments. There exists International standards, such as those offered by the International Electrotechnical Commission (IEC) or the International Organization for Standardization (ISO), and national standards, which are specifically designed for the Norwegian industry. The Norwegian Oil and Gas Association (NOG) develops national standards under the banner of The Norwegian shelf's competitive position (NORSOK) to ensure sufficient safety and efficiency in the Norwegian shelf [12].

IEC 61508 [7] is an international standard for the functional safety of electrical, electronic, and programmable electronic safety-related systems. It is designed to ensure that such systems are designed, implemented, operated, and maintained in a way that minimizes the risk of hazardous failures. IEC 61508 also includes guidance on aspects such as risk assessment, safety requirements, hardware and software design, verification and validation, and maintenance. Each topic is treated as a dedicated party of IEC61508, which in total consists of 7 parts. It emphasizes the importance of a systematic and integrated approach to functional safety, with clear roles and responsibilities for all involved parties. It outlines a well-defined process for safety demonstrations and stipulates the certification requirements for safety-critical components.

IEC 61508 [7] is known as an umbrella standard because it provides a general framework that can be applied to various industries and applications. It is designed to be application-independent, offering the flexibility to be tailored to specific sector requirements [4]. An example of how IEC 61508 can be adapted to cater to a particular industry is the IEC 61511 [9] standard. This standard is a sector-specific adaptation for safety instrumented systems (SIS) in the process industry. These safety instrumented systems are expressly designed to protect process plants from incidents that could inflict harm to people, the environment, or assets.

IEC 61511 has simpler requirements than IEC 61508 because it is tailored to a specific type of safety systems [9]. This is because IEC 61511 has a primary focus as to how to integrate IEC 61508 to project-specific applications. Nevertheless, IEC 61511 provides a

framework for the entire lifecycle of a safety instrumented system, including the specification, design, implementation, operation, and maintenance of the system. It also provides guidance on the verification and validation of safety instrumented systems, including testing and documentation.

Furthermore, these standards can also be specifically tailored for the Norwegian continental shelf in NOG 070 [11]. NOG 070 is a Norwegian oil and gas industry standard that sets out guidelines for the management of major accident risks in offshore drilling and well operations. Mainly, NOG 070 is a further tailoring of IEC 61508 and IEC 61511. Different risk reduction measurements and assessments are described in NOG 070. Additionally, NOG 070 contains lists of basic assumptions for the x-mas trees, which can be seen as minimum requirements that need to be adapted, as well requirements for specific subsea safety instrumented functions such as isolation of one subsea well [4].

Another standard which specifies safety design for offshore oil and gas facilities is NORSOK S-001 [10]. Notably, it is explicitly referenced in PSA regulation §33 concerning emergency shutdown systems. NORSOK S-001 sets out guidelines for the design and selection of safety systems and equipment in offshore installations, and provides guidance on safety critical elements, safety barriers, evacuation and rescue, fire and explosion protection, and safety instrumented systems. NORSOK S-001 is more specifically used to describe principles for design of safety systems for offshore oil and gas facilities. It also provides requirements for the selection of safety equipment, such as fire and gas detection systems, emergency shutdown systems, and life-saving appliances. Additionally, NORSOK U-001 [13] can be used for subsea production systems utilizing electro-hydraulic control systems based on requirements from ISO 13628 [14].

ISO 13628, especially ISO 13628-4 [14] provides specifications for subsea wellheads, as well as necessary tooling to handle, test and install the equipment. ISO 13628 consists of 12 parts, but only the part 4 covering subsea wellhead and tree equipment as well as and part 6 covering subsea production control systems will be relevant for this thesis. ISO 13628-6 [15] applies to design, fabrication, testing, installation and operation of subsea production systems. Additionally, it covers subsea-installed control systems and equipments which is relevant for the all-electric control system. ISO 13628 is recommended to be used to design prudent well control in x-mas trees by regulation §54 from the PSA [16].

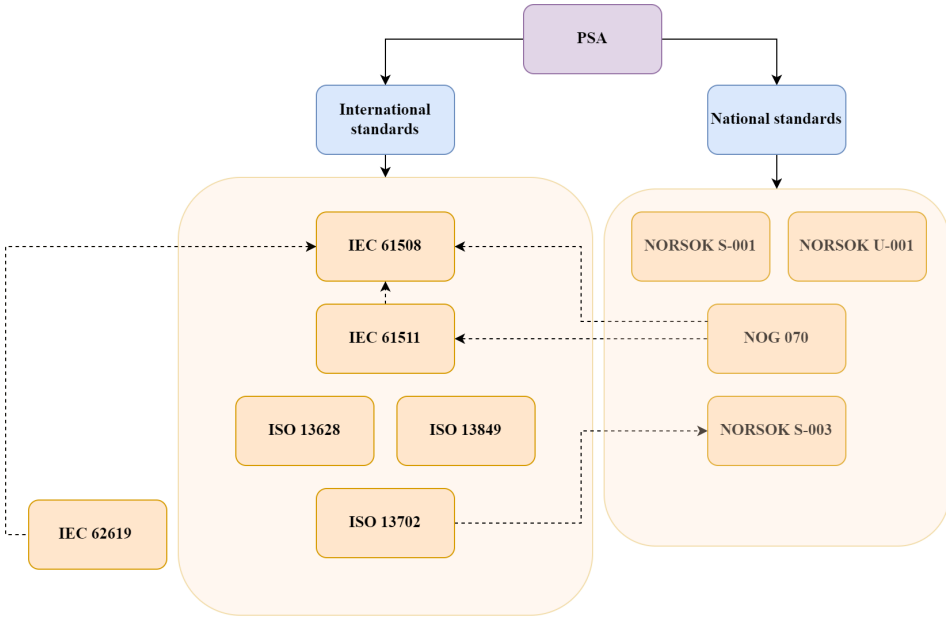


Figure 2.1: Overview of standards used for the all-electric actuation system

The selection of standards typically hinges on the end application of a product, and these standards often serve as a reference point for manufacturers, developers, and industry professionals. For instance, when dealing with an electronic programmable safety-related system, IEC 61508 [7] is the pertinent standard. However, it's worth noting that individual system components may also have their own unique standards that should be taken into consideration.

Take, for instance, the case of batteries. Batteries serve as a backup power source for the all-electric actuation system, and they are subject to safety requirements detailed in IEC 62619 [17]. This standard also describes the usage of a Battery Management System (BMS), which could be relevant for the backup power supply. Although the PSA does not mention this specifically, it contributes to the understanding of overall system safety and hence its inclusion in Figure 2.1.

Nevertheless, the PSA provides a relevant standard in §38 for Emergency power and emergency lighting offshore to be ISO 13702 [18], a standard which describes requirements for control and mitigation of fire and explosions offshore. In addition to ISO 13702, NORSOK S-003 [19] provides guidance for energy efficient design and operation of energy facilities which might be appropriate considering the energy storage needed for an electric solution.

The standards outlined above work in conjunction with the regulations established by the PSA. Figure 2.1 presents an overview of the interconnections between these various standards. The dashed lines in the figure represent relationships between the standards, demonstrating how they can be adapted and applied to one another.

The PSA, as previously mentioned, oversees inspections, audits, and investigations, verifying that platforms and the petroleum industry in Norway are adhering to stipulated regulations. Utilizing the relevant standards is critical to maintain safety and efficiency within oil and gas operations. Additionally, it creates a standardization for the different companies and industries, as well as creating a quality assurance. These standards, including risk- and performance-based requirements, ensure safe and efficient operations. The proposed system must therefore adhere to the PSA framework, thereby guaranteeing safety and reliability within Norway's offshore industry.

2.3 Safety demonstrations

Safety demonstrations refer to activities or evidence used to showcase the safety of a system or process, and are conducted to assure that a system or process is designed, implemented and operated safely [4]. Essentially, they validate that the proposed safety solutions and constraints are either equivalent to or exceed the performance of the existing ones in terms of their safety efficacy.

For systems which are based on Electrical/Electronic/Programmable Electronics, IEC 61508 [7] provides a guidance on the overall process for demonstrating the safety of a safety-related system. According to IEC 61508, the safety demonstration process includes the following steps:

1. Hazard and risk assessment
2. Safety requirement specification
3. Safety design and implementation
4. Safety verification and validation
5. Safety review and approval

These five steps can be seen as a summary of the 16 steps of the safety lifecycle provided in IEC 61508-1 [20]. The hazard and risk assessment concerns determining hazards, hazardous events and hazardous situations of the novel solution. Additionally, the system boundaries and the scope of the analysis, as well as reasonable safety measures, will be assessed. It is important to identify potential risk associated with the safety-related system and determine the severity of the consequences in this step.

The safety requirement specification is developed on the basis of a comprehensive hazard and risk assessment, which then forms a set of safety requirements. Subsequently, the safety design stage ensures that these requirements are appropriately addressed. This stage may encompass the selection of specific hardware and software requirements, along with the definition of critical safety functions and safety measures.

The safety validation and approval stage necessitates a rigorous verification process to ensure the design has been implemented accurately and maintains the safety requirements. This is achieved through testing and continual monitoring of system compliance with the

defined standards and regulations. Essential to this process is thorough documentation, resulting in a comprehensive report that not only states the system's current safety, but also its ability to maintain to be safe throughout the safety functions' lifetime.

Nonetheless, a project with an objective to develop a framework for standardized demonstration of safety, especially supporting digitalization, is The Safety 4.0 project [4]. Drawing inspiration from the principles of IEC 61508, this framework addresses the unique challenges posed by these new technologies and provides a structured and comprehensive approach to safety validation. An overview of the guidelines provided in the Safety 4.0 framework is summarized in Figure 2.2. The framework is designed to account for the full lifecycle of a system, from initial concept and design through to decommissioning, providing ongoing assurance of safety at every stage.

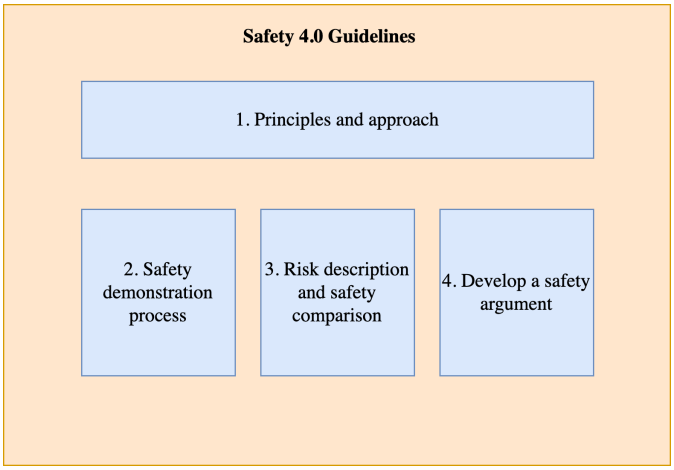


Figure 2.2: Safety 4.0 guidelines. Adopted from [4].

The safety demonstration part of Safety 4.0 focuses on what need to be done to demonstrate safety efficiently and effectively [4]. This includes activities related to risk assessment and technology qualification. Describing risk and comparing safety outlines how risks should be described and compared, and last but not least, Safety 4.0 describes how to develop arguments to prove that the novel technology is safe.
By integrating principles from IEC 61508 [7] and other relevant safety standards, Safety 4.0's fusion of safety and digital advancement opens up new possibilities for enhancing safety performance through the utilization of advanced technologies. It therefore ensures a seamless integration of safety practices within modern technological systems, improving the levels of safety and reliability [4].

2.4 Subsea control system

In offshore oil and gas production, a key component that ensures efficient operation is a subsea control system. Specifically tailored for subsea production systems, this technol-

ogy enables remote control and monitoring of an array of subsea components involved in the production process. These include wellheads, which are often integrated with an x-mas tree, manifolds, and flow lines[8]. A typical subsea control and production system including all the mentioned parts can be seen in Figure 2.3.

Situated at the interface between the well and the production system, the wellhead is a crucial component that provides a vital seal and control point for fluid flow from the well. It typically comprises a casing head and a tubing head as support structures, a set of valves to control flow, and connectors to attach the wellhead assembly together [8]. The wellhead serves as the foundation for surface control of the subsurface flow, effectively preventing leaks and uncontrolled discharge [21]. Atop the wellhead sits the x-mas tree, an array of valves configured to manage the flow of oil and gas from the well. The components installed on an x-mas tree play a crucial role to ensure safety in the well, and are designed to prevent accidents and protect the environment in the event of a control issue. How the wellhead and x-mas tree should be designed to achieve prudent well control is described in §54 [16] provided by the PSA.

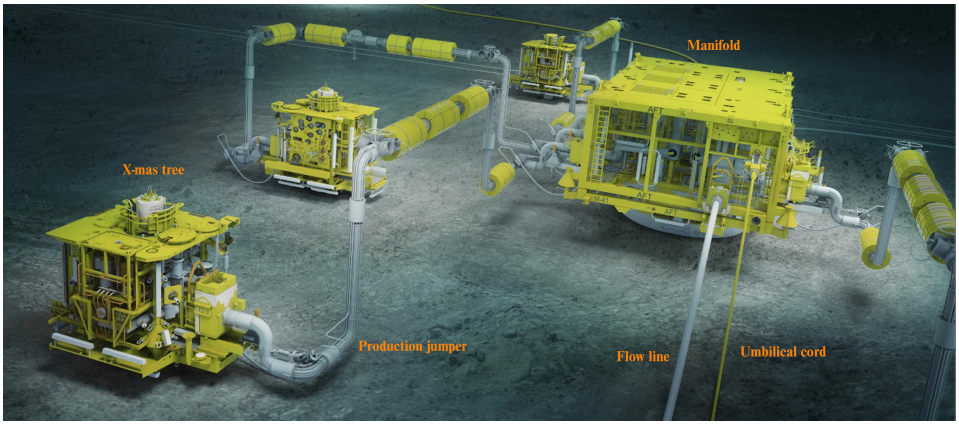


Figure 2.3: Subsea production system [21]

A manifold is a collection of valves, fittings, and pipelines that are used to distribute fluids from multiple wells to a central processing facility [21]. They can be designed with a range of configurations, depending on the specific needs of the production system. These configurations can range from simpler designs where all wells flow into a common header to more complex structures. For example, in a parallel configuration, each well has its own dedicated line feeding into the manifold, effectively isolating the wells from each other. This allows individual control of each well and is particularly beneficial when the wells have different characteristics or production rates. Such configurations depend on factors such as the number and location of production wells, the fluid flow rate, and the specific requirements of the production system. Requirements and regulations can be found in §53 [22].

Flow lines are essentially a network of pipelines that transport fluids from the well to the manifold or processing facility. They are typically made of steel or flexible materials and are designed to withstand the high pressures and corrosive conditions of subsea environments. There is also an umbilical cord, usually located right beside the flow line, which provides power to the subsea control system.

A subsea control system is a critical component of subsea operations, and it faces substantial safety, reliability and environmental challenges [23]. Safety is crucial, and the system must be designed to ensure safety in the event of an emergency or well control issues, according to §53 [22] and §54 [16] provided by the PSA.

2.4.1 X-mas tree

An x-mas tree in a subsea well plays host to numerous safety measures, such as fail-safe features, redundant controls and emergency shutdown systems. The design and structure of an x-mas tree is heavily influenced by the Norwegian legal framework, especially the implementation of independent and fail-safe mechanisms, as highlighted in NOG 070 [11] and NORSOK S-001 [10]. These are used to minimize risks and ensure the well's safe and efficient operation.

The most common type of x-mas tree is the conventional electro-hydraulic with generic architecture provided in NOG 070 [11]. A simplified schematic of an x-mas tree, particularly focusing on its safety system for isolating a single subsea well, is presented in Figure 2.4 [4].

Figure 2.4 depicts three safety valves crucial for assessing the control system's safety. However, it is worth mentioning that there exists other valves on x-mas trees, i.e., valves for chemical injection and gas lift [4]. The safety valves seen in Figure 2.4 are most commonly closed when the Emergency Shutdown (ESD) is initiated, and will be the only focus for this thesis when discussing safety. For more elaboration on ESD, see Appendix A.

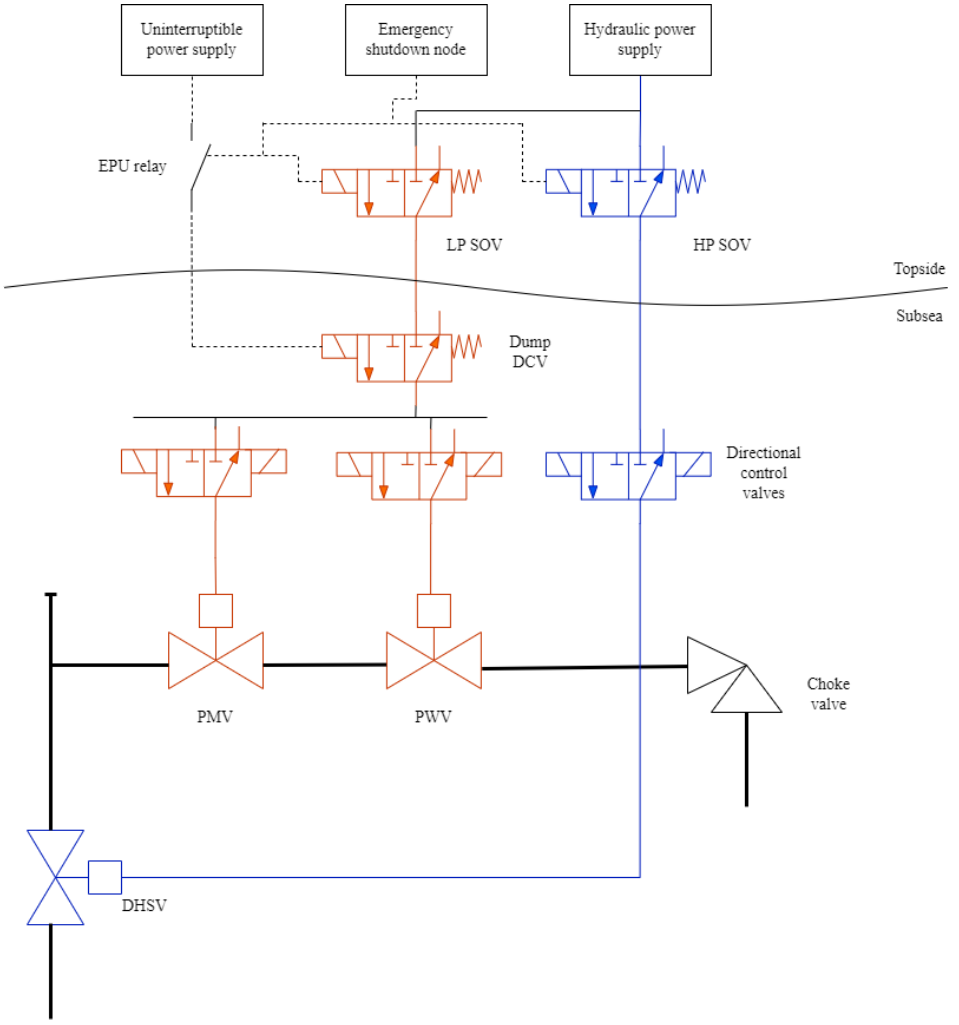


Figure 2.4: Electro-hydraulic x-mas tree with main components for safety and isolation of one subsea well. Adopted from [4]

As introduced, there are various functionality and equipment on an x-mas tree to ensure safety. To mention some equipment also depicted in Figure 2.4, the x-mas tree can protect production in the well with:

- Choke and kill valves that can be used to control the flow of fluids in the well bore. These valves can be used to quickly shut off the flow of fluids in the event of a well control issue, which can help prevent the release of fluids into the environment.
- Emergency shutdown systems (ESD) that can be used to quickly shut down the well in the event of an emergency. This can help prevent accidents and protect the environment in the event of a well control issue.

- Chemical injection systems that can be used to inject chemicals into the well bore to help control the flow of fluids or prevent corrosion in the production system. Note that this is not shown in Figure 2.4, but often marked as CIXT in x-mas tree architectures, and is mentioned for isolation of subsea wells in NOG 070, A13 [11].

As brought to attention earlier, NOG 070 [11] contain a list of basic assumptions regarding isolation of one subsea well for electro-hydraulic trees. These assumptions can be seen as requirements for subsea wells where proposed solutions needs to be as good as, or better than, the recommended guidelines [4].

These pivotal assumptions, slightly paraphrased from NOG 070 [11], include:

- The response time of the system should always be less than the process safety time. This ensures that the system can react swiftly to any changes or incidents, maintaining control over the process.
- ll valves assigned for well isolation and their corresponding DCVs should operate under the fail-safe principle. This means that in the event of a power or control signal loss, these valves should automatically revert to a safe state.
- The ESD logic should be equipped with redundant I/O interfaces and redundant CPUs to avoid single points of failure and increase system reliability.
- The subsea control system should have the ability to actuate x-mas tree valves, but commands for closing valves from ESD can override the commands from the subsea control system. This is to ensure that safety is prioritized over operational commands.

Keeping in mind that it exists in various types to ensure safety, through redundancy and various valve types and configurations, the focus for this thesis will be in the choke valves. Especially focusing on the simplified architecture in Figure 2.4, consisting of the three safety valves, a PMV, PWV and a DHSV. The part of the isolation system marked in orange will be the main focus for the all-electric valve actuation system further on.
Additionally, subsea well tree design and specifications are also provided in ISO 13628-4 [14], as recommended by §54 [16] from the PSA. Different performance and design criteria for the mentioned components in an x-mas tree are covered, for example, required response time for the isolation valves and fail-safe philosophy [4]. Specifically, ISO 13628-6 states that:

> *Subsea control systems shall be designed to render the production system to a fail-safe status upon loss of hydraulic power. Typically, this is achieved by closure of a USV. Such closure can be achieved by either de- energization of electrical circuits or depressurizing of the hydraulic power supply. If an all-electric-type control system is used, the system shall be fail-safe upon loss of electric power.*

The underwater safety valve (USV) mentioned often refers to PMV, PWV or DHSV, located in the x-mas tree.

2.4.2 Electro-hydraulic valves

The three safety valves (PMV, PWV and DHSV) located in an electro-hydraulic tree are used to ensure safe reservoir isolation and shutdown of production. The three valves seen in Figure 2.4 are hydraulic spring-return valves, and need hydraulic fluid pressure to open and to stay open. A safety valve using electro-hydraulic spring actuation has a mechanical spring which is compressed whenever the valve is in normal operation [24]. When the hydraulic pressure disappears or is lost, the valves close. This is referred to as a "De-energize to safe"-principle, because it returns to a fail-safe state when the hydraulic pressure is lost. The hydraulic fluid which causes the pressure disappears when the fluid from valve cylinders are drained by the control module.

Safety valves are used to create a barrier to the reservoir [4]. The first barrier element is the production master valve (PMV) and the production wing valve (PWV). The PMV and PWV prevents hydrocarbon release in any case except the case of destruction of the wellhead. A simplified schematic of valve isolation can be seen in Figure 2.5. The PMV is typically located at the top of the well, as seen in Figure 2.5 which is a sketch from NOG 070 [11]. The PMV is used to isolate the well from the subsea production system, which allows the well to be shut in for maintenance as well as for safety, without affecting the overall production.

The PWV is used in conjunction with the PMV and is located further down the well. It can control the flow from the well bore into the production system. The PWV can be closed independently of the PMV, which allows for more precise control over the flow.

A crucial feature of subsea valves is their ability to operate efficiently over extended periods without requiring any maintenance. This is a result of the valves being hard to reach when they are located at water depths between 1 and 3 km. The hard-to-reach locations are also the reason for the use of fewer subsea valves in comparison to topside valves, to reduce potential failure points and minimize the need for intervention [8]. Nevertheless, there exists a variety of subsea valves other than choke PMV and choke PWV. For on and off purposes on manifolds, Ball and TCG valves are widely used. There are also choke valves for flow regulations, and small check-valves used to prevent back flow in injection lines. Having Figure 2.4 as the base for the valve system, the primary focus will be on DHSV, PMV, PWV and a choke valve, since they are key safety valves used during an ESD. For a more comprehensive description of ESD, the reader is referred to Appendix A.

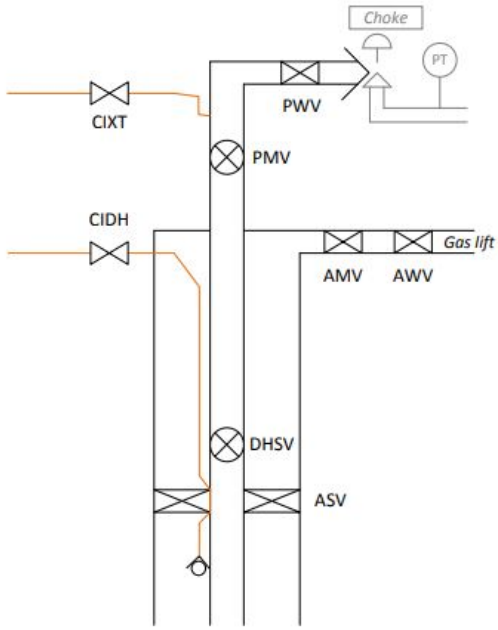


Figure 2.5: Simplified well isolation [11]

2.5 All-electric valve actuation

While spring actuation is common in current electro-hydraulic x-mas trees, there are new possibilities for the all-electric x-mas tree. A possibility is to use electric actuation instead of spring-return closing, which will require the system to rely on at least a battery equipped with a BMS and a motor supplemented with a motor controller to control the valve stem position. Both of these components, BMS and motor controller, depends on software. However, traditional safety functions are not software dependent, which imposes a problem for the new proposed all-electric safety system. Part 3 of IEC61508, IEC 61508-3 [25] sets requirements on software and creates guidelines for compliant software. An important aspect to retract from this is the fact that the software should be proven-in-use, requiring the software to be previously used under similar conditions, and that it can perform reliably. Additionally, IEC 61508-3 provides specific requirements applicable to support tools used to develop and configure a safety-related system within the scope of IEC 61508-1 [20] and IEC 61508-2 [26].

To be able to prove the software in use, a digital twin, in cooperation with SUBPRO, is being developed for the system. A digital twin is a precise virtual copy of a machine or a system, used to stipulate the expected behavior of the physical entity [1]. An overview of the total all-electric actuation system which is in development can be seen in Figure 2.6. The digital twin is seen as the red boxes in Figure 2.6, and it is mainly the mechanical

components in the valve actuation. It is used to test out if the software from the motor controller, seen as the purple box, as well as the BMS seen in the green box, works as intended.

Replacing the hydraulic components for the electro-hydraulic valve actuation with electric equivalents will result in an electric valve actuation in the x-mas tree as desired. This means that the hydraulic power supply no longer is present as in Figure 2.4 for the orange valves. In addition to exchanging the power supply of the valves, a control system is essential for correct functionality of the x-mas tree and the belonging safety valves. The control system will perform functional testing, execute lower level ESDs and execute diagnostic functions [4]. However, it is worth mentioning that electric valve actuation is no novel technology, but it is novel for the use of electric actuation of safety valves in subsea and offshore installations.

A typical sketch for an architecture required for isolating one subsea well using an electric control system can be seen in [4] as Figure 17.2, depicted again in this chapter as Figure 2.7. This is essentially the same as Figure 2.6, except for Figure 2.6 being slightly more detailed, and there are different definitions of the same components. The BMS and battery is swapped out with “Energy Storage”, and the motor controller is replaced with “Motor Driver”. The box for mechanical components are directly mapped to the digital twin in the system, depicted as the red boxes in Figure 2.6. To recall, a digital twin is a digital model of a physical entity.

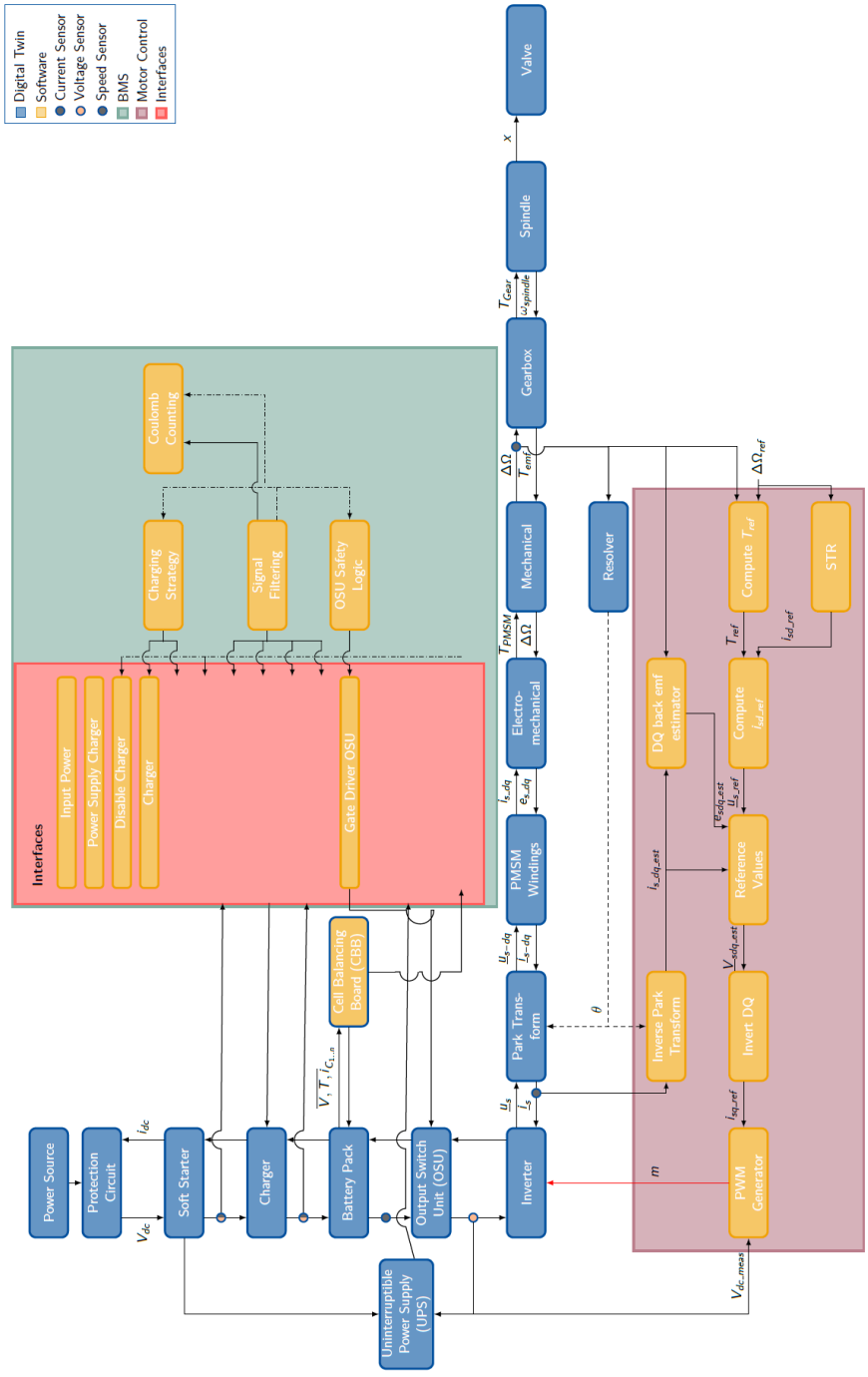


Figure 2.6: Overview of DT by Ludvig Björklund.

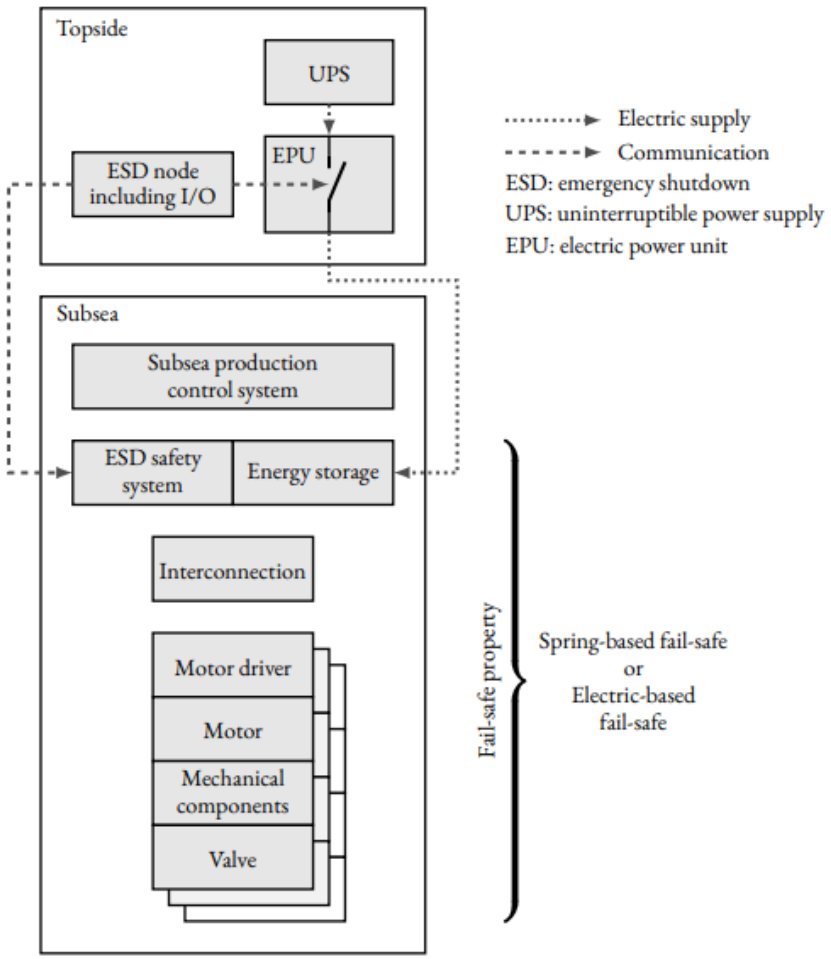


Figure 2.7: Sketch for isolating one subsea well [4]

There exists a range of generic architectures for a subsea all-electric valve actuation system, and they are usually depicted with functional blocks and functional elements instead of using components. These functional blocks denote key functionalities that can be achieved through a range of alternative methods or components. The architecture can thus vary substantially, typically influenced by the approach of each supplier.

Nevertheless, a design of an all electric control system has been proposed by Imle et al. [27] and a tree actuation influenced by the proposed actuation can be seen in Figure 2.8. The intended tree actuation consists of the valves, the mechanical actuators and the subsea control module (SCM). The SCM can control the DHSV, PMV and PWV [4], whereas The Subsea Electronic Module (SEM) consists of the actuation and motor controller, the

battery and a BMS, which is the electronics used to control the valves. The BMS has the role of taking care of the power supply in the system by monitoring charge level and state of health of the battery. It is modelled digitally as the light green functional block in Figure 2.6. The SCM has also added redundancy by the use of both SEM A and SEM B. SEM A and SEM B are essentially the same, and only one of them are needed for correct valve operations.

The reason for added redundancy is to be able to perform system maintenance on one of the SEMs while the other does the required work, such that the downtime is minimized. Redundancy is also used to ensure that safety functionality is performed even though one of the SEMs fail. This is included as a basic assumption from NOG 070 [11], applicable for the electric x-mas tree as well.

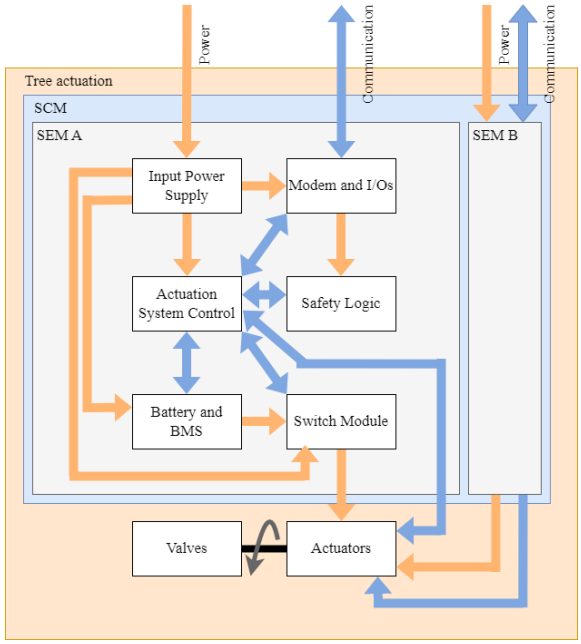


Figure 2.8: Proposed all-electric tree design [1]

All the elements of the system seen in Figure 2.8 are dependent on power supply, seen as the orange arrow. When the Input Power Supply from topside, marked as a power line coming from outside the SCM, is not available, the BMS will use the Switch Module to apply the Battery as the new power source. The battery then acts as an uninterruptible power supply in case the power link from topside is interrupted [24]. The Safety Logic block, further referred to as the safety controller, is responsible for the safety functionality of the system being correctly executed. Common safety functions from the Safety Logic can be closing the valve or opening the valve. The safety commands from the safety controller are communicated with both topside, the Actuation Control system and the BMS, and is visualized using blue arrows. Hence, a proper distinction of power signals and command

signals by the use of orange and blue arrows.

2.5.1 The Battery Management System

The BMS, represented by the green box in Figure 2.6, is an integral component of ensuring that the battery acts as a dependable reserve power source in the all-electric valve actuation system. It is paramount to guarantee optimal functionality of the battery [4]. Its critical role extends to enhancing the battery's reliability, safety, and overall performance by thorough monitoring and state estimation of the battery [28].

As earlier mentioned, the BMS has the role of monitoring the state-of-charge (SOC) and state-of-health (SOH) of the battery. The SOC provides information on the amount of charge available in the battery, while the SOH assesses the battery's aging level and current condition [28]. Knowledge about the SOH of the battery leads to an improvement of reliability and a reduction of risk and maintenance efforts in safety-critical systems, because it will detect when the battery is close to failing and enable predictive maintenance [29]. As well as the SOC and SOH, the BMS might include state-of-function (SOF) estimation as well, which is estimated according to the SOH and SOC [30]. SOF is used to describe how the battery performance meets the real commands while the battery is employed. The details of how the SOH, SOC and SOF algorithms are constructed are not important further for this thesis, and therefore not taken into account.

It is important to recognize that the BMS needs the voltage, current, and temperatures from the cells in the battery in order to properly control it. In addition to SOH, SOC and SOF estimation, BMS also takes care of cell balancing, charge management and thermal management [31]. Cell balancing is used to correct discrepancies in the cells in a battery pack such that the SOC is the same over all the cells, whereas charge management controls how the battery should charge to maximize its lifespan and ensure safety. In addition to charge, heat can also minimize the battery's lifespan, so thermal management is important for maintaining the battery within safe operating temperature range. The entire BMS estimation algorithm framework can be visualized in Figure 2.9.

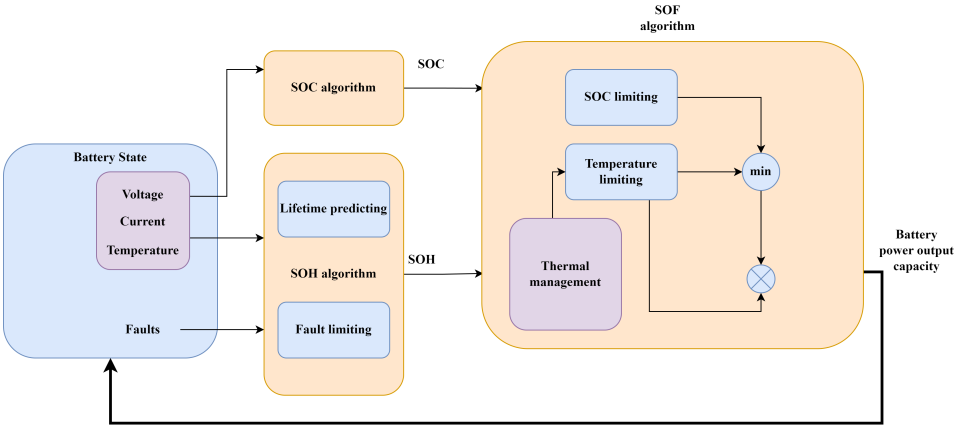


Figure 2.9: BMS state estimation framework

A common choice of battery for daily application is the Lithium-ion (Li-ion) battery. and they have gained significant attention due to their application in electric vehicles [31]. Li-ion batteries are advantageous because of their high density and low weight. However, the use and focus of these batteries has been a low lifetime need, whereas for subsea implementation, consequences of aging and little to none maintenance have to be considered [32].

Li-ion battery operation can be impacted by various factors, such as overheating, overcharging, overdischarging, thermal runaway and accelerated degradation [31]. These factors can result in increased temperature and pressure within the battery, posing a risk of explosion. The consequences can be mitigated by the BMS through monitoring and controlling the involved parameters such as temperature, voltage, and current.

The main tasks of the BMS can be recognized as [30]:

- Protect the cells and battery packs from being damaged
- To make the batteries operate within the proper voltage and temperature interval and guarantee the safety as long as possible
- Maintain the batteries to operate in a state such that the batteries fulfills the systems' requirements

Hence, the use of a battery equipped with a BMS serves as a reliable backup power source for the all-electric valve actuation. The BMS, depicted as the green box in Figure 2.6, plays a crucial role in ensuring optimal performance, durability, safety, and reliability of the battery [4]. The BMS monitors the state-of-charge (SOC), state-of-health (SOH), and state-of-function (SOF) of the battery, providing valuable information on the amount of charge available, battery aging level, and current condition. This knowledge leads to improved reliability, reduced risk, and minimized maintenance efforts in safety-critical systems [29], [30].

2.5.2 Proposed motor and motor controller

Permanent magnet synchronous motors, known as PMSM, are typically used for high-performance and high-efficiency motor drives, and are widely used for industrial and transport application [33]. PMSM has great advantages and interesting characteristics, such as high efficiency factor, low rotor inertia yet a high torque-to-inertia ratio, high reliability, and significant life expectancy which yields practically no need for service during the usage period [34]. Hence, the PMSM depicted in Figure 2.6 has been selected as the motor of choice.

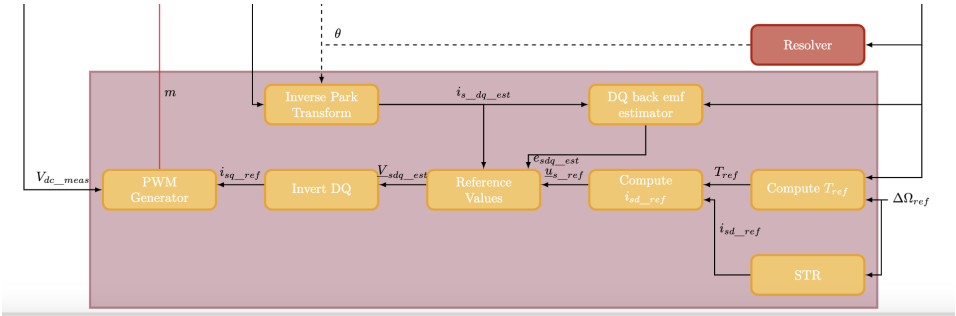


Figure 2.10: Snippet of Figure 2.6 concerning the motor controller

The motor is controlled through the motor controller, depicted as the purple box seen in Figure 2.6, depicted here as Figure 2.10. The motor controller uses Field Oriented Control with Space Vector Pulse Width Modulation to control the motor. The output from the motor controller is a pulse signal with the desired frequency and voltage for the motor to operate on, where the Pulse Width Modulation pulses are generated from the controller from the inverter. This is done in "PWM Generator" seen in Figure 2.10.

Further details on the functioning of the module are provided in Appendix C to strengthen the total understanding of the system. Insight as to how the motor is controlled using Field Oriented Control, Pulse Width modulation and Park/Clarke transformations as seen in Figure 2.10 are described. However, since the details about the motor model is not relevant for the safety analysis to be conducted further, it is not presented in detail here, and the reader is recommended to read Appendix C. The most important take-away is that the motor controller controls the starting and stopping of the motor.

3

System-Theoretic Process Analysis

This chapter presents the theory and method behind the System-Theoretic Process Analysis, and builds on the theory presented in the specialization project "Using Safety Analysis to generate test scenarios of an All-Electric Control System in Subsea Wells" from autumn 2022 [1].

3.1 Background

Traditional hazard analysis, such as HAZOP, Fault-Tree Analysis (FTA) and FMECA are known safety analysis methods used in a variety of industry today. However, these methods were developed and designed almost 50 years ago, when systems were less complex and consisted of more analog components, and not as today's software intensive systems [35]. With the new increasing complexity, these traditional methods may have limitations in effectively capturing and analyzing the hazards associated with them. Hence, another methodology for safety analysis is developed, such as System-Theoretic Process Analysis (STPA). STPA is a safety analysis technique that uses system theory as a basis, known for its ability to identify underlying system vulnerabilities, such as controller and software failure.

By using system theory, the system is treated as a whole and not as the sum of its parts. One of the advantages of using system theory in safety analysis is its ability to capture emergent properties that arise from the interactions between system components [1]. Conversely, conventional hazard analysis relies on decomposition of the system, such that complex systems becomes smaller and more manageable to analyze instead. However, a drawback with this decomposition is that it is based on the assumption that the system is not distorted by separating it into smaller parts, and that there are no phenomena which are no longer captured in the subsystems. This may not always be the case, leading to potential misinterpretations and oversight in the analysis. Nonetheless, since a method like STPA evaluates the system as a whole, it identifies problems such as troublesome interactions, design errors, human error or other interaction errors [36]. A key aspect of the STPA ap-

proach is its acknowledgement that errors or hazards in software-intensive systems often originate from controller errors. Importantly, by evaluating the total system, one can predict characteristics and system behavior during the design phase and not necessarily wait until the process is in operation.

Another advantage of STPA is, as introduced, its ability to identify underlying system vulnerabilities. Instead of focusing on component failures, STPA emphasizes the identification of control actions that could lead to hazards [37]. This approach helps to identify potential failure modes early in the system development, allowing for effective mitigation measures to be implemented. Furthermore, STPA is an adaptable method that can be applied to different domains and industries, as well as easily modified.

The STPA method can be summarized in four steps [37]:

1. **Define the purpose of the analysis.** What losses are to be prevented? What system is to be analyzed? What is the system boundaries?
2. **Model the control structure.** The control structure is a model of the system which captures function relationships and interactions by a set of feedback control loops.
3. **Identify unsafe control actions.** How can the control actions lead to losses?
4. **Identify loss scenarios.** What is the reason that the losses occur?

The steps are visualized in Figure 3.1, and will be elaborated further.

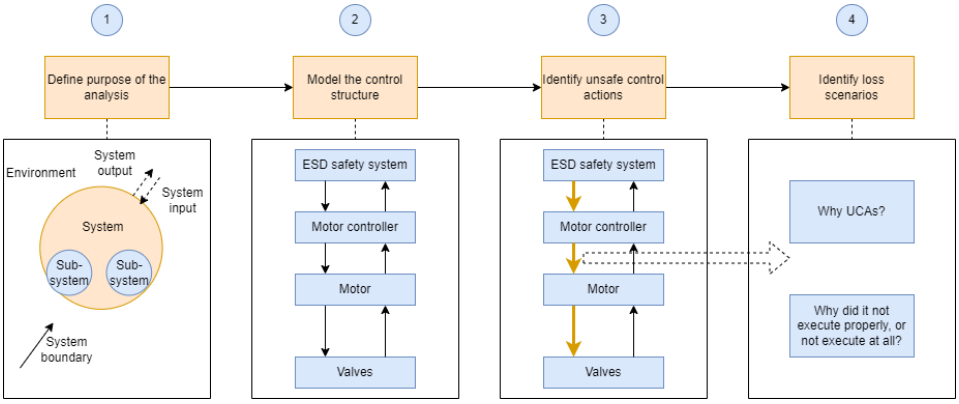


Figure 3.1: Overview of the basic STPA method [1]

3.2 Purpose of analysis

Defining the purpose of the analysis is the first part of a STPA, and consists of four subroutines: Identify losses, identify system-level hazards, identify system-level constraints and eventually refine hazards [37]:

- **Identify losses:**

The primary objective of the STPA is to reduce the scope of or prevent losses, which necessitates the identification of possible losses. A loss can be defined in various ways: it may refer to the inability to sustain something, the lack of a specific function, an instance of misplacing or losing someone or something, or a reduction in quantity, value, or size [38]. This can also be applicable to systems which are about to be analyzed, and is defined in the STPA handbook [37] as:

> *A loss involves something of value to stakeholders. Losses may include loss of human life or human injury, property damage, environmental pollution, loss of mission, loss of reputation, loss, or leak of sensitive information, or any other loss that is unacceptable to the stakeholders.*

- **Identify system-level hazards:**

Once the losses are identified in the previous step, the next step in the STPA process is to identify the hazards that are related to these losses. Hazards can be defined as situations or sets of circumstances that have the potential to lead to a loss when the worst-case scenarios occur [37]. A key consideration during this stage is to not only focus on the obvious hazards, but to also delve into complexities of the system that could potentially give rise to hidden, unanticipated hazards. The functionality of each system component, as well as interactions with other system components, needs to be considered.

In order to determine what hazards are present in the system, it is necessary to define the system boundaries. Figure 3.2 illustrates the relationship between the system, the surrounding environment, and the system boundaries. The system boundary separates the system from its environment. This means that it outlines the scope of the system that is subject to analysis, helping to clearly identify what is considered during the analysis.

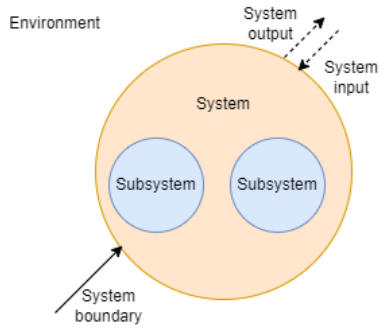


Figure 3.2: System, system boundaries and environmental relationships [1]

A system-hazard is closely related to a loss, and should be traced to the related loss identified earlier in the analysis.

- **Identify system-level constraints:**

System-level constraints define necessary system conditions or behaviors that aid in the prevention of hazards and losses [37]. Such constraints can also define how the system should behave to minimize loss if hazards occurs.

In the STPA process, each system-level constraint can be mapped to one or more system hazard that were identified in the previous step. This mapping helps to establish a clear connection between the constraints and the hazards, enabling a systematic approach to addressing potential risks. Furthermore, each hazard identified in the system can be traced back to one or more losses. This traceability helps to understand the potential consequences of hazards in terms of losses and their impacts on the system and its environment [1].

System-level constraints can be written as such:

$$< \textit{System-level constraint} > =< \textit{System} > + < \textit{Condition to enforce} > + < \textit{Link to Hazards} >$$

or

$$< \textit{System-level constraint} > = \text{If} < \textit{hazard} > \text{occurs, then} < \textit{what needs to be done to prevent or minimize a loss} > + < \textit{Link to Hazards} >$$

System-level constraints serve as a means to guide the system design and operation, ensuring that the system behaves in a manner that prevents hazards and minimizes the potential for losses. These constraints can include requirements related to system performance, functionality, reliability, safety, security, and other relevant aspects. They are derived from a thorough understanding of the system's purpose, intended use, and the context in which it operates.

- **Refine hazards:**

This step, although optional, tends to be necessary for large and complex systems [37]. This is because the refining of hazard can be used as a guidance for modelling the control structure, since the purpose of refining hazards is to create sub-hazards, which can subsequently lead to more specific constraints for the system.

This highlights the importance of incorporating an iterative approach in the steps of the analysis. It's crucial to remember that the process doesn't necessarily have to strictly follow a step-by-step order. Depending on the specific requirements and the complexity of the system being analyzed, there might be a need to revisit and refine certain stages of the process for a more thorough and comprehensive analysis. The most important aspect of this analysis is that everything should be traceable.

3.3 Model the control structure

The next part of the STPA is to model a hierarchical control structure of the system.

A hierarchical control structure consists of a system model containing feedback control loops, where the controller and the controlled process with the highest executive authority is placed on top [37]. Each entity has the authority to control the entity immediately below

it. In addition to this, control actions, feedback and other inputs to and outputs from other components which are neither control nor feedback are included in the model.

A model structure can also include sensors and actuators, and is usually added in further steps of an analysis. Such details will be clearer during refinement and further detail adding at the end of the analysis [1].

The controllers can be categorized into two parts: the control algorithm and a process model [37]. The control algorithm is responsible for deciding the control actions to be executed by the controller. On the other hand, the process model represents the controller's perception and understanding of the system and its environment, and it provides awareness of the components involved in the process being controlled [36]. A process model which is inconsistent with reality can lead to unsafe control actions, and are updated continuously by feedback loops in the system. In other words, it can be perceived as the information received by the controller regarding the system's states through sensor data and any predictive analysis that may exist. An example of an inconsistent process model from the STPA handbook [37] is when an aircraft "believes" that it is in a state where it is descending even though it is ascending, and control actions will be executed accordingly.

Hence, a controller's process model is determined by a controller's beliefs about current states, previous states, previous actions, predictions, other controllers in the system and beliefs about the system currently being controlled. The development of the functional model structure in this step should aid in identification of control actions done in the next step [39].

3.4 Identify unsafe control actions

After the control structure is modelled, the next step is to identify the unsafe control actions, also known as UCAs, of the controllers. An UCA is in the STPA handbook [37] defined as: *"[...] a control action that, in a particular context and worst-case environment, will lead to a hazard."*

The hazards in this context are usually the ones identified in the first step of the analysis.

Control actions are considered unsafe or hazardous when [36]:

- the action required for safety is not applied
- the action is provided when it is unsafe to do so
- the action is done too early or too late
- a continuous control action is applied for too long or stopped too soon

This means that there are four ways that a control action can be unsafe. In addition to this, every UCA must be traceable to a system-level hazard. If there is an UCA which is not related to a system-level hazard, a refinement of hazards is usually the solution. It is worth noting that STPA does not follow a strict linear fashion, allowing for flexibility in the analysis process, as mentioned in the last part of the first step.

In the same manner, UCAs can also occur from human behavior. Such UCAs can stem from miscommunication between a crew, lack of training beforehand, or stress.

The total part of the identification of UCAs for the STPA can be concluded to three parts:

- Identification of general UCAs: When are the control actions unsafe or hazardous?
- Identification of human UCAs: What unsafe control actions can stem from human behavior?
- Defining controller constraints: What should be done in the controller to prevent unsafe control actions?

Defining controller constraints

Controller constraints are defined to specify the controllers' behavior that should be satisfied in order to prevent UCAs [37]. This can be done by translating the UCAs found earlier to constraints for each controller.

3.5 Identify loss scenarios

Furthermore, in the process of conducting an STPA, it is necessary to identify loss scenarios that could lead to UCAs. A loss scenario is a situation or an event which can lead to an UCA or a hazard.

During this stage of the analysis, additional design information may be incorporated, such as information about the controller process model and control inputs [36]. Incorporating additional information about the process and the design helps to identify potential scenarios that closely resemble the actual process design. Potential design implications can then be assessed related to the losses and UCAs. The scenarios which are found in this part of the analysis form the basis of the possible test scenarios which can be created and used to verify the system behavior. Hence, an important part of this analysis is to create a context in a scenario and not list individual factors.

There are two main types of loss scenarios that must be considered. The first type involves scenarios which causes UCAs to arise, whereas the second type identifies reasons for why control actions would be improperly executed or not executed at all, which could ultimately result in hazards.

The first type of loss scenarios, i.e., scenarios that could lead to UCAs are typically attributed to failures related to the controller, a flawed control algorithm, an inadequate process model, or unsafe control inputs. Controller failures may include issues with the hardware or insufficient power supply, while flawed control algorithms may result from incorrect internal decision-making within the algorithm. A control algorithm is designed to specify the appropriate control action to be performed based on the controller's process model, inputs, outputs, and environmental factors.

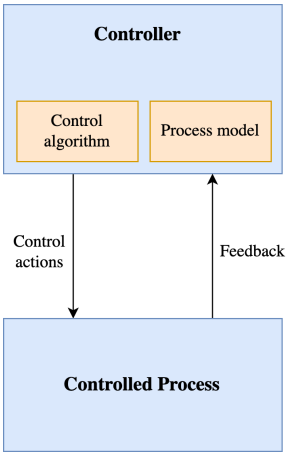


Figure 3.3: Control loop [1]

An inadequate process model may cause the algorithm decision-making to deviate from reality, which can occur due to wrongful feedback information, incorrect interpretation of feedback, or loss of information. This causes the process model to deviate from reality. The first type of loss scenario is related to the controller in a generic control loop, seen in Figure 3.3.

The second type of loss scenarios result from control actions that are not executed or are executed improperly. To identify and create such scenarios, it is necessary to consider the factors that affect the control path as well as those that affect the controlled process [37]. The control path is the path that transfers the control actions to the controlled process, and is usually modelled as an arrow pointing to the controlled process seen in Figure 3.3. A control path can consist of actuators, switches, routers and other equipment, and problems along this path can occur, which can cause control actions to be improperly executed or not executed at all.

On the other hand, loss scenarios can also occur due to control actions not being executed due to missing process inputs, disturbances, delays or conflicting commands from other controllers. This can be recognized as loss scenarios related to the controlled process.

All the possible ways UCAs can occur are summarized in Figure 3.4.

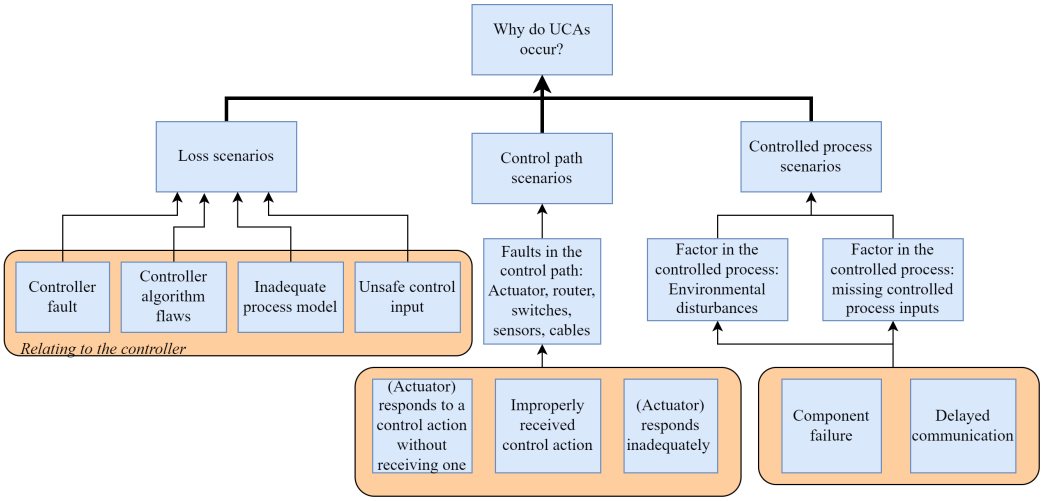


Figure 3.4: Summary of different ways UCAs can occur [1]

Recognizing and evaluating loss scenarios using STPA is essential for gaining insight into potential failures or weaknesses in control systems. This understanding is crucial for creating effective measures to mitigate unwanted consequences, such as UCAs and hazards. By thoroughly examining and addressing loss scenarios through STPA, organizations can improve the safety and reliability of their systems and prevent such events from occurring.

4

Formal Verification

This chapter delves into formal verification, highlighting its strengths and limitations in the use of software verification. Event-B, the chosen formal language for verification in this thesis, will be comprehensively explained. Furthermore, the SE-STPA method, which integrates formal verification into safety analysis, will be introduced and discussed in detail. Recognizing the limitations of SE-STPA, this chapter presents SE-STPA-mod, a novel safety analysis method specifically designed for the safety analysis to be conducted in this thesis.

4.1 The use of formal methods

Advances in safety critical software systems has introduced new awareness of software quality and an expansion in the definition of safety critical software. The quality of such software is essential for the correctness, safety, and security of a system. Research indicates that inadequate development in system designs is a prominent cause of system failures [40]. To guarantee that a system meet critical specifications, the use of formal methods coupled with formal verification techniques is beneficial, as the traditional development methodologies, such as simulation and testing, are insufficient to guarantee the software quality of safety software.

By formalizing specification, the ability to perform a more rigorous analysis and transform effort from the test phase to the specification phase improves the quality of the system [2]. Formal methods are also described in IEC 61508-7 [41] as follows:

> *Formal methods transfers the principles of mathematical reasoning to the specification and implementation of technical systems therefore increase the completeness, consistency or correctness of a specification or implementation.*

Mathematical reasoning, which involves using logical and analytical thinking based on mathematical principles and concepts to create a model applied to our real-world model,

is often said to be too complex [42]. This is because it often requires abstract concepts and symbolic language, and it also requires looking at design of systems in a fundamentally different way as in usual development [43]. Despite this complexity, it remains an invaluable tool in understanding and describing the workings of automated systems, because it yields valuable results such as increased productivity in testing, and it could identify tricky, probably hidden, bugs.

The mathematical model created from the mathematical reasoning consists of a formal language with mathematically defined syntax and semantics to provide a precise formal model. A precise specification contributes to eliminate any ambiguities and remove obvious defects by disallowing different interpretations [44]. By developing a formal model based on a desired language for either hardware, sequential systems or concurrent systems, an increased understanding of a system is achieved. The increased understanding will reveal inconsistencies, ambiguities, and incompleteness that might otherwise go undetected [45].

Nevertheless, there is no "best method" regarding formal methods, and the choice of suitable method is based on the understanding of the system which is to be verified [41]. Choosing a formal method is also not always straight-forward, because there exists a range of formal method vendors [42]. For example, there exists a language called CCS which is tailored to concurrent systems, Higher-order-logic (HOL) for hardware specifications and OBJ for system specifications with user feed-back and system validation [25]. Since Event-B is most commonly used in a version of STPA called SE-STPA, it is chosen as the preferred formal language for this thesis.

4.2 Event-B

Event-B is a method for formalizing and developing systems which has evolved from the Classical B language [46]. The Classical B language were developed around 1970, and are closer to traditional mathematical notation which facilitates for specifying data types, variables, constants, sets, functions, and predicates, and it allows for the definition of operations on these constructs. This is all based on set theory and first-order logic. However, there is no support for events and concurrent behavior for a system, which is where Event-B serves as an extension to the Classical B language. It was first introduced in the 80s as a way to correctly design safe software, especially in Railway systems [47]. Later on, Event-B was used in the late 90s as an extension to B by adding a way to study and specify not only software, but whole systems, such that systems which contain software, hardware and other equipment could be modelled – and not only software or only hardware as previously. Hence, Event-B provides a way to formally model, specify, and reason about the behavior of whole systems

A model is the primary concept of doing formal developments, especially in Event-B, and the Event-B modelling language consists of two main constructs which are used to model a system: context and machines [42]. Contexts are used to formalize the static parts of the system, whereas machines are used to specify dynamic behavior of the system.

Event-B allows models to gradually be developed through context extension and machine refinement, to enable development of target systems from abstract specifications and later to add more details. This allows for maintained properties, because if an abstract property is proven, it is guaranteed to be correct even when it is refined subsequently. This concept can be seen in Figure 4.1.

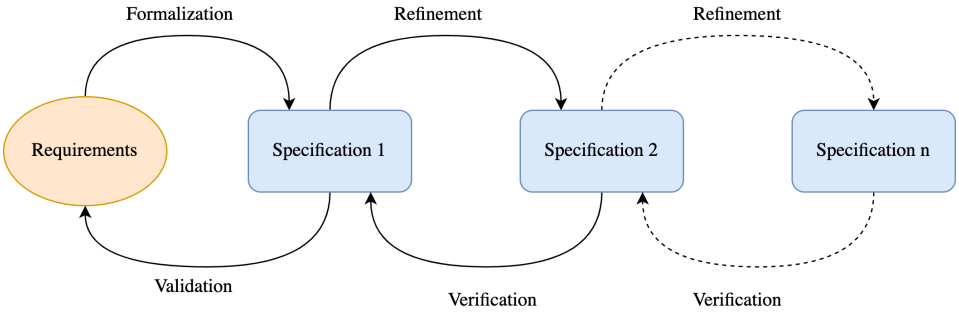


Figure 4.1: Concept of formal methods and refinement

Event-B puts emphasis on gradual construction of the model through refinement, and finally verification of, in particular, safety properties through formal proofs [48]. The theorems which are to be proved are automatically generated and then proved by the use of The Rodin Platform [49]. The Rodin platform is an open source IDE for Event-B that provides support for refinement and mathematical proofs.

The basis of the Event-B language is first order logic, or more specifically predicate logic, and mathematical theory of sets and relations [40]. Predicate logic is based on predicates (or statements), which is an expression of one or more variables which can be quantified [2]. In other words, a predicate is an expression where the value is either true or false. The main concepts of the language will be elaborated further to gain an understanding on how to make Event-B applicable to the desired system.

4.2.1 Contexts and machines

The two main components used in Event-B are, as introduced, contexts and machines. Contexts consists of carrier sets (s), constants (c) axioms and theorems, whereas a machine consists of variables, invariants, events, and theorems [40]. A context can be seen as a module which defines a user-defined set which are used in formal specification. A carrier set in a context is defined as the underlying set of an algebraic structure, the constants are defined using a set of predicates, and the set of axioms can describe properties of the carrier sets and the constants. An example of a context from the Rodin handbook for a traffic light controller, where a new set of the traffic light colors are made, can be seen in Figure 4.2.

```
CONTEXT
  ctx_1
SETS
  COLOURS
CONSTANTS
  red
  yellow
  green
AXIOMS
  axm1   :   partition(COLOURS, {red}, {yellow}, {green})
END
```

Figure 4.2: Traffic light context

The axiom seen in Figure 4.2 for partition indicates that red $\neq$ green $\neq$ yellow, which is a common notation in Rodin and Event-B to ensure that the constants are not equal. The theorems which are also included in a context are derived properties of carrier sets and constants [46].

A machine describes the dynamic properties of the system, and uses specified contexts to describe system properties. The contexts are usually used to define variables used in the machine. Invariants can be seen as requirements for the system, and are an essential feature of an Event-B machine [2]. They show the properties that must hold in every reachable states of the machine. An invariant property is proved by induction:
1. Prove that the property is established by the initialization
2. Prove that the property is maintained when some variables are modified by an event

On the other hand, variables (v) define the state of a machine and are constrained by invariants $I(v)$ [46].

An initial machine also related to the traffic light controller can be seen in Figure 4.3.

```
MACHINE
  mac
VARIABLES
  cars_go
  peds_go
INVARIANTS
  inv1   :   cars_go ∈ BOOL
  inv2   :   peds_go ∈ BOOL
  inv3   :   ¬ (cars_go = TRUE ∧ peds_go = TRUE)
EVENTS
  INITIALISATION   ≙
  STATUS
   ordinary
  BEGIN
   act1   :   cars_go ≔ FALSE
   act2   :   peds_go ≔ FALSE
  END
```

Figure 4.3: First traffic light machine

The machine called **mac** introduces the variables **cars_go** and **peds_go**, serving as indicators for allowing cars and pedestrians to move, respectively. These variables are initially set to FALSE, and the traffic can be seen as standing still. The initial machine configuration seen in Figure 4.3 is designed to be the most basic form that ensures safe passage for both cars and pedestrians. This safety is achieved through the utilization of invariants, which has to hold for every event in the machine. The most important requirement to ensure safety is preventing the simultaneous movement of cars and pedestrians, as stipulated by the invariant **inv3**.

In order for a machine M to have information of context C, it must connect to C. Connection means that M *sees* C, such that M has access to all carrier sets and constants. This is also called a "Sees Context Relationship" [2]. A machine can see several contexts, and it can see the extensions of the contexts.

The relationship between a context C and a machine M can be seen in Figure 4.4.

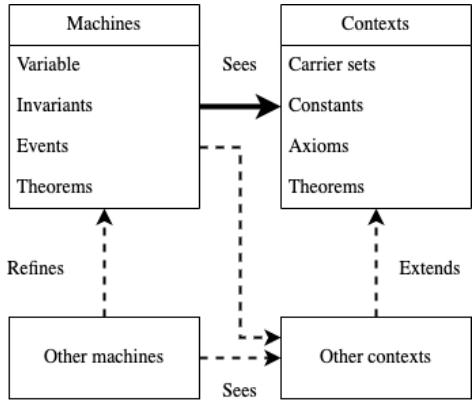


Figure 4.4: Relationship between machines and contexts

In order for **mac** to use the colors from **ctx_1** in Figure 4.2, **mac** needs to *see* **ctx_1**. The context relationship can be seen in Figure 4.5.

```
MACHINE
  mac1
REFINES
  mac
SEES
  ctx1
VARIABLES
  peds_color
  cars_colors
INVARIANTS
  inv1   :   peds_color ∈ {red,green}
  gluing_peds   :   peds_go = TRUE ⇔ peds_color = green
  inv2   :   cars_colors ⊆ COLORS
  gluing_cars   :   cars_go = TRUE ⇔ green ∈ cars_colors
EVENTS
  INITIALISATION   ≙
  STATUS
   ordinary
  BEGIN
   act3   :   peds_color ≔ red
   act4   :   cars_colors ≔ {red}
  END
```

Figure 4.5: Machine with context relationship

mac1 is now a refined machine **mac**. More on refinement later in Section 4.2.4. Nevertheless, to relate the previous invariants to the new invariants which includes colors, *gluing invariants* are used [2]. The variables now have new names, **peds_color** and **cars_colors**, where **peds_color** is deterministic with only two values, whereas **cars_colors** is a set of all colors in **ctx_1**.

The most general machine structure and context structure can be seen in Figure 4.6 and Figure 4.7.

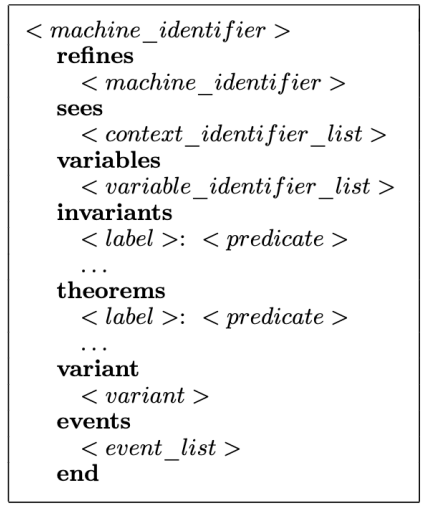


Figure 4.6: Machine structure [42]

- **refines** yields the possibility of the machine to refine another.
- the **sees** section makes it possible to use a context's sets, constants, and axioms
- **variables** constitutes the states of the machine
- **invariants** are predicates that should be true for each reachable state. The type of each variable is declared in this section
- **variants** are specified to support proofs for termination
- **events** can assign new values to variables

$< context\_identifier >$
extends
$< context\_identifier\_list >$
sets
$< set\_identifier\_list >$
constants
$< constant\_identifier\_list >$
axioms
$< label >:\ < predicate >$
...
theorems
$< label >:\ < predicate >$
...
end

Figure 4.7: Context structure [42]

- **extends** are used to extend the existing context. More on extension in Section 4.2.3
- **sets** defines used-defined types
- **axioms** define rules that will always be the case for the given elements
- axioms can be marked as **theorems** to prove a predicate

4.2.2 Events

In Event-B, an event is a fundamental concept used to describe a change or transition in the system state [42]. It is a unit of behavior that captures a single action or a group of actions which are to be executed by the system. An event e can be represented as:

$$e = \text{any } x \text{ where } G(x, v) \text{ then } Q(x, v) \textbf{ end} \tag{4.1}$$

where

- x: the event's parameters
- $G(x, v)$: the guard
- $Q(x, v)$: the action

The guard states the necessary condition under which an event may occur [46]. The actions specify the changes that occur in the system states when the event is executed. The guards and actions must be consistent with the invariants defined for the system. Events can also be written in short form, or not contain any parameters or guards:

$$\text{short form: } e = \text{when } G(x, v) \text{ then } Q(x, v) \textbf{ end} \tag{4.2}$$

$$\text{no parameter: } e = \text{begin } Q(x, v) \textbf{ end} \tag{4.3}$$

$$\tag{4.4}$$

```
set_cars   ≙
STATUS
 ordinary
ANY
 new_value
WHERE
 grd1   :   new_value ∈ BOOL
 grd2   :   new_value = TRUE ⇒ peds_go = FALSE
THEN
 act1   :   cars_go ≔ new_value
END
```

Figure 4.8: Event

An event action can be either deterministic or non-deterministic [42]. An action is non-deterministic if two different copies of the action may behave differently given the same input [50]. If the action is deterministic, it will always behave the same way when given the same sequence of inputs. Deterministic actions are made of a variable identifier, followed by :=, followed by an expression as follows:

$$\boxed{< variable\_identifier >:=< expression >} \tag{4.5}$$

A non-deterministic action can be recognized as follows:

$$\boxed{< variable\_identifier >: \epsilon < set\_expression >} \tag{4.6}$$

where set expression consists of all predicates of the variables in the machine.

Relating this to the traffic light example, a possible event can be allowing the cars to drive but taking into account that pedestrians have to stop. An event named **set_cars** can be created for this purpose, seen in Figure 4.8.

The event **set_cars** is the most abstract form for setting the traffic light, related to the machine **mac**. It is important to note that all events are modelled as easy as possible, and further refined if more details are needed. The event **set_cars** is modelled such that **peds_go** are set to FALSE each time the new value for **cars_go** is set to TRUE.

4.2.3 Context Extension

When new details and specifications are added to an existing model, it may be necessary to define new constants, axioms, or definitions which are not already present in the existing context. To introduce more static details into an Event-B model, context extension is used [46]. When a context D extends another context, C is referred to as the abstract context whereas D is referred to as the concrete context. When a context is extended, the concrete version inherits all the elements of the abstract context. This means that if we have a context D which extends context C:

- A context extending D also implicitly extends C
- A machine seeing D also implicitly sees C

The relationship of a context extension can be seen in Figure 4.4. It is important that the "Refines" and "Extends" relationship does not lead to any cycle [42].
If we want to add new sets to the already existing context, **ctx_1** from Figure 4.2, **ctx_1** is extended to a new context **ctx_2**. An example could be if we wanted to add a possibility for a button for the pedestrians to press when they want to cross, and are in need of a context for describing the button states.

Context extension allows for modular development of specifications, and reduces the need for duplication of definitions. It also simplifies maintenance of the formal specification due to the fact that all changes to a parent context are automatically inherited by the child context.

4.2.4 Refinement

Refinement is a mechanism used to introduce more details about the dynamic properties of a machine. When a machine N refines a machine M, M is referred to as the abstract machine whereas N is referred to as the concrete machine [48]. Machine refinement is a process of progressively refining an abstract machine into a concrete implementation that can be executed. It is a key aspect of Event-B modelling that allows for stepwise development and verification of a system, instead of introducing all details at once. This yields better model maintenance.

The events of a concrete machine are supposed to be simple and have deterministic assignments. Also, the concrete guard is stronger than the abstract one: when a concrete event is enabled, the corresponding abstract event should be enabled too. Explaining this in a clearer picture: The refinement process begins with an abstract machine with captures some desired behavior of the system at a high level of abstraction. This machine contains invariants, variables, events and guards, but a lot of the system implementation details are left unspecified. To add more detail, the abstract machine is refined to be more concrete, creating a so-called concrete machine. The concrete machine may introduce new variables, guards and events, and may refine existing elements of the abstract machine. A clear example is seen in the refinement from **mac** to **mac_1** from Figure 4.3 and Figure 4.5.

Assume a one-to-one correspondence between an abstract even e and a concrete event f where they are defined as follows:

$$e = \text{any } x \text{ where } G(x, v) \text{ then } Q(x, v) \textbf{ end} \tag{4.7}$$

$$f = \text{any } y \text{ where } H(y, w) \text{ then } R(y, w) \textbf{ end} \tag{4.8}$$

f refines e if the guard of f is stronger than the guard of e. This is called guard strengthening [46].

When an abstract event e is refined by more than one concrete event f, the abstract event e is said to be split [46]. It has to be proven that each concrete event is a valid refinement. A refined event of **set_cars** can be seen in Figure 4.9.

```
set_cars_colors   ≙
STATUS
 ordinary
REFINES
 set_cars
ANY
 new_value_colors
WHERE
 grd2   :   green ∈ new_value_colors ⇒ peds_color = red
 grd3   :   new_value_colors ⊆ COLORS
 grd_y_r   :   cars_colors = {yellow} ⇒ new_value_colors = {red}
 grd_r_ry   :   cars_colors = {red} ⇒ new_value_colors = {red, yellow}
 grd_ry_g   :   cars_colors = {red, yellow} ⇒ new_value_colors = {green}
 grd_g_y   :   cars_colors = {green} ⇒ new_value_colors = {yellow}
WITH
 new_value   :   new_value = TRUE ⇔ green ∈ new_value_colors
THEN
 act1   :   cars_colors ≔ new_value_colors
END
```

Figure 4.9: Refinement of event **set_cars**

The refinement of the event, now labelled **set_cars_colors**, provides a more detailed event with stronger guards. In addition, this refined event also introduces a way to transition colors for the cars, while assuring correct color for the pedestrians. As a result, the invariants that were previously established in **mac**, are still maintained but with strengthened conditions, allowing preservation of the refinement relationship. This process aligns with the principles of refinement of the Event-B methodology, where refinement ensures correctness between concrete and abstract events, while refining and expanding the functionality of the system.

If several abstract events e can be refined by one concrete f, the abstract events are merged. The abstract events must have identical actions as a requirement for merging events. To prove that the refinement of the merged events is correct, the guard of the concrete event needs to be proved to be stronger than the disjunction of the guards of the abstract events. This can be done with proof obligations.

4.2.5 Proof Obligations

Proof obligations define what is to be proved for an Event-B model to show the consistency of the machine and the correctness of theorems [42]. A proof obligation is a logical formula generated automatically in the Rodin platform. Rodin is based on the formalism of Event-B, and allows for the development of correct-by-construction systems through rigorous refinements [49]. Correct-by-construction is a methodology which guarantees that the system adheres to its specified requirements from its inception. The tool which provides proof obligations is called “Proof Obligation Generator”, and it static-checks each context or machine, and decides what to be proved.

Proof obligations are typically generated during the refinement process, the stepwise development of a formal specification from an abstract high-level description to a concrete low-level implementation. During the refinements, new properties are continuously added, or existing ones are modified, so the proof obligations are generated to ensure that the new properties are consistent with the existing specifications. This ensures that the overall sys-

tem behavior is correct.

A proof obligation in Rodin consists of a label, a number of hypotheses that can be used in the proof and a predicate that must be proven [2]. They are often seen as **eventlabel/label/PO**, where label is the name of the event and the relevant identifier, and PO stands for proof obligation. The PO is changed dependent on the proof obligation, i.e., WD for well-defined expressions, and INV for invariant preservation. When a proof obligation is proved to be correct and consistent for the model, Rodin *discharges* the proof. In order to discharge a proof, a theorem-prover in Rodin performs a formal proof or a counterexample with a model checker. Elements that have a proof which is not discharged are marked with a question mark, whereas discharged proofs has a green checker. This is seen in Figure 4.10

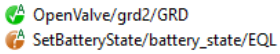


Figure 4.10: Discharged and non-discharged proof obligation

There are several types of proof obligations that can be generated in Rodin, depending on the nature of the refinement and the properties being verified. Some important, and the most common, proof obligations in Event-B machines are invariant establishment and preservation, variable preservation, guard strengthening, action simulation and action feasibility [46].

Invariant Establishment states that any possible, $\mathbf{K}(v')$ must satisfy the invariant I state after initialization given by the after predicate. The invariant establishment can be seen as a special invariant proof, but for the initialization event of the model. The initialization is not similar to other events in the sense that it does not contain guards and parameters [2]. The rule can be stated as follows:

$$\mathbf{K}(v') \vdash I(v') \tag{4.9}$$

where $\vdash$ means that the goal, here I, hold under the assumption of the set of hypotheses, here $\mathbf{K}(v')$ [46]. The initialization ensures that actions does not use variables before the initialization occurs, and that all variables are assigned a value. The invariant establishment is recognized as **INITIALISATION/invlabel/INV**.

Invariant preservation states that every event occurrence reestablishes the invariants. This means, that for every event e, assuming the invariants I and guards G, it must be proven that the invariants still hold in any possible state after the event execution [46]. The invariant should also hold when there are added new variables, guards, and predicates. The invariant preservation are seen as **eventlabel/invlabel/INV**.

Guards strengthening, identified as **eventlabel/guardlabel/GRD** states that a concrete event must only be enabled if the abstract event is enabled. Strengthening the guard means making the event more restrictive in the concrete machine, and the proof obligation ensures that the guards from the abstract machine is still not violated. Nevertheless, if a variable is a variable of both the abstract and the concrete machine, and the concrete event assign

a value to the variable and the abstract does not, it is important to prove that the variable's value does not change. This is a proof obligation called the equality of a preserved variable, and is recognized as **eventlabel/variable/EQL**.

The feasibility proof obligation rule purpose is to ensure that a non-deterministic action is feasible [42]. This means that there are always possible after-values, values of variables after the operation is performed, for variables satisfying the before-after predicate [46]. A before-after predicate if a condition or relation that should be valid before and after the execution. This means that there must always exist at least one possible state of the system (defined by the values of variables) after the action is performed, such that the before-after predicate remains true. For deterministic actions, feasibility is trivial. This proof obligation is kind of similar to the action simulation proof, where it is proven that the abstract event's behavior corresponds to the concrete event's behavior if an abstract event's action assigns a value to a variable that is also declared in the concrete event. This is seen as **eventlabel/actionlabel/SIM**.

There exists other proof obligations which Rodin also takes into account, such as guard merging proof obligation rule, well-definedness proof and witness feasibility proof [2]. These are all based on the use of context extension and refinement in Event-B. However, they are not taken into account in this thesis because they are not deemed necessary for the SE-STPA. For further reading on proof obligation, the reader is encouraged to read Section 5.2 in [42] and the Rodin Handbook [2].

4.3 SE-STPA

A way to incorporate formal verification and formal methods into risk analysis can be done by modifying the STPA introduced in Chapter 3.1. A significant improvement to STPA which includes security as well as safety is Security-Enhanced STPA, further referred to as SE-STPA. SE-STPA includes security analysis to the traditional STPA, in addition to integrating formal modelling to the analysis process. This approach provides a higher level of assurance to the STPA, making it possible to perform security and safety analysis within a single, highly traceable framework.

The primary objective of SE-STPA is to generate critical requirements, also known as system-level constraints, that mitigate against both malicious behaviors and hazardous control, similar to the output of traditional STPA [39]. To ensure the completeness of these critical requirements, formal modelling is done using Event-B, which is a powerful verification tool that can help to detect potential errors and ensure the accuracy of the resulting model.

Enhancing STPA is no novel aspect, as Young and Levenson themselves provided STPA-sec which attempts to introduce security into the STPA. However, SE-STPA takes this approach a step further by incorporating formal modelling to enhance the analysis process. By doing so, it becomes possible to generate more robust and comprehensive critical requirements, which can help to ensure the security and safety of complex systems in a range of domains. This differs from STPA in the sense that the requirements are

usually verified and produced by experts in the area.

The SE-STPA methodology consists of eight steps:

- **Step 1—Establishing the system engineering basis.** This step is essentially the same as the first STPA step of defining the purpose of the analysis, elaborated in Section 3.2.
- **Step 2—Build the control structure.** Step 2 relates directly to the second step in traditional STPA.
- **Step 3—Identify control actions**. Unlike STPA, all control actions and not only unsafe control actions are to be identified.
- **Step 4—Building the initial formal model using Event-B**. This step introduces formal modelling based on the identified control actions, and is one of the new steps.
- **Step 5—Control action analysis and identification of critical requirements** relates to analyzing the control actions when the action is applied, is not applied, applied too soon or too late, or issued for too long/too short. This is the same analysis done in the second step in STPA.
- **Step 6—Adversarial modelling and further generation of critical requirements**, where the purpose is to provide traceable and explicit security assurance
- **Step 7—Integration of generated critical requirements** leverages the existing formal model in Event-B using Rodin to validate each critical requirement
- **Step 8—Causal factors analysis** is the same as the fourth step in STPA, where the purpose is checking the hazards and how they might happen in the control loop. Possible additional requirements can also be generated.

In addition to the eight steps listed, it can be argued that there exist a ninth step which encompasses the iterative process of design, further refinements and potential re-scoping of the analysis. This allows for continuous improving and optimization of the safety analysis, ensuring that the safety requirements are adequately addressed.

Comparing SE-STPA to the STPA from Chapter 3.1, the differences and similarities can be seen in Table 4.2

Step	STPA	SE-STPA
Step 1	Establish the system engineering basis	Establish the system engineering basis
Step 2	Model the control structure	Model the control structure
Step 3	Identify unsafe control actions	Identify control actions
Step 4	Perform scenario analysis and generate constraints to address unsafe control actions	Build the initial formal model
Step 5	Iterate	Hazard analysis and critical requirement generation
Step 6	N/A	Adversary modelling and critical requirement generation
Step 7	N/A	Critical requirement integration into formal model
Step 8	N/A	Perform scenario analysis
Step 9	N/A	Iterate

Table 4.2: Comparisons of steps of STPA and SE-STPA. Adopted from [39]

As seen in Table 4.2, SE-STPA and STPA shares many steps even though the order is changed. However, Step 4 for building the initial formal model, Step 6 consisting of adversary modelling and Step 7 including integration of the model are unique for the SE-STPA, and will be further elaborated in the following sections.

4.3.1 Step 4—Building the initial formal model

When the control actions are identified from the model structure, an abstract model based these can be built using Event-B [44]. This step allows for abstraction of the system behavior which is to be modelled. Construction of a formal model also aids in determining whether the system understanding is adequate [39]. This means that any unclear effect of a given control action or if the process model seems unclear, it can be discovered in this step.

In Event-B, control actions are represented as events, whereas process models are depicted as a blend of variables and invariants. Additionally, restrictions or conditions on control actions can also be portrayed using guards on events. This method of system representation facilitates for a comprehensive understanding of the system [42].

To further describe the system formally, environmental requirements (ENV-X) can be determined and defined, as well as safety properties (SAF-X) and functional requirements (FUN-X). This helps to prevent misunderstandings, ensuring that all edge cases are con-

sidered, and making it easier to verify that the system as built correctly implements the specified requirements.

Environmental requirements and definitions are concerned with the structure of the system and its components, in addition to describing the context in which a system operates [42]. This includes interactions with the external environment, the available states in the system, and the system behavior in response to the system actions. Moreover, inputs and outputs for the system are usually described using the environmental requirements, as well as requirements for memory usage and timing. Hence, environmental requirements specify the assumptions and the constraints on the system behavior, often in relation to its environment.

Safety requirements define the properties which ensures that no accidents could happen [42]. More specifically, it specifies conditions or properties that a system must satisfy to ensure safe operation. This also includes requirements on fault tolerance, such as redundancy, and safety limits.

Functional requirements define what a system is intended to do, or the functionality it is expected to provide [42]. They serve as detailed blueprints of the desired operations of the system. This can be done in the form of expected events, intended goals of the system, or desired transitions in the system. For example, in a navigation software, an intended goal could be to provide the quickest route from point A to point B, serving as a functional requirement.

Environmental requirements, together with the functional requirements and the safety requirements, will define what is to be taken into account in the development of the formal model. Usually, there exist multiple environmental requirements to describe a functional requirement, which together form a safety requirement. The relationship between the environmental, functional, and safety requirements can be depicted in Figure 4.11. It is important to note that there is a feedback loop from the safety requirements to the functional requirements. This is because the foundation of formal verification based on the use of these requirements is the fact that the analysis is based on refinement.

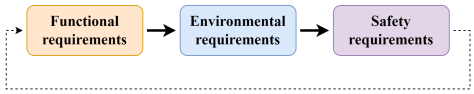


Figure 4.11: Relationship between the functional, environmental and safety requirements

The feedback loop ensures that the formal model accurately reflects the system's intended behavior, environmental constraints, and safety requirements, which is crucial for the formal verification process. Formal verification is a rigorous and systematic process for checking whether a system satisfies its intended behavior and safety properties, hence ensuring the foundation for the initial modelling is well described is important.

In other words, the functional requirements are refined to meet the safety requirements, and the environmental requirements are also refined to ensure that the system operates

safely in its specific context. This is seen in Figure 4.12. If the safety requirements necessitate more detailed functional or environmental requirements, going back to add more details is encouraged.

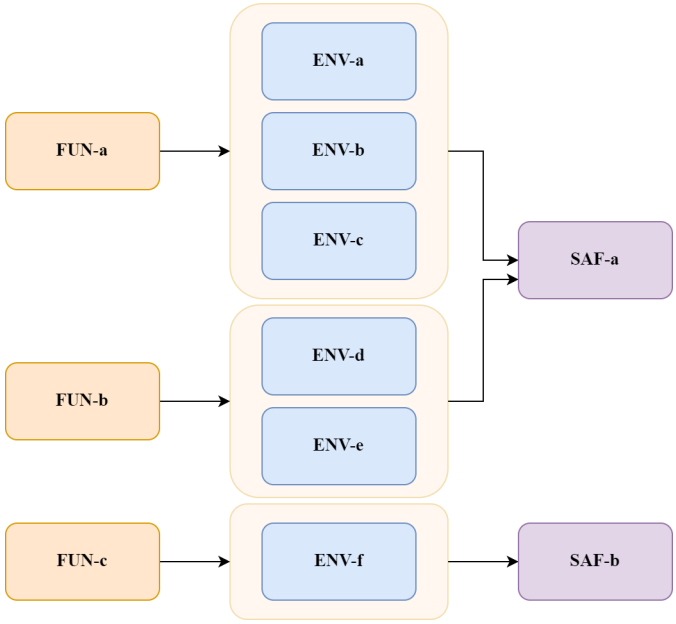


Figure 4.12: Detailed relationship between the functional, environmental and safety requirements

4.3.2 Step 6—Adversary modelling and generation of further critical requirements

An adversary is described as:

> [...] *an abstraction of any unauthorized party that may undermine the purpose of the system* [44].

An unauthorized party interacting with the system can be anyone who is not supposed to interact with the system, i.e., a fraudulent user or an unknown security guard.

An adversary can consist of the following six properties [39]:

- An identifier such that other aspects might be mapped back to this specific adversary
- A name/categorization
- The intent of the adversary, which can be something as simple as “curiosity” or “damage”.
- The information held by the adversary about the system

- The actions that an adversary might undertake

The primary objective of adversary modelling is to establish a traceable and well-defined framework for assessing and assuring the security of a system [39]. Adversaries, in this context, are akin to hazards in that they cannot be directly controlled by the system. However, unlike hazards, the behavior of adversaries is influenced by a range of factors that often present as hazards themselves. Thus, an adversary can be thought of as a malicious environment that the system must operate within.

To effectively manage the risk posed by adversaries, it is crucial to generate requirements that ensure that the system maintains control in a controlled manner, even in the face of identified hazards. By defining these requirements, it becomes possible to implement appropriate countermeasures that can mitigate the impact of potential attacks and prevent unauthorized access to sensitive data.

This part of the analysis will explore various approaches to adversary modelling, focusing on the development of requirements-based methodologies that can be used to systematically identify and analyze potential threats. By integrating these methodologies into existing security frameworks, it will be possible to enhance the security of complex systems and provide greater transparency and assurance. This way of modelling differs from Step 3 because instead of identifying possible contributing factors of loss of control, adversary modeling seek to directly undermine system control.

The critical requirements are essentially the same as constraints which are defined during step 3 in STPA, elaborated in Section 3.4.

4.3.3 Step 7 – Integration of critical requirements into the formal model

Integration of the critical requirements are done using the Rodin tool, where the goal is to ensure that the critical requirements mitigate against their associated hazard sufficiently. Each critical requirement is integrated using its own distinct refinement to enable traceability [39].

The integration can consist of the following to refine the model:

- Addition of invariants to constrain variables
- Addition of guards to narrow circumstances where they occur
- Addition of more actions to events
- Addition of axioms to add properties to the constants and carrier sets
- Addition of new events, variables and other elements to the model restricting existing events and variables

Many critical requirements usually result in invariants created. A violation of these will be indicated through an inability to discharge all the proof obligations associated with that invariant in the used tool.

4.4 SE-STPA-mod

In this thesis, the primary focus is on safety rather than security. Therefore, a modification to the SE-STPA methodology is proposed to align with this. While SE-STPA serves as the foundation, the aspect of adversaries and security considerations are not given significant attention in this modified approach. It is mainly the use of formal modelling in the STPA which will be the main novel aspect. By excluding the security aspect, the modified SE-STPA, further referred to as SE-STPA-mod, can provide a more specialized and targeted analysis specifically tailored for safety-critical systems.

Additionally, SE-STPA-mod has a slightly different order for the analysis. For instance, identification of unsafe control actions happens before building the formal model. Besides, the initial formal modelling is not taken into account as the control action analysis is done before formally defining the system. Every refinement can be done simultaneously.

This means that functional, environmental and safety requirements are defined after identifying and analyzing the control actions. The reason for swapping these steps is that the definition of different requirements seemed more reasonable after analyzing control actions and gaining a better understanding of what needs to be taken care of.

Loss scenario analysis

The scenario analysis in SE-STPA-mod is slightly modified as well. For STPA, loss scenarios are essential for gaining insight into potential failures or weaknesses in the control system, by evaluating how UCAs can occur. This is summarized in Figure 3.4. Typically, scenarios identified and designed during STPA can form a foundation for test scenarios that verify system behavior, with a key focus on creating context within the identified scenario. However, when formal verification is involved in the safety analysis, much of the system behavior is validated and verified through formal modeling and corresponding proof obligations. Ambiguities are usually excluded by creating relevant requirements and constraints. Consequently, there is no need for additional loss scenarios describing situations that have already been addressed and verified in the formal model.

For a STPA, loss scenarios are used to help identify hazards or unsafe conditions, and by understanding these scenarios, engineers, and designers can better anticipate potential issues in the design, and prevent them. They also serve as a basis for formulating safety requirements and constraints, but when incorporating formal modelling to the safety analysis, this is already handled directly after control action analysis. Therefore, the loss scenarios should not be necessary.

However, there may be events or requirements that cannot be adequately proven or modeled using formal methods, or there might be some aspects of the system that are overseen through the control action analysis early on. These can be regarded as ambiguities in the formal model and require verification through alternative means. In such cases, the STPA scenario analysis should be conducted. Once these loss scenarios are identified, test cases or new constraints can be designed to confirm the functionality to properly verify

every part of the novel system. So a scenario analysis should be conducted for the sake of completion and to ensure everything is taken into account.

The new and improved safety analysis, named SE-STPA mod, consists of the following steps:

- **Step 1:** Establish the system engineering basis.
- **Step 2:** Model the control structure and identify control actions
- **Step 3:** Identify unsafe control actions and perform a hazard analysis
- **Step 4:** Build the formal model
- **Step 5:** Further refinement and integration of new requirements
- **Step 6:** Perform scenario analysis for ambiguities in formal model
- **Step 7:** Iterate

How SE-STPA-mod differentiates from the traditional STPA, as well as a visualization of the modified method, can be seen in Figure 4.13. The differences are marked in purple. Notice that even though the loss scenario analysis not necessarily are conducted because of the formal modelling, it is included in the figure to show the purpose of througouh refinement. The original description of STPA is seen in Figure 3.1.

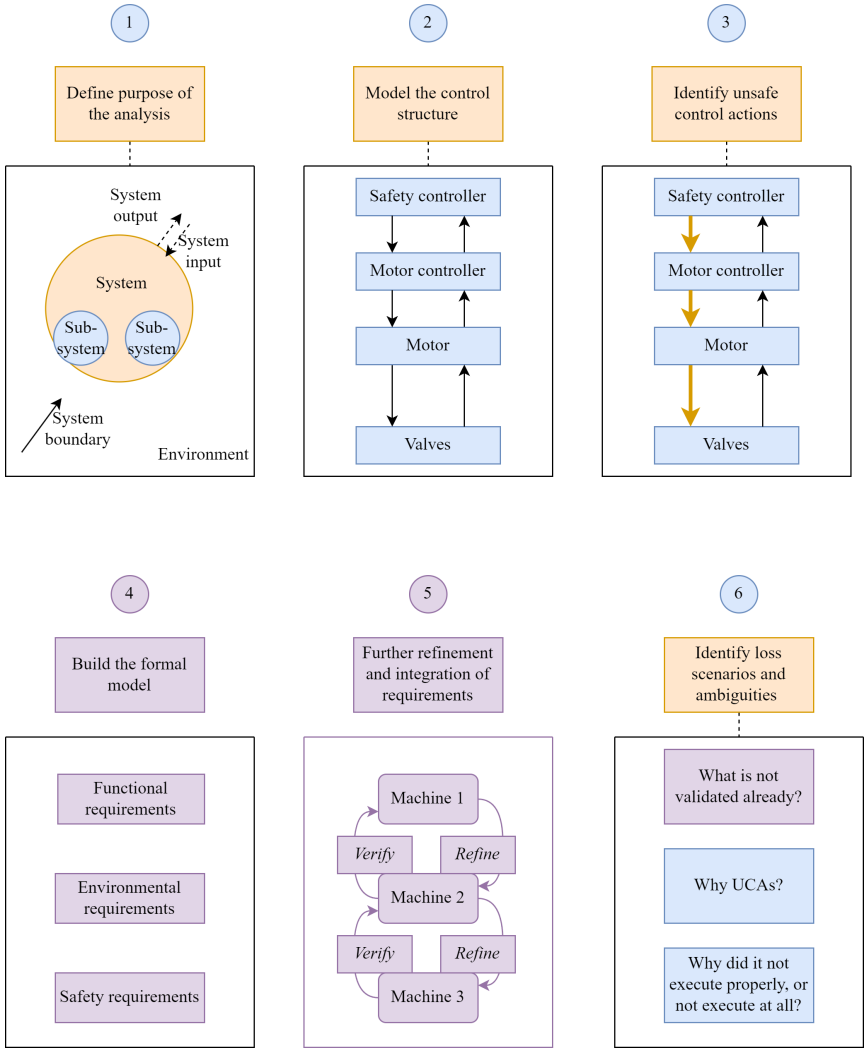


Figure 4.13: SE-STPA-mod method

Conducting an SE-STPA-mod analysis entails performing a comprehensive STPA while incorporating formal modeling into the process to achieve a better understanding of the system. The inclusion of formal modeling in STPA offers additional insights that may not necessarily require domain expertise, addressing a limitation of traditional STPA that typically relies on a team of experts, as well as reducing possible inconsistencies in requirements.

5

Safety analysis using SE-STPA-mod

In this chapter, the all-electric actuation system introduced in Section 2.5 is thoroughly analyzed by utilizing the SE-STPA-mod methodology described in Section 4.4 on the all-electric actuation system introduced in Section 2.5. Some parts of this analysis, such as the purpose of the analysis, are based on the analysis conducted in the specialization project [1], providing a continuity of exploration and expanding of the understanding of the system safety and functionality.

5.1 Purpose of the analysis

The purpose of the safety analysis conducted on the all-electric actuation system is to prove that the safety functions of the electric control system are safe. This involves ensuring that these safety functions complies with relevant regulations and standards, that they are fit for purpose, and that the all-electric safety valve systems improves the overall risk picture for subsea production.

As a reminder, a safety analysis using STPA was previously conducted in a specialization project about safety demonstration of the all-electric actuation system [1]. The purpose of the analysis is mainly the same, considering the losses, and the system-level constraints. Additionally, the model structure is essentially the same as in [1], but more detailed in this analysis, and the setup in terms of visualization is somewhat different. However, the hazards, the control actions and loss scenarios are not entirely traceable to the specialization project due to the fact that the analysis in this thesis is conducted in a more systematic manner, and with more knowledge than earlier, as well as the introduction of formal modelling, which requires the analysis to take a different approach.

Nevertheless, the analysis is to be conducted anew, but using SE-STPA-mod described

in Section 4.4. To determine the purpose of the analysis, and further elaborate on the goal and purpose of the system, losses, system-level hazards and system-level constraints will be determined.

5.1.1 Identify losses

The all-electric control system aims to prevent each loss that threatens the safety perspective of the installation. Such loss can be pollution of the environment by closing the flow path. The flow path is closed by the motor moving the valve stem. Additionally, it aims to prevent loss in the production of hydrocarbons when emergency shutdown is initiated, in other words that there is no production due to failed operational mechanisms. The motor and the motor controller is responsible for the valve opening and closing correctly.

Furthermore, the system aims to prevent explosions and blowouts, which can be hazardous and cause loss of life, loss of equipment, and loss of production. Blowouts can release highly flammable gases and fluids under high pressure, which can cause explosions and fires, which further can result in loss of life due to burns, inhalation of toxic gases and other related injuries on the people working at the reception facility [51]. They can also cause mechanical hazards such as equipment failure, or release toxic gas which will be quite damaging for the people on the well site or nearby facilities.

Hence, the losses that the all-electric control system aims to prevent can be recognized as L-1 to L-5, summarized in Table 5.2:

ID	Loss	Description
L-1	Environmental loss	Loss due to pollution of the environment
L-2	Production loss	No production due to failed operational mechanisms
L-3	Loss of life or person injury	Explosions and blowout
L-4	Loss of sensitive information	Information leakage, security breach or missing diagnostics
L-5	Loss of equipment	Wear and tear or destruction of equipment

Table 5.2: Summary of losses aimed to be prevented

However, L-4 is usually related to security and not safety. The analysis will focus on safety through the use of SE-STPA-mod and not on security. Hence, L-4 will therefore not be treated in sense of security in detail further but mentioned here for completeness. On the other hand, loss of information can also be traced to missing diagnostics in the control system, and is how the loss, L-4, is treated further.

5.1.2 Identify system-level hazards

A hazard is a system state which will lead to loss in some worst-case environment. Each loss L-1 to L-5 from Table 5.2 can be traced to a system-level hazard.

It's important to differentiate between the terms "system-level hazard" and "hazard" as they are distinctly defined within the framework of STPA. As a hazard can be seen as any potential source which could lead to loss, the system-level hazards can be considered as top-down system-level state conditions that must be prevented [36]. For this analysis, the total all-electric control system for valve actuation is considered to be the system in question, whereas the motor, motor controller, battery, and BMS are subsystems and refines hazards in more details. The total safety logic is collected in a safety controller, recognized as the top level in the hierarchy.

The system boundary can then be recognized as the electric x-mas tree which includes the all-electric actuation system, its controllers and the safety valves. This can be regarded as anything inside the box marked "Subsea" in Figure 2.7. Hazards concerning the environment outside the box in the x-mas tree, or topside, are not taken into account further.

Possible hazards related to the system can then be recognized as:

- **H-1**: The all-electric actuation system is not able to close the valve when the topside power supply is interrupted [L-1, L-3, L-5]
- **H-2**: The all-electric actuation system does not notify when the battery does not have enough energy [L-2, L-3]
- **H-3**: The all-electric actuation system is unable to provide diagnostics [L-2, L-4]
- **H-4**: The all-electric actuation system violate fail-safe standards during production [L-1, L-3]

5.1.3 Identify system-level constraints

Now that system-level hazards are identified, they need to be mapped to one or more system constraint. The constraints specify conditions that are needed for the system to prevent hazards, and how the system should behave to minimize these if they should occur.

Constraint	Condition
SC-1	The electronics should be powered by the battery when other power supply is unavailable [H-1, H-2]
SC-2	The all-electric valve actuation system must isolate well in case of ESD [H-1, H-4]
SC-3	The all-electric valve actuation system must provide diagnostics under all conditions [H-3]
SC-4	The all-electric valve actuation system must satisfy fail-safe standard requirements [H-4]

Table 5.4: System-Level constraints

An important aspect of the system-level constraints defined is that it includes the part of the system over which the system designers have some control [37]. This is why haz-

ards and losses are distinguished, because losses usually involve environments where the operator or system designer have no control at all.

Nonetheless, the constraints are based on basic assumptions and acceptable solutions from NOG 070 [11]. For instance, NOG 070 [G.4] states that the safety controller shall be able to monitor the ability to transfer commands [H-3 & SC-3], as well as ensuring that the necessary power sources are available and adequate such that the safety function can be performed [H-1, H-2 & SC-1].

5.2 Model the control structure

The control structure is modelled hierarchically, which means that the functional element with the highest executive authority is placed on top. This is recognized to be the safety logic controller. To elaborate further, a hierarchical control structure is a common approach used in complex systems to ensure that the control actions are coordinated and prioritized based on their level of importance.

The safety controller, serving as the system's highest executive authority, plays a critical role in ensuring that the system always can reach a safe state when required. It is responsible for making critical decisions that impact the overall safety of the system. The safety controller typically has the authority to override or modify the actions of subordinate control elements, such as sensors and actuators, to ensure that the system operates within safe limits. This is done by continuous monitoring of motor diagnostics, valve diagnostics and battery diagnostics. The monitoring relies on the feedback from the system components and the other controllers. The process model represents the controller's internal beliefs, which includes relevant aspects of the system or the environment.

In terms of authority, human interaction typically supersedes the safety controller. Human interaction can include pressing ESD initiation buttons, changing controller values or maintenance typically done in a control room topside. Such interaction are not considered further in this analysis, and therefore not included in the model structure.

The safety controller possesses the power to override decisions made by the BMS if it determines that the system cannot attain a safe state due to internal decisions made within the BMS. This is because of the authority of the safety controller.

Handling controller interaction is important, especially in systems where safety functions are reliant on software. In such systems, it is crucial to account for how controllers interact with each other to ensure proper coordination and functionality. However, from the model structure depicted in Figure 5.1, there is no connection between the motor controller and the BMS even though there is a possibility they have to collaborate. This is instead thought to be handled by the safety controller, leveraging its overview and authority.

The motor controller manages commands related to the opening and closing of the valves. This includes starting and stopping the motor such that the stem moves correctly, and ensuring that the commands are executed correctly through feedback. This can include

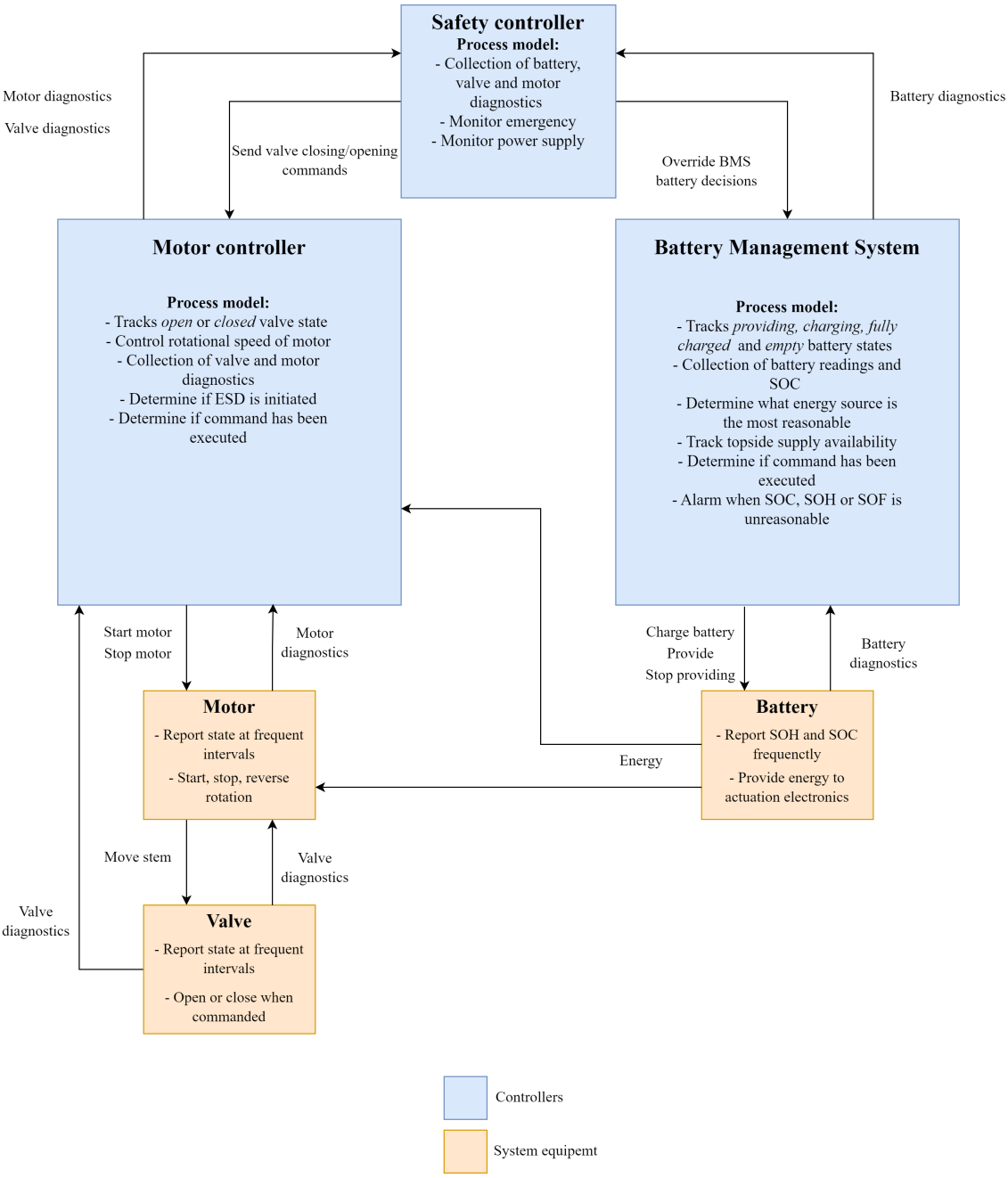


Figure 5.1: Hierarchical control structure

ensuring the valve is fully closed when supposed to, and open when required to.

The BMS handles energy providing. The main responsibility is to switch from topside supply to battery, and vice versa, as per the availability of the topside supply. The most crucial task of the BMS is to ensure that there always is a sufficient amount of energy to the required parts of the system, such that the system reaches a safe state when needed. Having fail-safe positions and ensuring that the system reaches fail-safe states are required by NORSOK S-001 [13].

5.2.1 Identify control actions

The model structure depicted in Figure 5.1 comprises three controllers: the safety controller, the motor controller and the BMS. Table 5.6 refers to the control actions which can be recognized from Figure 5.1, and is mainly a summary of the relevant control actions and the related controller.

ID	**Control action**	Controller
CA-1	Start motor	Motor controller
CA-2	Stop motor	Motor controller
CA-3	Send valve closing commands	Safety controller
CA-4	Send valve opening commands	Safety controller
CA-5	Switch to battery supply	BMS
CA-6	Switch to topside supply	BMS
CA-7	Override BMS decision that orders battery to not provide energy during emergency	Safety controller

Table 5.6: Identified control actions

5.3 Control action analysis and identification of critical requirements

Using the identified control actions from the control structure, this purpose of this part of the analysis is to examine how they could lead to the losses defined in the first step, i.e., how these action may lead to unsafe conditions. These unsafe control actions can be the basis for further development of functional requirements and constraints to incorporate in the formal model. The analysis investigates how the control actions from Table 5.6 can be hazardous when they are not executed, when they are executed, when they are executed for too long or when a control action is stopped too soon. The findings of this analysis are presented in Table 5.7.

Control action	Not providing causes hazard	Provides causes hazard	Started too early, too late, or in wrong order	Ended too soon or applied too long
Motor controller				
CA-1: Start Motor	**UCA-1**: Not starting the motor when the stem needs to be advanced or retracted in case of ESD or high DH pressure detected [H-3, H-4]	**UCA-2**: Starting the motor when the phase is unbalanced leads to a rise in the temperature in the motor windings [H-3, H-4] **UCA-3**: Motor is started to open the valve when valve is required to be closed [H-4]	**UCA-4**: Motor is started too early while temperature is still high [H-1] **UCA-5**: Motor controller reacts too late to an ESD and valve does not shut in time [H-1, H-4]	**UCA-6**: Stopping the action too soon might fail a soft start when required [H-3]
CA-2: Stop motor	**UCA-7**: Motor controller does not stop motor when approaching high temperatures or high friction, which could lead to quicker equipment wear down and damage to the interior of the motor [H-3, H-4] **UCA-8:** Motor is not stopped when valve reaches correct position	**UCA-9**: Motor is stopped when valve is in the wrong position [H-4]	**UCA-10**: Motor command to stop is given before stem has reached correct closing position [H-4]	**UCA-11**: Stopping the motor for too long might yield the motor unable to start when supposed to [H-4]
Safety controller				

Continued on the next page

Control action	**Not providing causes hazard**	**Provides causes hazard**	**Too early, too late, wrong of order**	**Stopped too soon, applied too long**
CA-3: Send valve closing commands	**UCA-12**: Valve not able to stop uncontrolled flow [H-3, H-4]	N/A	**UCA-13**: Valve subject to high stresses during closure, leading to potential degradation and damage to interior [H-4]	**UCA-14**: Valve might be stopped in a mid-position where hydrocarbons can flow through [H-4]
CA-4: Send valve opening commands	N/A	**UCA-15**: Valve opens when there is still unwanted hydrocarbon flow in well[H-4] **UCA-16**: Valve reopens accidentally during an emergency situation, leading to further escalation [H-4]	**UCA-17**: Valve is ordered to open during stressful situations [H-2, H-4]	N/A
CA-5: Override BMS decision that orders battery to not provide energy during emergency	**UCA-18**: ESD is initiated, and topside supply is unavailable [H-1, H-4]	**UCA-19**: Unnecessary stress and wear on battery while topside is available [H-2, H-3]	**UCA-20**: Overriding BMS decision too late might cause valve to not close in time [H-1, H-4]	**UCA-21**: Overriding BMS decision for too long might harm the battery SOH [H-1, H-2, H-3, H-4]
Battery management system				
CA-6: Switch to battery supply	**UCA-22**: Motor is not provided with power to open or close valve [H-1, H-4]	**UCA-23:** Unnecessary discharge and stress to battery, potentially reducing battery SOH [H-1, H-2, H-3]	**UCA-24:** Motor not able to start on time, leading to longer valve closing time [H-1, H-2, H-4]	**UCA-25**: Stopping the action too soon leads to BMS not registering a switch [H-1, H-3]

Continued on the next page

Control action	**Not providing causes hazard**	**Provides causes hazard**	**Too early, too late, wrong of order**	**Stopped too soon, applied too long**
CA-7: Switch to topside supply	**UCA-26**: Unnecessary discharge and stress to battery when topside supply is available [H-1, H-2, H-3]	**UCA-27**: Topside supply is not available to provide the actuation system with enough energy to start or stop motor [H-1, H-4]	**UCA-28**: Switch to topside supply is done too early while topside is not yet available, such that power is not provided to the system [H-1, H-4]	N/A

Table 5.7: Control action analysis

The control action analysis was based on some findings from the previously conducted specialization project [1], as well as discussions with the supervisors for this thesis. The parts marked with N/A are considered to not be unsafe when occurring. Through multiple iterations and revisions, Table 5.7 was formed.

5.4 Building the formal model

Once the unsafe control actions have been identified, this stage of the analysis facilitates the abstraction of system behaviors, setting the stage for modeling them formally. Using the process model and responsibilities from Figure 5.1, the initial formal model can be quite straightforward. The control actions can be modelled as events, the process models as combinations of variables and invariants, and restrictions or conditions on the actions are modelled as guards. Such translation of the system aids in determining if the understanding of the system is adequate, as previously mentioned. Additionally, insight from relevant standards such as ISO 13268-6 [15] and IEC 61508 [7] are also considered.

To determine what should be taken into account in the formal model, environmental, functional and safety definitions and requirements are to be defined. These requirements will describe the informal model of the system, before translating it to the formal language with mathematical notation. A formal model is a precise representation of the system's behavior based on the conditions and constraints under which the system operates using the appropriate mathematical language.

Functional requirements:

- **FUN-1:** The goal of the actuation system is to safely control the valve stem position.
- **FUN-2:** The safety controller is in control of executing safety commands.
- **FUN-3:** A valve during ESD is always closed.
- **FUN-4:** The BMS is in charge of the battery state-of-health and state-of-charge to ensure optimal performance and reliability of the battery.
- **FUN-5:** The temperature and DH pressure level is limited.
- **FUN-6:** The valve is normally open during ordinary operations.
- **FUN-7:** The stem and motor is usually idle during ordinary production.

The functional requirements are derived from the desired functionality or behavior of the actuation system, mostly from the general description given in Chapter 2. In addition to this, some functional requirements are inspired by the functional requirements given in NORSOK S-001 [10].

FUN-1 emphasizes the need for the actuation system to be equipped with appropriate and necessary control mechanisms and feedback systems to ensure safe positioning of the valve stem. It is important that all safety functions where electrical, electronic and programmable electronic systems are used, such as the electric actuation system, comply to IEC 61508, especially IEC 65108-2 [26].
FUN-2, derived from the safety guidelines outlined in NOG 070 [52], implies that the system should have a dedicated safety logic that can interpret safety commands and take appropriate actions to ensure safe operation of the system. The guideline underscore the need for a dedicated safety controller that should have the ability to bring the system to a safe state when there are interconnected systems.

FUN-3 is based on the requirement of a safe state during ESD, which is a closed valve in this system. The ESD shall be designed so that it enters safe conditions if a fault occurs that prevents the system from functioning, as required by §33 [53].

FUN-4 stems from the general application of a BMS, described in Section 2.5.1. It is important that the all-electric system has reliable backup power in case of emergency. Critical power consumer functions should always be maintained during ESD-shutdowns [54]. The same goes for FUN-6 and FUN-7, which describes otherwise normal and expected behavior of the valve, the stem, and the motor. The valves which are relevant for this system are recognized as PMV, PWV and DHSV, which are closing valves and therefore usually open [11]. FUN-7 are especially rooted in the fact that the system should take measures to reduce energy consumption, as indicated by §55 [55].

FUN-5 on the other hand, is based on the specification and requirement from NORSOK S-001 [10] which specifies that the system and components shall be designed and protected to ensure that it will remain operative if the system has a role of a safety barrier.

Overall these functional requirements are aligned with industry guidelines and standards, specifically the guidelines mentioned in Section 2.2, and are crucial in ensuring reliable operation. All the guidelines have basis in regulations from the PSA. The different references for the functional requirements found in various standards, as well as the relevant regulations from the PSA, can be seen in Table 5.8.

ID	Regulation	Standard or guideline [Chapter]
FUN-1	§55 [55]	IEC 61508-1: *7.7*, IEC 61508-2: *7.6*
FUN-2	§15 [56]	NOG 070: *G.4.1*, IEC 61508
FUN-3	§33 [53]	NORSOK S-001: *11*
FUN-4	§38 [57]	ISO 13702: *10*
FUN-5	§8 [58]	NORSOK S-001: *12.4, 13*
FUN-6	§53 [22]	NOG 070: *A.6*
FUN-7	§55 [55] (§61b)	NORSOK S-003: *5.3*

Table 5.8: References to relevant standards for functional requirements

To ensure that the functional requirements are maintained, environmental requirements as well as safety requirements are to be determined. The environmental requirements and the safety requirements are based on the analysis conducted in Table 5.7, as well as using the relevant guidelines mentioned in Table 5.8. Therefore, the environmental requirements can be seen as further elaborations of the functional requirements.

Environmental requirements:

- **ENV-1:** An all-electric valve actuation system may contain special components: Motor, stem, battery, and valves.
- **ENV-2:** A motor may have three working states: stopped, working or reversed.
- **ENV-3:** A valve may have two positions: open or closed.
- **ENV-4:** The battery has three states: Idle, charging, or providing.
- **ENV-5:** A motor is always attached to a stem and a valve.
- **ENV-6:** A motor is attached to a motor controller, which sends pulses to control the motor speed and state.
- **ENV-7:** A valve cannot be open and closed at the same time.
- **ENV-8:** The stem cannot be moving if the motor is stopped.
- **ENV-9:** The motor and motor controller may receive energy from a battery or topside power source.
- **ENV-10:** The valve is supposed to close during an ESD.
- **ENV-11:** The battery should provide energy as soon as topside power supply is interrupted.
- **ENV-12:** The battery should never be empty.
- **ENV-13:** The battery should not be providing energy to the system if topside power supply is working
- **ENV-14:** The safety controller system is equipped with a pressure sensor to detect unsafe DH pressure levels.
- **ENV-15:** The battery should be charging whenever the battery SOH allows it.
- **ENV-16:** The stem has three movements: Inactive, retracting or extending.
- **ENV-17:** The safety system is equipped with a temperature sensor to detect high temperatures in the battery and the motor.
- **ENV-18:** The temperature sensor indicates whether the temperature is acceptable or unacceptable.
- **ENV-19:** The pressure sensor indicates whether the pressure is safe or unsafe.

The environmental requirements are formulated based on the functional requirements, and can be mapped such as seen in Figure 5.2

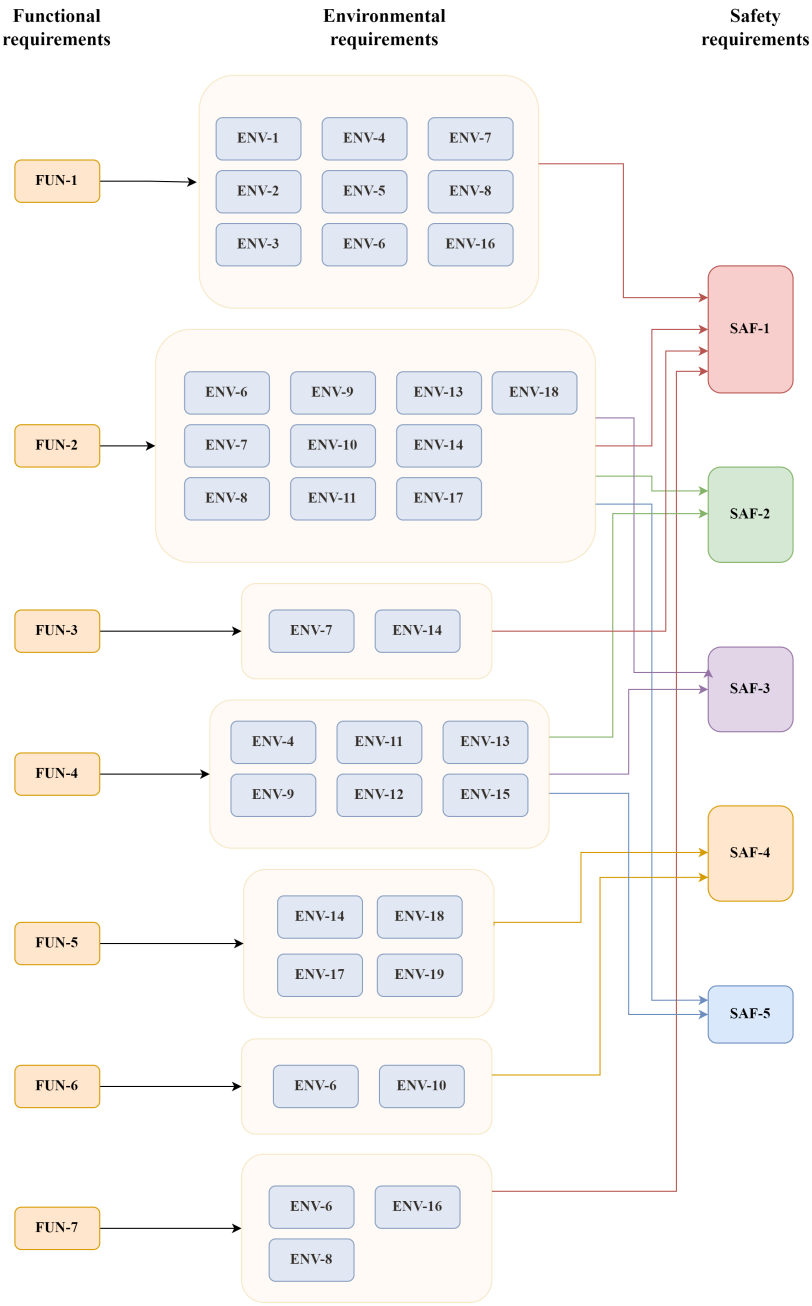


Figure 5.2: Mapping between functional, environmental and safety requirements

The safety requirements, SAF-1 to SAF-5 seen in Figure 5.2 are formulated based on

the functional and environmental requirements, as well as the UCA analysis results seen in Table 5.7.

Safety requirements:

- **SAF-1:** When the ESD is engaged, the motor must start and run until the safety valve is closed.
- **SAF-2:** When UPS is unavailable, the battery must deliver energy to the electronic system.
- **SAF-3:** The battery should not stop providing energy during an action. Hence, the system should always have energy available.
- **SAF-4:** If DH pressure is too high, the valve cannot be opened.
- **SAF-5:** The safety logic can override the BMS if the motor needs the battery.

Furthermore, these requirements are abstracted and simplified to fit into an initial model of the system in Event-B. The simplification can be seen as a translation from text requirements into a mathematical formulation of the requirement.

Starting off with creating a context to describe the equipment and the belonging states. The first context is named **as_ctx** to represent the first context for the “Actuation System” and can be seen in Figure 5.3.

```
CONTEXT
  as_ctx
SETS
  MotorStates
  ValveStates
  BatteryStates
  StemMovements
  TemperatureSensor
  PressureSensor
  TopsideSupply
CONSTANTS
  Stopped
  Working
  Open
  Closed
  FullyCharged
  Depleted
  Charging
  Providing
  Retracting
  Extracting
  Inactive
  Acceptable
  Unacceptable
  Safe
  Unsafe
  Available
  Unavailable
AXIOMS
  axm1   :   partition(MotorStates, {Stopped}, {Working})
  axm2   :   partition(ValveStates, {Open}, {Closed})
  axm3   :   partition(BatteryStates, {FullyCharged}, {Depleted}, {Charging}, {Providing})
  axm4   :   partition(StemMovements, {Retracting}, {Extracting}, {Inactive})
  axm5   :   partition(TemperatureSensor, {Acceptable}, {Unacceptable})
  axm6   :   partition(PressureSensor, {Safe}, {Unsafe})
  axm7   :   partition(TopsideSupply, {Available}, {Unavailable})
END
```

Figure 5.3: Initial state context

as_ctx takes mainly into account the environmental requirements to describe the states in the system, such as ENV-1, ENV-2, ENV-3, ENV-4, ENV-16, ENV-17, ENV-18 and ENV-19.
The axioms seen in **as_ctx**, for example, axm1, states that $Stopped \neq Working$. This is the definition on the use of **partition**, as described in Section 4.2.1.

To ensure that the system always is provided with energy as required by SAF-3, the first machine of the formal model focuses on the power supply. To retain maintainability in the formal model, it is important to keep certain parts of the system in its own machine, and further refine to add details. This means that the initial machine only takes care of power supply, and should be the only machine which handles the battery state. The initial machine, named **as_mac**, which handles transitions and the dynamic part of the model regarding the battery state, can be modelled as seen in Figure 5.4.

```
MACHINE
  as_mac
SEES
  as_ctx
VARIABLES
  battery_state
  topside_supply
  battery_level
INVARIANTS
  inv4   :   battery_state ⊆ BatteryStates
  inv5   :   topside_supply ∈ TopsideSupply
  inv6   :   0 < battery_level ∧ battery_level ≤ 100
  inv7   :   topside_supply = Unavailable ⇒ Providing ∈ battery_state
EVENTS
  INITIALISATION   ≙
  STATUS
   ordinary
  BEGIN
   act1   :   battery_state ≔ {Charging}
   act2   :   topside_supply ≔ Available
   act3   :   battery_level ≔ 100
  END

  SetBatteryState   ≙
  STATUS
   ordinary
  ANY
   new_battery_state
  WHERE
   grd1   :   Charging ∈ battery_state ∧ battery_level = 100 ⇒ FullyCharged ∈ new_battery_state
   grd2   :   topside_supply = Unavailable ⇒ new_battery_state = {Providing}
   grd3   :   topside_supply = Available ∧ battery_level < 100 ⇒ new_battery_state = {Charging}
   grd4   :   0 ≤ battery_level ∧ battery_level ≤ 20 ⇒ new_battery_state = {Depleted}
   grd5   :   topside_supply = Unavailable ∧ battery_level < 100 ⇒ battery_state = {Providing, Charging}
   grd6   :   topside_supply = Available ∧ battery_level = 100 ⇒ new_battery_state = {FullyCharged}
   grd7   :   battery_state = {Depleted} ∧ (battery_level < 100 ∧ battery_level > 20) ⇒ new_battery_state = {Charging}
  THEN
   act1   :   battery_state ≔ new_battery_state
  END

END
```

Figure 5.4: Initial actuation system machine model

Contexts usually encompass only environmental requirements, whereas machines also can include safety requirements. The safety requirements are usually modelled as guards or invariants, but typically, they are treated as invariants, given their essential role in stipulating conditions that must be consistently maintained throughout the system's operation. For instance, *inv6* can be associated with ENV-12, and *inv7* is tailored to fulfill SAF-2. During the refinement process, additional guards and invariants may be included to refine the machine's behavior, and mirror more detailed behavior of the system.

The first event always created by a machine is the INITIALISATION event, seen in Figure 5.4. The initialization event for **as_mac** sets the battery level to full to begin with, with topside supply available. It is important that the initialization maintains the invariants, even though it is straight forward with no transitions.

Not taking the initialization into account, the first event of this machine is the **SetBatteryState**-event, which works as a state machine for the battery. The transition diagram for

the **SetBatteryState**-event can be seen in Figure 5.5. The transition diagram represents the same as the guards and transitions in the event, but graphically, for a better understanding of the context of the transitions.

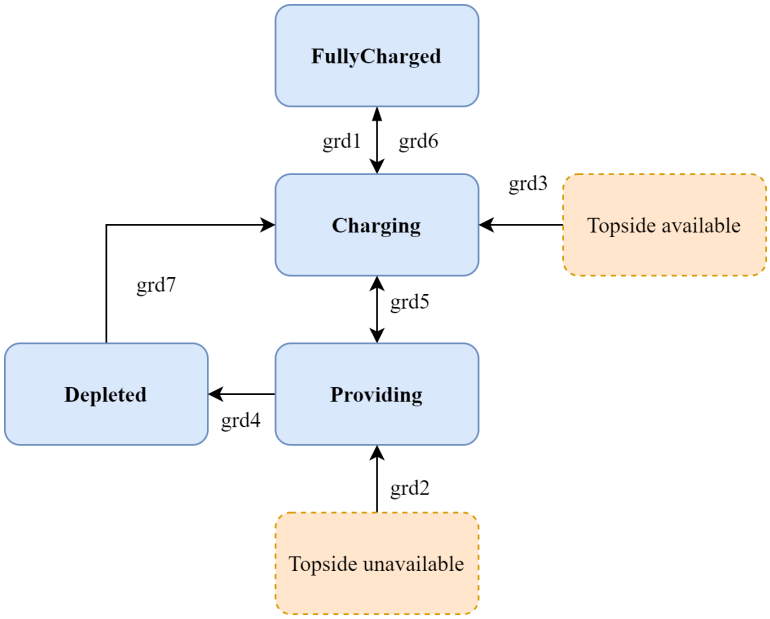


Figure 5.5: Transition diagram for **SetBatteryState**

While modelling, Rodin automatically keeps track of the proof obligations and ensures that no invariants are violated in the events. This is seen as a green checker in the Rodin Platform, seen in Figure 5.6

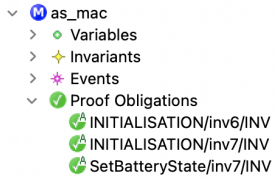


Figure 5.6: Discharged proof obligations

The proof obligation labels seen in Figure 5.6 indicate the origin in the model where they were generated. For example, *INITIALISATION/inv6/INV* is the proof obligation that must be verified to ensure that the event *INITIALISATION* preserves the invariant (INV) with label *inv6* [2]. However, if there were an event or an initialisation which set the battery level to anything outside 0-100, the proof would not have been discharged.

5.5 Further refinement and integration of requirements

The integration of requirements outlined in Table 5.4, as well as the functional, environmental and safety requirements from the previous section, can be achieved through the process of refinement. Refinement, as previously mentioned, involves systematically elaborating and specifying the existing formal model. In this process, additional environmental, functional, or safety requirements may be identified and incorporated to enhance the critical constraints if needed.

The presentation of this integration will also be carried out using the Rodin Tool, as in previous formal modelling presentations in earlier sections. However, some events turned out to be too long and comprehensive to be visualized with the typical Rodin format. For the purpose of better visualization of the events, an extension called "B2Latex" was added to Rodin for generating the formal model to Latex output [59].

Furthermore, the integration process of new and refined events will be continuously validated through proof obligations, ensuring that the resulting model is reliable and consistent with previous proofs.

5.5.1 Further refinement of as_mac

The current formal model, **as_mac**, only takes the battery state and transitions into account, as seen in Figure 5.5. The next refinement concerns some ESD functionality and the use of the pressure sensor. More specifically, this refinement concerns when to initiate the ESD with regard to the pressure level. The previous event, SetBatteryState from **as_mac**, is not refined, hence not taken into account for this refinement. This is done to spare the reader from repetition of events when they are unchanged. The total model refinement is seen in Appendix B. Furthermore, two new invariants and a new event, InitiateESD, is added to the refined machine **as_mac1**, as well as the Initialisation event being refined. This can be seen in Figure 5.7

```
MACHINE
  as_mac1
REFINES
  as_mac
SEES
  as_ctx
VARIABLES
  battery_state
  topside_supply
  battery_level
  pressure_level
  ESD_initiated
INVARIANTS
  inv1   :   pressure_level ∈ PressureSensor
  inv2   :   ESD_initiated ∈ BOOL
EVENTS
  INITIALISATION   ≙
   extended
  STATUS
   ordinary
  BEGIN
   act1   :   battery_state ≔ {Charging}
   act2   :   topside_supply ≔ Available
   act3   :   battery_level ≔ 100
   act4   :   pressure_level ≔ Safe
   act5   :   ESD_initiated ≔ FALSE
  END

  InitiateESD   ≙
  STATUS
   ordinary
  ANY
   new_ESD_state
  WHERE
   grd1   :   pressure_level = Unsafe ∧ ESD_initiated = FALSE ⇒ new_ESD_state = TRUE
   grd2   :   pressure_level = Safe ∧ ESD_initiated = TRUE ⇒ new_ESD_state = FALSE
  THEN
   act1   :   ESD_initiated ≔ new_ESD_state
  END

END
```

Figure 5.7: First refinement

The event InitiateESD describes the transitions of the ESD functionality. The ESD indication is FALSE when the pressure level has reached a safe state, and TRUE when the pressure level is over acceptable pressure levels. The transitions can be seen in Figure 5.8

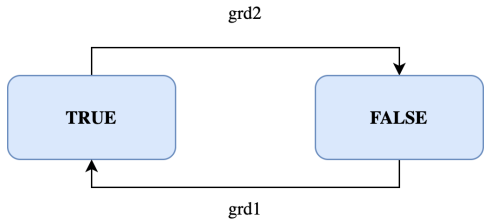


Figure 5.8: ESD_initiated state transition

With the refinements done in **as_mac1**, there are still a lot of functionality defined in the model requirements which are missing. Consequently, the refinement process was repeated twice, resulting in **as_mac2** and **as_mac3**.

The model **as_mac2** underwent further refinement to incorporate the safety controller. As a result, any event that depended on the safety controller could also be refined. In particular, this allowed the original InitiateESD event to be extended and refined to include safety commands emanating from the safety controller. The revised event can be seen in Figure 5.9.

Event InitiateESD_safety ⟨ordinary⟩ $\widehat{=}$
refines InitiateESD
any
new_ESD_state
where
grd1: $pressure\_level = Unsafe \land ESD\_initiated = FALSE \Rightarrow new\_ESD\_state = TRUE \land safety\_command = \{CloseValve, OverrideBMS\}$
grd2: $pressure\_level = Safe \land ESD\_initiated = TRUE \land CloseValve \in safety\_command \Rightarrow new\_ESD\_state = FALSE$
grd3: $pressure\_level = Safe \land ESD\_initiated = FALSE \Rightarrow new\_ESD\_state = FALSE$
grd4: $ESD\_initiated = FALSE \Rightarrow safety\_command = safety\_command \setminus \{CloseValve\}$
grd5: $new\_ESD\_state = TRUE \Rightarrow safety\_command = safety\_command \cup \{CloseValve\}$
grd6: $new\_ESD\_state = TRUE \Rightarrow CloseValve \in safety\_command$
grd7: $ESD\_initiated = TRUE \land CloseValve \in safety\_command \Rightarrow new\_ESD\_state = TRUE$
then
act1: $ESD\_initiated := new\_ESD\_state$
end

Figure 5.9: Refined InitiateESD event

Note that the event in Figure 5.9 differs in appearance from those in Figure 5.7. This is because the event definitions became too long and comprehensive, and the Rodin output were too small. Hence, the “B2Latex” extension for Rodin was used here.

In line with requirement FUN-2, it’s crucial to adjust the state of the safety controller

to match the current state of the system appropriately. This is done using the event SetSafetyCommand. The safety command state requires new states which are not defined in **as_ctx**, hence a context extension were required. The new context includes the states which can be used by the safety controller, $Idle$, $CloseValve$, $OpenValve$ and $OverrideBMS$, and can be seen in Figure 5.10.

```
CONTEXT
  as_ctx1
EXTENDS
  as_ctx
SETS
  SafetyController
CONSTANTS
  CloseValve
  OpenValve
  OverrideBMS
  Idle
AXIOMS
  axm1   :   partition(SafetyController, {CloseValve}, {OpenValve}, {OverrideBMS}, {Idle})
END
```

Figure 5.10: Context extension of **as_ctx**

However, safety_command is a set from SafetyController. This means that there is a possibility of the safety state consisting of two states at the same time, such as

$$\{OpenValve, OverrideBMS\}$$

This is defined in the invariants for **as_mac2**, seen in Figure 5.11

```
INVARIANTS
    inv1:   safety_command ⊆ SafetyController not theorem ›
    inv2:   ESD_initiated = TRUE ⇒ CloseValve ∈ safety_command not theorem ›
    inv3:   valve_position ∈ ValveStates not theorem ›
    inv4:   OverrideBMS ∈ safety_command ⇒ Providing ∈ battery_state not theorem ›
    inv5:   ESD_initiated = FALSE ⇒ CloseValve ∉ safety_command not theorem ›
```

Figure 5.11: Invariant definition for new machine, **as_mac2**

The safety controller state depends on the valve position to determine whether the system is in a safe state or not, hence an event which sets the valve position is also required. Additionally, the safety controller can override BMS decisions in parallel with determining if the system is in a safe state. The flow chart for the safety controller decisions can be seen in Figure 5.12, and describes how the guards are defined for the corresponding event.

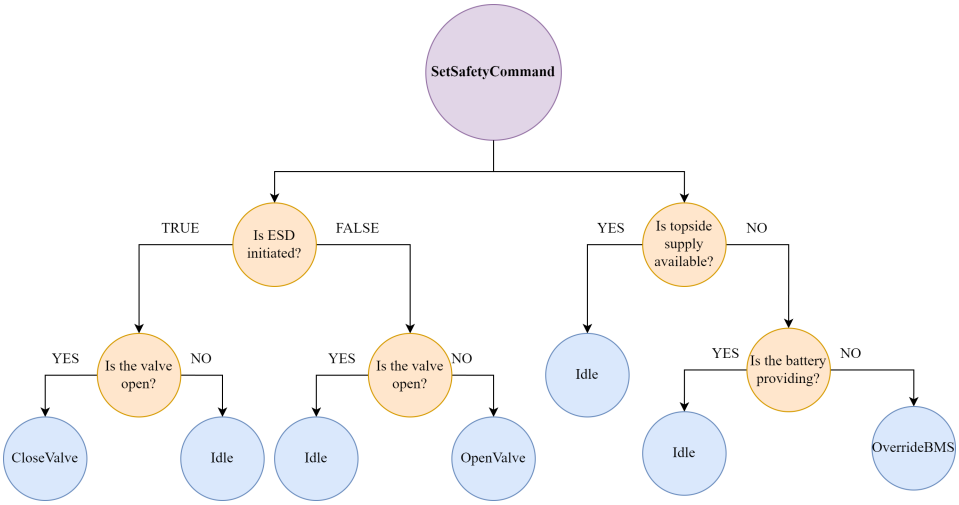


Figure 5.12: SetSafetyCommand flowchart

The two events, SetSafetyCommand and SetValvePosition respectively, can be seen in Figure 5.13. The event for SetSafetyCommand is the same as seen in Figure 5.12, represented using the Event-B formulation instead.

Event SetSafetyCommand ⟨ordinary⟩ $\widehat{=}$

any

new_safety_command

where

grd1: $ESD\_initiated = TRUE \wedge valve\_position = Open \Rightarrow new\_safety\_command = \{CloseValve\}$

grd2: $ESD\_initiated = TRUE \Rightarrow new\_safety\_command = \{CloseValve\}$

grd3: ⟨theorem⟩ $topside\_supply = Unavailable \wedge Providing \notin battery\_state \Rightarrow OverrideBMS \in new\_safety\_command$

grd4: $ESD\_initiated = TRUE \wedge valve\_position = Closed \Rightarrow new\_safety\_command = \{Idle\}$

grd5: $ESD\_initiated = FALSE \wedge valve\_position = Closed \Rightarrow new\_safety\_command = \{OpenValve\}$

grd6: $ESD\_initiated = FALSE \wedge valve\_position = Open \Rightarrow new\_safety\_command = \{Idle\}$

grd7: $ESD\_initiated = FALSE \Rightarrow new\_safety\_command = \{Idle\}$

grd8: $OverrideBMS \in new\_safety\_command \Rightarrow Providing \in battery\_state$

then

act1: $safety\_command := new\_safety\_command$

end

Event SetValvePosition ⟨ordinary⟩ $\widehat{=}$

any

new_valve_position

where

grd1: $pressure\_level = Safe \wedge OpenValve \in safety\_command \wedge ESD\_initiated = FALSE \Rightarrow new\_valve\_position = Open$

grd2: $pressure\_level = Unsafe \wedge CloseValve \in safety\_command \Rightarrow new\_valve\_position = Closed$

then

act1: $valve\_position := new\_valve\_position$

end

Figure 5.13: New events in **as_mac2**

The next refinement, **as_mac3**, ensures that the correct stem movements are set compared to the safety action and previous valve position. To do this, a new event SetStemMovement is created, and SetValvePosition is refined to include the stem movements in the guards. The refined events for **as_mac3** can be seen in Figure 5.14

Event SetStemMovement ⟨ordinary⟩ $\widehat{=}$
any
new_stem_movement
where
grd1: $valve\_position = Open \wedge CloseValve \in safety\_command \Rightarrow new\_stem\_movement = Extracting$
grd2: $valve\_position = Closed \wedge OpenValve \in safety\_command \Rightarrow new\_stem\_movement = Retracting$
grd3: $valve\_position = Closed \wedge CloseValve \in safety\_command \Rightarrow new\_stem\_movement = Inactive$
grd4: $valve\_position = Open \wedge OpenValve \in safety\_command \Rightarrow new\_stem\_movement = Inactive$
grd5: $stem\_movement = Retracting \wedge valve\_position = Open \Rightarrow new\_stem\_movement = Inactive$
grd6: $stem\_movement = Extracting \wedge valve\_position = Closed \Rightarrow new\_stem\_movement = Inactive$
grd7: $safety\_command = \{Idle\} \Rightarrow new\_stem\_movement = Inactive$
then
act1: $stem\_movement := new\_stem\_movement$
end
Event SetValvePosition ⟨ordinary⟩ $\widehat{=}$
refines SetValvePosition
any
new_valve_position
where
grd1: $pressure\_level = Safe \wedge OpenValve \in safety\_command \wedge ESD\_initiated = FALSE \Rightarrow new\_valve\_position = Open$
grd2: $pressure\_level = Unsafe \wedge CloseValve \in safety\_command \Rightarrow new\_valve\_position = Closed$
grd3: $valve\_position = Closed \wedge stem\_movement = Retracting \Rightarrow new\_valve\_position = Open$
grd4: $valve\_position = Open \wedge stem\_movement = Extracting \Rightarrow new\_valve\_position = Closed$
then
act1: $valve\_position := new\_valve\_position$
end

Figure 5.14: Inclusion of stem movement in **as_mac3**

Initialization of the different machines are left out for convenience, but the complete formal model can be seen in Appendix B.

5.5.2 New safety constraints

An essential consideration of this safety and risk analysis is to assess whether the analysis conducted in Table 5.7 has introduced some novel safety constraints or new environmental requirements. It is important to avoid the UCAs in the most effective way possible by either adding new invariants, variables, or guards to some events. Accordingly, the analysis in Table 5.7 inspires the formulation of previously unconsidered safety constraints, which could prove useful in enhancing the overall safety and reliability of the system:

Safety requirements:

- **SAF-7:** Motor should not start when temperature is too high [*UCA-4*]
- **SAF-8:** Motor should stop when correct valve position is reached [*UCA-8, UCA-9, UCA-10*]

To include SAF-7 to the existing machine, **as_mac3**, a new refinement, **as_mac4**, was required to incorporate the motor states and the use of a temperature sensor. In addition to the use of a temperature sensor, an event **SetMotorState** was added to include SAF-8, seen in Figure 5.15.

Event SetMotorState ⟨ordinary⟩ $\widehat{=}$
  any
    new_motor_state
  where
    grd2: $(stem\_movement = Retracting \lor stem\_movement = Extracting) \land temperature = Acceptable \Rightarrow new\_motor\_state = Working$
    grd3: $Providing \notin battery\_state \land topside\_supply = Unavailable \Rightarrow new\_motor\_state = Stopped$
    grd4: $temperature = Unacceptable \Rightarrow new\_motor\_state = Stopped$
    grd5: $stem\_movement \neq Inactive \Rightarrow new\_motor\_state = Working$
  then
    act1: $motor\_state := new\_motor\_state$
  end

Figure 5.15: New event for **as_mac4**

However, determining whether the correct valve position was reached proved to be too complex for the existing formal model. To include some constraints as to when the motor should be working and not, the use of stem movements and the temperature sensor was used instead. This is seen in the SetMotorState event in Figure 5.15. When the stem is in movement and the temperature in the motor is acceptable, the motor should be working, whereas the motor should stop as soon as the temperature reaches an unacceptable level. The temperature level changes depending on the phase in the motor, since an unbalanced phase might lead to unacceptable motor temperatures, or if the speed is incorrect. Additionally, the motor will not be working when there is no energy supply, and as soon as the stem is required to be inactive. The transitions are described graphically in Figure 5.16

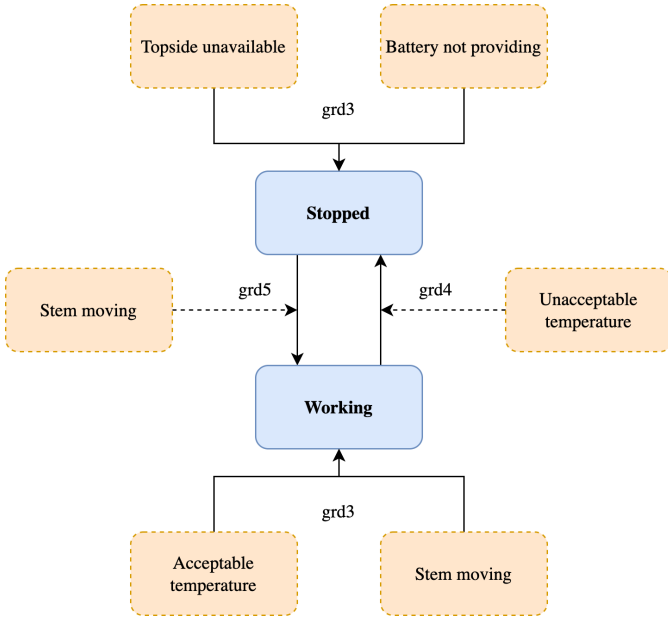


Figure 5.16: State transition diagram of SetMotorState

The new invariants and variables used in **as_mac4** can be seen in Figure 5.17, as well as their initialization.

```
MACHINE
  as_mac4
REFINES
  as_mac3
SEES
  as_ctx1
VARIABLES
  valve_position
  battery_state
  topside_supply
  battery_level
  pressure_level
  ESD_initiated
  safety_command
  stem_movement
  motor_state
  temperature
INVARIANTS
  inv1   :   motor_state ∈ MotorStates
  inv2   :   temperature ∈ TemperatureSensor
  inv3   :   temperature = Unacceptable ⟹ motor_state ≠ Working
  inv4   :   motor_state = Stopped ⟹ stem_movement = Inactive
EVENTS
  INITIALISATION   ≙
   extended
  STATUS
   ordinary
  BEGIN
   act1   :   battery_state ≔ {Charging}
   act2   :   topside_supply ≔ Available
   act3   :   battery_level ≔ 100
   act4   :   pressure_level ≔ Safe
   act5   :   ESD_initiated ≔ FALSE
   act6   :   safety_command ≔ {Idle}
   act7   :   valve_position ≔ Open
   act8   :   stem_movement ≔ Inactive
   act9   :   motor_state ≔ Stopped
   act10   :   temperature ≔ Acceptable
  END
```

Figure 5.17: as_mac4 refinement of initialisation

5.5.3 Complete overview

To gain a complete view of all the refinements done and what was added throughout the refinement process, including new safety constraints, an overview of the refined machines can be seen in Figure 5.18.

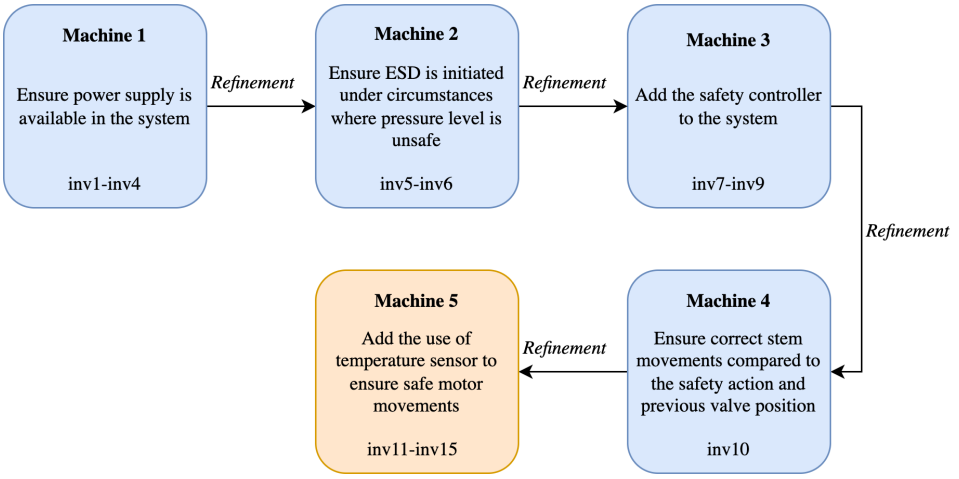


Figure 5.18: Machine overview

Total list of invariants:

InvariantID	Predicate
inv1	BatteryState $\subseteq \{FullyCharged, Depleted, Charging, Providing\}$
inv2	TopsideSupply $\in \{Available, Unavailable\}$
inv3	$0 <$ BatteryLevel $\wedge$BatteryLevel ≤ 100
inv4	TopsideSupply = $Unavailable \implies Providing \in$ BatteryState
inv5	PressureLevel $\in \{Safe, Unsafe\}$
inv6	ESDinitiated $\in BOOL$
inv7	SafetyCommand $\subseteq \{CloseValve, OpenValve, OverrideBMS, Idle\}$
inv8	ESDinitiated = TRUE $\implies CloseValve \in$ SafetyCommand
inv9	ValvePosition $\in \{Open, Closed\}$
inv10	StemMovement $\in \{Retracting, Extracting, Inactive\}$
inv11	MotorState $\in \{Stopped, Working\}$
inv12	Temperature $\in \{Acceptable, Unacceptable\}$
inv13	Temperature = $Unacceptable \implies$ MotorState $\neq Working$
inv14	MotorState = $Stopped \implies$ StemMovement $= Inactive$
inv15	ESDinitiated = FALSE $\implies CloseValve \notin$ SafetyCommand

Table 5.9: Complete list of invariants

The total formal model for the Actuation System can be seen in Appendix B.

5.6 Identification of loss scenarios

In Chapter 5, it was mentioned that analyzing loss scenarios can help identify the causes for UCAs to occur, why control actions might not be executed, and the improper execution

of control actions. These scenarios are essential for developing a thorough understanding of the safety hazards associated with a system, by describing causal factors that can lead to UCAs and hazards. However, the control action analysis served as a basis for the formal modelling in Section 5.4, and most of the unsafe control actions are handled through guards and invariants in the model. This means that when loss scenarios are identified, new constraints are formed to ensure they are taken care of in the software and modelled formally. Thus, loss scenarios are covered by the formal model and therefore not mentioned in a loss scenario analysis. This methodology is described in Section 4.4, and is developed by the author in response to a notable absence of comparable and readily accessible methods, yet still based on SE-STPA and STPA.

However, there are some UCAs that cannot be adequately validated through proof obligations, as mentioned in Section 4.4, and there are some UCAs from Table 5.7 which cannot be modelled properly using formal languages either. An example is UCA-11, which involves the motor stopping for too long, thereby rendering the motor incapable of starting as required. Such UCAs needs to be verified through testing, because the lack of experience with formal modelling and how to do this most reasonable in a mathematical sense. Test cases can be generated to specifically target these UCAs through carefully defined loss scenarios, akin to the traditional STPA approach, or new constraints and requirements might be developed after a thorough analysis. The methodology is seen in Figure 5.19.

Moreover, scenarios detectable via existing diagnostics are not considered in this analysis. For instance, system miscalibrations resulting from wear and tear — such as a discrepancy between feedback and actual pressure values in the well — could be identified by an alarm system. Since these alarms can be directly addressed by an on-site operator, they do not necessitate testing, or to be modelled formally.

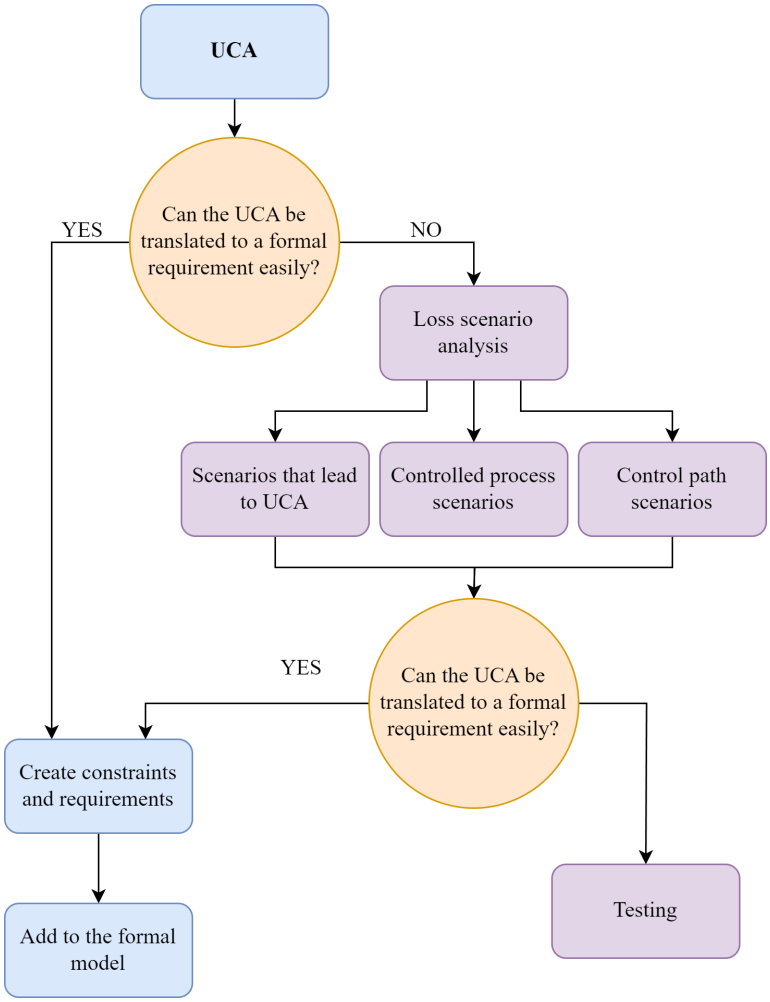


Figure 5.19: Scenario analysis methodology

5.6.1 Scenarios that lead to UCAs

In Table 5.7, the BMS, safety controller, and motor controller have been identified as the controllers associated with the all-electric actuation system. This part of the loss scenario analysis will investigate what could cause the controller to provide a control action which is unsafe, or not provide a control action when needed, which could lead to an unsafe situation.

As previously mentioned, the formal modelling eliminates many loss scenarios which otherwise should have been identified. To clarify this, some loss scenarios can be mentioned.

Loss scenario 1: While ESD is initiated, the motor controller fails to start the motor

because of flaws in the controller logic detecting the ESD [*UCA-1*]. Trouble can be detected in the refined event InitiateESD_safety.

Loss scenario 2: The phase in the motor is unbalanced, but the motor controller starts the motor due to a wrongful frequency in the three-phase inverter. This leads to a rise in temperature in the motor windings [*UCA-2*]. The temperature should be detected early enough for the state to be correctly set in SetMotorState.

Loss scenario 3: The pressure in the well is high, and the valve is required to stay close. However, the motor is starting such that the valve can open due to a wrongful process model, causing to loss of control and potential damage to equipment [*UCA-3*]. SetMotorState relies on the state of the stem movement and the temperature, and this troublesome behavior is caught using SetStemMovement.

Loss scenario 1-3 illustrates the implications of incorrect process models, which mislead the system into accepting a false representation of reality. Such scenarios should be handled during the proof obligations and requirement definitions. The modelled system should accurately represent the system's process model, and it should be consistent with its requirements and specifications. Formal verification is a technique to handle ambiguous behavior beforehand. Hence, if the system is described well enough and the concrete machine has taken every requirement into account, troubles regarding the process model can be neglected.

Loss scenario 4: The temperature in the motor windings is high because of increased friction in the system, but the motor is required to start before the temperature has reached an acceptable level, causing the motor equipment to wear out more quickly [*UCA-4*].

Since the safety controller has the highest executive authority, it may override the motor's temperature alarm, prioritizing the command to start the motor despite the hazardous conditions. Loss scenario 4 highlights the challenges posed by the hierarchical structure of the actuation system, where the safety controller, as seen in Figure 5.1, holds the highest executive authority. In these scenarios, even if the motor provides an alert indicating that the temperature is at a critical level, it may not receive the necessary attention due to the safety controller's prioritized command.

This scenario emphasizes the importance of carefully designing the interactions between different controllers in a hierarchical system, ensuring that critical alerts from lower-level components, such as the motor's temperature warning, are not overlooked or overridden by the higher-level controller's commands. By identifying loss scenario 4, a new safety requirement can be formed as:

- **SAF-9**: The safety controller cannot override motor start when the temperature is not adequate.

This safety requirement aligns with SAF-7 defined in Section 5.5.2, but is only considered in the motor logic in the formal model. It is important that this is considered in the logic for the controller with the highest executive power as well. SAF-9 can then be added to the formal model through a new refinement of the existing model in Appendix B.

Other design aspects in the all-electric actuation system which is not considered in the current formal model is possible delay. Delay in response from the controlled process can occur when it improperly receives a control action due to connection error or wiring error. This is considered to be loss scenarios related to the controlled path.

5.6.2 Control path scenarios

Control path scenarios are related to the second type of loss scenarios described in Figure 3.4, which results from control actions not being executed or not being executed correctly. Control action not being executed can be caused by delay or the command not being sent properly. Other loss scenarios in conjunction with the controlled path and the control actions identified in Table 5.7 for the all-electric actuation system can include:

- If the feedback providing motor diagnostics and/or battery diagnostics seen in Figure 5.1 fails, the system might operate on the wrong premises and behave erratically.
- The actuator, or motor, might not receive the commands sent by the motor controller or the safety controller. The command can either be manipulated because of trouble in the path, the actuator might not have received the command because of signal cut or the actuator just not responding to the command. This can cause the system to be unresponsive and not behave correctly. Insufficient well isolation might occur.
- The power can be lost during a transaction of a command, such that the command never reaches the destination it is supposed to.
- Environmental factors as humidity, temperature, vibration, or pressure can alter the commands and affect the performance of the controlled path.

These possibilities can be concretized into a single loss scenario:
Loss scenario 5: The safety controller has initiated an ESD due to high DH pressure, but the motor controller is delayed to start the motor by more than TBD minutes/hours due to connection error in the wiring. This causes the valve to not shut in time [*UCA-5*].

There are other possibilities to delay other than wiring error, such as latency in the motor controller, subsea noise, interference from other equipment or environmental factors from subsea conditions such as high pressure, corrosive elements or extreme temperatures. They could all be formulated into separate loss scenarios, but are spared here for convenience since it would result in multiple, somewhat identical loss scenarios. This is because most of the troubles related to the controlled path can be solved with an additional safety requirement:

- **SAF-10**: The safety controller should resend the command if control action is not executed within TBD minutes

This could be solved by adding a timer in the safety controller software if it is not added already, or adding additional software for the safety controller to detect this through available feedback diagnostics.

5.6.3 Controlled process scenarios

Even though the control actions are sent from a controller and received by the controlled process, they may not be executed properly by the component receiving the control action because of missing process inputs, environmental disturbances or component failures.

Component failures are often caused by debris and contamination on the controlled process. This is especially relevant if the component, i.e., the motor, has not been used for a long time. Such phenomena are hard to model using mathematical language and proof, and are not taken into the current modelled system. Such troubles are related to UCA-11 from Table 5.7, which states that stopping the motor for too long might yield the motor unable to start when needed. These can be resolved using partial stroke testing (PST). PST is a diagnostics procedure used to assess the functionality and performance of systems without disruption to the process because of complete shutdown for testing [60]. PST moves the valve, or the motor, approximately 10-20% of the operation range to verify that the component is operational without causing a total process interruption.

Possible scenarios are:

Loss scenario 6: Motor has been stopped for too long, and the sealing in the PMSM has failed, leading to dust and debris contamination on the components. This can cause corrosion, erosion, or other types of damage, leading to reduced performance or even failure of the motor [*UCA-11*].

Loss scenario 7: Motor has been stopped for too long, leading to the motor bearings drying out. This can lead to expensive repairs or motor not being able to start [*UCA-11*].

The same situations can occur for the battery. There could be corrosion or dirt on the terminal causing poor conductivity. A loss scenario can be formulated as:

Loss scenario 8: Topside PS is unavailable, and BMS orders the battery to provide energy to the valve actuation system. The battery is not able to provide energy due to corrosion or dirt on the terminal, causing poor conductivity [*UCA-22, UCA-24*].

These loss scenarios lead to additional requirements. Regarding the battery, the loss scenarios yields additional constraints in the BMS for the software which evaluates the safety of the battery, consisting of the SOH-algorithm and SOC-algorithm, described in

Section 2.5.1. The SOH-algorithm should include condition monitoring of the battery, which can predict whether the performance of the battery is affected because of corrosion. This is related to the functional requirement FUN-4: *The BMS is in charge of the battery SOH and SOC to ensure optimal performance and reliability of the battery*. When the functional requirements already exist, it can be further elaborated with environmental requirements, as seen in Figure 4.12. The additional environmental requirement can be:

- **ENV-20**: The BMS should identify signs of wear and corrosion of the battery through the SOH-algorithm.

The subsea motor and its components, such as sealing and bearings, should be designed to prevent dust, debris, and corrosion-related issues, even after extended periods of inactivity. However, it is not always a guarantee when environmental factors which cannot be controlled are present. Nevertheless, the motor controller receives diagnostics from the motor through feedback, as seen in Figure 5.1, and could monitor if the motor is not able to perform functions as desired. To ensure that the motor correctly monitors the behavior, environmental requirements, which are concerned with the structure of the system and its components, can be added. An environmental requirement regarding the motor controller can be:

- **ENV-21**: The motor controller should detect if the motor lacks proper performance.

If ENV-21 is not incorporated into the software, loss scenario 6 and loss scenario 7 can be detected through PST occasionally. Especially should the PST be performed after a period of inactivity to ensure that everything is intact even though the diagnostics says so. A PST can be performed as such:

Possible test scenario for loss scenario 6 and 7: PST after motor has been stopped for an extended period.

1. Stop motor
2. Start timer for partial stroke test
3. Send start signal to motor
4. Reset start motor command after reaching a limited portion of operation range.
5. Evaluate if motor works as needed

If there should be trouble with the motor, maintenance is needed.

The examination of loss scenarios and their consequences has revealed potential weaknesses in the formal model, emphasizing the need for continuous refinement of the model based on the findings of the safety analysis. However, analyzing loss scenarios has provided insights into the development of comprehensive test scenarios that can effectively address the identified loss scenarios, allowing for a more efficient and targeted testing approach. By focusing on the most critical safety hazards, system developers can prioritize their efforts and ensure that the all-electric actuation system meets the necessary safety standards. The analysis has highlighted the importance of understanding the system components, their interactions, and potential failure modes to ensure the development of a safe and reliable system.

Moving forward, it is essential to continue refining the system design, formal model, and requirements based on the findings of this analysis. For a complete analysis, the loss scenarios should not be present, and every requirement should be tested to be adequate through a digital twin. Additionally, incorporating redundancy and fault tolerance measures can further enhance the robustness and reliability of the all-electric actuation system. By addressing potential issues, the development team can ensure the successful deployment of a safe, reliable, and efficient all-electric actuation system for safety valves in subsea applications.

6

Discussion

This chapter discusses the benefits of combining formal verification with STPA, also highlighting the limitations of the SE-STPA-mod method presented in this thesis. To gain a complete overview of the pros and cons of combining STPA with formal verification, a summary, and discussion of how the new method differs from the previous conducted STPA is provided. Furthermore, the discussion extends to address the limitations in the selected formal language for this thesis, Event-B.

6.1 Comparison to previous conducted STPA

An STPA was previously conducted by the author for the all-electric actuation system for an in-depth study in the course TTK4550 [1]. The main objective of the specialization project was to gain an understanding of safety demonstrations, particularly the use of digital twins and STPA. For this thesis, the SE-STPA-mod was employed as the preferred method of analysis, which differs from the STPA approach used in the previous study because of the integration of formal methods. SE-STPA-mod was created for the purpose of this thesis, to fit the incorporation of formal verification and STPA to a better extent, and is based on the security-enchanced STPA (SE-STPA), which aimed to remove ambiguities from the STPA in cybersecurity. How this methodology differs from STPA and SE-STPA can be seen in Figure 6.1

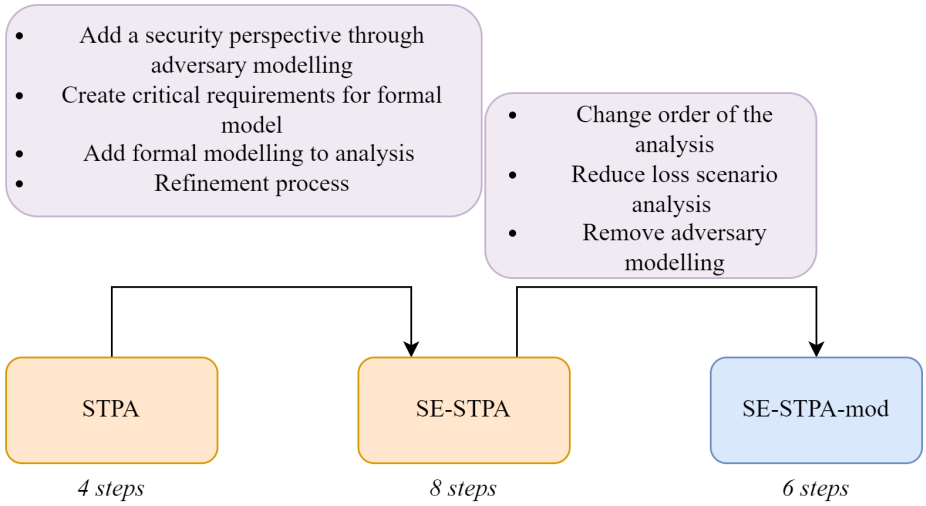


Figure 6.1: Differences in safety analysis method

Although the analyses were conducted on the same all-electric actuation system, the approaches and scope of research differed significantly. In this thesis, previous knowledge of the system was utilized to provide a deeper understanding of the total system, while further research on the system components including the motor, the BMS, and the control algorithms used to regulate the battery supply, as well as motor performance were given more attention in this thesis. Insight in how the different components were controlled were not given much attention in the previous study, which was primarily focused on gaining a general understanding of the safety analysis tool. However, how the total usage of components affects the system as a whole is important for the STPA for this thesis.

The differences in the approaches and scope of research resulted in a more comprehensive analysis of the all-electric actuation system in this thesis. By delving deeper into the control mechanisms of the system, the control actions were more detailed to be able to translate it into a formal model. This was necessary to create more precise specifications of the requirements to be able to formally model the system. Additionally, the loss scenarios found in the analysis were no longer superficial as to the previous study, where illustrating the purpose of the loss scenarios and how they can occur were important.

Furthermore, the loss scenario analysis did not have to be conducted to full extent for this thesis. A significant portion of the system functionality was validated through the use of guards and invariants, which reduced the necessity for a comprehensive loss scenario analysis. Despite this, the analysis was carried out to some extent, but rather than always yielding loss scenarios that needed testing, it sometimes led to the discovery of new requirements for the formal model.

Should the scenario analysis be carried out as described in both SE-STPA and tradi-

tional STPA, the list of loss scenarios would be more extensive. This is because each UCA that cannot be conclusively addressed in the software must be tested and validated. Ensuring the software safely handles UCAs is critical, and the safety demonstration must guarantee that UCAs will not result in unsafe situations during normal operations. This level of assurance, however, can be provided by formal methods.

Overall, the SE-STPA-mod approach used in this thesis provides a more comprehensive and detailed analysis of the all-electric actuation system compared to the previous, traditional STPA approach from the specialization project [1]. By conducting a more thorough research of the system components and control algorithms, the formal model specifications are more precise, which can help to improve the safety and reliability of the system in real-world applications.

6.2 Combining formal modelling and STPA

STPA is a widely used qualitative safety analysis technique that relies on expert judgement and experience to identify potential hazards and assess risks. While it provides a systematic approach to safety analysis and is one of the few safety analysis methods that includes error in software and controllers as a possibility, it is also prone to bias and subjectivity. This has been proven to be a limitation in this thesis, because the analysis has been carried out by the author with help from the supervisor and the co-supervisor, and not a team of experts. However, this is an aspect that formal modelling can eliminate, by reducing the use of expert judgement and focus on the system qualifications directly.

One challenge of employing STPA is its time-consuming nature, as new hazards and risks may emerge throughout the analysis, necessitating multiple iterations. While this iterative approach effectively identifies all possible hazards or risks and is a crucial aspect of STPA, it may not be suitable for rare, unexpected, and difficult-to-anticipate events. Additionally, combining it with formal methods do not reduce the time spent conducting the analysis in SE-STPA-mod, since formal verification also is quite time-intensive. Nevertheless, it is effective in identifying all possible hazards or risks if they are possible to anticipate, and it lays a great foundation for requirement generation for formal verification. This is because the better and more detailed the risk, hazards, and constraints are, the easier it is to convert it to a formal model. But, even though STPA provides a structured approach for identifying potential hazards and loss scenarios though multiple iterations, there is no guarantee that the system behavior described is correct. Formal modelling offers a way to verify that the system will perform as intended by mathematically modelling the system and analyzing its behavior, and is also the reason for the creation of SE-STPA-mod.

When using formal verification with STPA, there is a shift in the way the process is approached. Instead of simply identifying potential hazards and control structures, maybe from previous experience or with help from experts in the field, formal modelling requires a deeper understanding of the system and its available states. This requires a more in-depth analysis of each system element and its interactions, as mentioned when comparing it to the previously conducted STPA. Additionally, a more rigorous approach to requirement

identification and description is needed to ensure that every state transition is traceable and easily maintained, something that STPA provides because of the well-defined approach in the analysis, as well as defining the functional, environmental and safety requirements of the system by researching the intended behavior of the system through standards and regulations for the formal model. The rigorous approach also helps to eliminate possible inconsistencies that STPA do not address. The formal approach combined with the STPA, illustrating how the methods actually are used, can be seen in Figure 6.2.

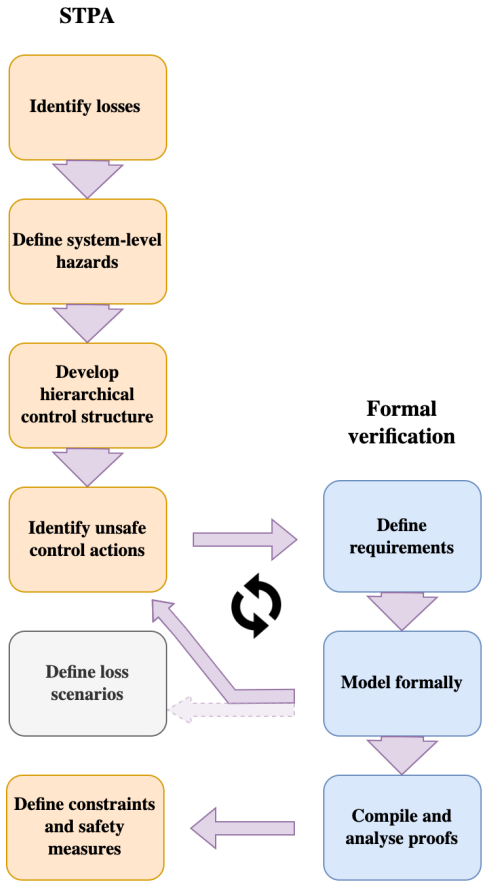


Figure 6.2: STPA approach

Formal verification makes it possible to verify that the system will perform as intended and identify any potential issues or weaknesses in the system design. This can greatly improve the safety of the system and reduce the risk of accidents or losses because of the need to ensure that the states are correctly set, and that no invariants are violated through proof obligations. Incorporating formal verification into safety analysis can therefore provide a higher degree of confidence in the system's behavior and reduce the risk of potential safety hazards. However, one potential downside is that it can also provide a false sense

of safety. While the developer may feel that the system has been thoroughly described and mathematically modelled, and that all proof obligations have been discharged, there is still a risk of errors in both the modelling and the system understanding. This is because that even with formal proofs in place, there is still a chance that errors or oversights in the modelling process, and the UCA analysis from the STPA, can lead to a failure of the system to perform as intended. Even though the system as modelled is considered safe after the formal proofs, there could be unforeseen trouble which might lie in the system description, or phenomena that cannot be described beforehand. It is important to recognize that while formal verification can provide a greater sense of safety, it is not foolproof.

A great way to detect these insecurities is through loss scenario identification. This might yield additional system design, as well as new software constraints. Important aspects such as controller priority and transitions which were not well thought through might be identified, such as for the case in loss scenario 7 and 8 in Section 5.6.1. Otherwise, the loss scenario analysis is not done in a full extent due to the assumption that potential causal factors and control action that lead to ineffective or inadequate control of the process is already considered in the formal model. Hence, for only some potential weaknesses in the formal model descriptions, loss scenario analysis is done, and therefore not given that much attention as shown in Figure 6.2.

It is important to supplement formal verification with other methods of analysis and testing for verifying the behavior, such as simulation, and testing on hardware. This multifaceted approach can help to identify potential issues that may have been missed during the formal verification process and identification of requirements through the UCA analysis, and can provide a more comprehensive understanding of the system's behavior. Ultimately, by using a combination of methods, the safety of complex systems can be improved. For this thesis, no method for testing through simulation were in place yet, so how well this proves to work is not definite, and therefore not added to the approach in Figure 6.2.

Using formal verification with STPA was quite time-consuming, although the entire system behavior was described in great detail prior to beginning the formal modelling process. Despite simplifications made to the system from the overview depicted in Figure 2.6 to the modelled system in Figure 5.1, the modelling process was still difficult. Specifying the properties to be verified is also a challenging task that requires a detailed understanding of the system's requirements and safety goals. Another challenge that arose was the multitude of mathematical proofs and expressions available to define the same behavior, and that different formulations might pass a proof while others— while still describing the same guard— did not. This proved to be challenging to navigate with the existing amount of experience in formal verification.

6.3 Choice of Event-B for formal modelling

While modelling in Event-B, there were a lot of challenges with maintaining the invariants due to the new way of development for software. The states of the system had to pass strict

proofs such as equality proof, invariant preservation and feasibility proof, which eliminates unsafe or loose definitions in the system. However, sometimes the proof obligations were hard to discharge because of ambiguous details and an old platform. This led to the guard or the event to be simpler than anticipated.

Event-B is complex and powerful, and hence also challenging to learn and use effectively. There are also a lot of extensions to be added to the platform which could ease the transition to use Event-B, but they were not easy to find. Nonetheless, the bigger and complex the system, the more difficult it is to create a clear model using this language. Even so, Event-B is a language that places a strong emphasis on gradual refinement, meaning that it can be difficult to create highly abstract models that can be easily understood. Hence, starting with a simpler statement than anticipated and further refining by adding more details is the preferred method of Event-B.

Even though Rodin provides support for formal verification of Event-B models, it may not be so suitable for complex verification tasks, or systems which contain certain phenomena which are difficult to model mathematically. The lack of visualization support of the model created can also make it difficult to analyze complex models. The only visual aid in Rodin is the use of the Event-B Explorer seen in Figure 6.3 and the generated code as seen in Figure 5.17.

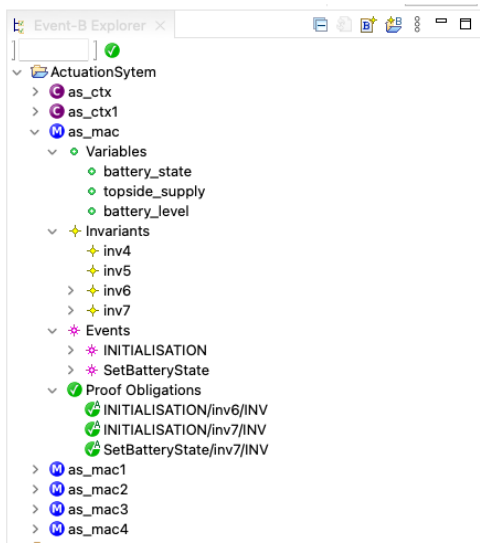


Figure 6.3: Event-B Explorer in Rodin

Additionally, there is no obvious solution for testing if the system benefits from redundancy, or if the system is affected by phenomena which are difficult to model mathematically, which can be a limitation in some verification tasks. This is also a limitation by using Event-B, since proving the correctness of the model relies on the developer being able to model the properties properly. In some cases, it may not be possible to prove cer-

tain properties in a system due to limitations in the underlying logic, or the understanding on the use of mathematical logic.

Nevertheless, modelling the system in Event-B has made it possible to create requirements and eliminate design errors early on, which is beneficial despite the downsides mentioned. The way of thinking of the system and the control actions in terms of mathematical logic and using a systematic formal approach has proven to give greater insight to the system specifications in detail.

These challenges highlight the importance of having a deep understanding of both the system and the formal language being used, and the need for continuous improvement and learning in the field of formal verification.

7

Conclusion and further work

7.1 Conclusion

The subsea oil and gas industry is developing all-electric safety valve actuation systems to replace traditional electro-hydraulic solutions, with a goal of optimizing diagnostics and lessen environmental impact. One of the notable design changes is replacing the spring-assisted actuation of hydraulically operated safety valves with a motor and gear powered by a battery. The battery becomes critical in ensuring that the valve can close on demand under all foreseeable situations, and it is a requirement from the authorities that the suppliers of such systems can demonstrate that the performance is as good as or better than the traditional solutions. The research-based innovation center in subsea production and processing (SUBPRO) at NTNU has an ongoing research activity to develop a digital representation, or twin, of the all-electric valve actuation system, to test the novel control strategies and diagnostic functions. The objective of this master thesis has been to develop and test a method to identify new and possible inconsistencies in existing requirements that the all-electric actuation system must satisfy to be safe.

A System-Theoretic Process Analysis (STPA) was introduced as a method for safety analysis, to identify possible requirements necessary for ensuring appropriate safety assurance of the all-electric actuation system. STPA employs a systematic approach suitable for software-intensive systems to uncover risks and potential unsafe control actions due to interactions between system components, which are otherwise challenging to detect with traditional safety methods. However, a difficulty in the STPA approach can be to identify potential inconsistencies between requirements. To address this, formal verification is incorporated in the STPA.

The integration of formal verification provides for a more comprehensive safety assessment by identifying safety risks and verifying the correct implementation of safety requirements, forming a new method, SE-STPA-mod. SE-STPA-mod is an adaptation of an existing method, based on enhancing security with formal verification and adversary

modelling for cybersecurity, called SE-STPA, which also attempts to address the challenge with STPA.

SE-STPA-mod is a contribution of this thesis which was beneficial to identify functional requirements, as well as safety requirements, for the all-electric actuation system. By including a mathematically modelled, correct-by-construction formal model, it enhances system behavior assurance, reducing potential hazards, and assists in identifying inconsistencies in early development. Despite its effectiveness, to cope with growing complexity of systems, and the advancement of safety-critical systems, improvements in visualization tools and system behavior validation are required, strengthening the integrated approach.

7.2 Further work

Although formal verification presented some challenges during the development of the all-electric actuation system model, its advantages outweigh the difficulties. There were parts of the system that proved to be hard to model using the Event-B language, but the benefits of including these in the model would be significant. However, a variety of other formal languages could potentially offer a better fit for this system. Event-B was chosen as the preferred language due to its previous use in the SE-STPA method. Articles employing SE-STPA for safety analysis consistently utilized Event-B, leading to the assumption that it would be a suitable choice for the all-electric actuation system. Moving forward, it would be beneficial to conduct an in-depth investigation of the most fitting formal language for this system. This should ideally include a comprehensive depiction of the system's behavior, factoring in redundancy. This investigation should be a priority for future research.

Building a simulator to validate the formal model's functionality could also be beneficial. Extensions to Rodin could be utilized to create an executable software representation of the formal model, which then could be examined through a simulator. It was noted that a limitation in Rodin lies in its lack of graphical visualization and absence of a function to run the model and observe the output. The use of a simulator would eliminate this limitation, and also provide a more intuitive way of understanding the formal model. Simulations increases the confidence of the formal model, providing the formal verification process with an additional layer of validation. Moreover, the simulations can also be done in conjunction with the digital twin, which is in development for this system through SUBPRO.

One limitation of the STPA conducted in this thesis is that the discussions regarding the findings only have been with the thesis supervisors. However, the incorporation of formal verification serves to mitigate this limitation, enhancing the validity of the analysis regardless. Despite this mitigation with formal verification, it is advisable to replicate this analysis in a team setting, for instance during a workshop, using this thesis as the foundation for further exploration and discussion.

Furthermore, the loss scenario analysis which was conducted for some ambiguities not included in the model resulted in new requirements for the system. These requirements should be incorporated into the model during the next refinement of the system, and it should be repeated until there are no ambiguities left. All the requirements should then be tested using a digital twin, or a simulator developed for the all-electric actuation system software.

Bibliography

[1] S. A. Radi, "Using safety analysis to generate test scenarios of an all-electric control system in subsea wells," *NTNU*, Dec. 2022, Specialization project.

[2] M. Jastram, *RODIN USER'S HANDBOOK: covers rodin*.
Place of publication not identified: CREATESPACE, 2014, OCLC: 1014347466, ISBN: 978-1-4954-3814-1.

[3] D. Leon, S. Imle, and M. Glaser, "Wear-optimized partial stroke test for all-electric actuation systems," *Gas Science and Engineering*, vol. 111, Jan. 2023. DOI: 10.1016/j.jgsce.2023.204875. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S2949908923000031 (visited on Apr. 12, 2023).

[4] T. Myhrvold and M. van der Meulen, *Demonstrating safety of software-dependent systems - With examples from subsea technology*. Høvik, Norway: DNV AS, 2022, ISBN: 978-82-515-0323-5.

[5] International Electrotechnical Commission, *IEC 61508-6: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3*, 2nd ed.
Geneva, Switzerland: Norwegian electrotechnical publication, Apr. 2010, vol. NEK IEC 61508-6:2010.

[6] SUBPRO NTNU. "SUBPRO - subsea production and processing - NTNU." (2023), [Online]. Available: https://www.ntnu.edu/subpro/ (visited on Apr. 12, 2023).

[7] International Electrotechnical Commission, *IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems*, 2nd ed.
Norwegian electrotechnical publication, Apr. 2010, ISBN: 978-2-88910-530-4.

[8] K. Sotoodeh, "A review on subsea process and valve technology,"
Marine Systems & Ocean Technology, vol. 14, no. 4, pp. 210–219, Dec. 1, 2019, ISSN: 2199-4749. DOI: 10.1007/s40868-019-00061-4. [Online]. Available: https://doi.org/10.1007/s40868-019-00061-4 (visited on Mar. 3, 2023).

[9] International Electrotechnical Commission,
IEC 61511: Safety industrumented systems for the process industry sector, 2.1. Aug. 2017, vol. NEK IEC 61511-1:2016 +A1:2017 CSV, ISBN: 978-2-8322-4752-5.

[10] NORSOK, *NORSOK S-001: 2020 Technical safety*. NORSOK Standard, 2021, vol. NORSOK S-001:2020+AC:2021 (en).

[11] Norsk olje og gass, *NOG 070: APPLICATION OF IEC 61508 AND IEC 61511 IN THE NORWEGIAN PETROLEUM INDUSTRY*, 3rd ed. 2018.

[12] Standards Norway. "NORSOK standards — standard.no." (Dec. 2021), [Online]. Available: `https://www.standard.no/en/sectors/energi-og-klima/petroleum/norsok-standards/#.Y5rs_nZKj-g` (visited on Mar. 3, 2023).

[13] NORSOK, *NORSOK U-001: Subsea production systems*.
NORSOK Standard, Dec. 2021, vol. NORSOK U-001:2021 (en).

[14] European Committee For Standardization,
ISO 13628-4: Petroleum and natural gas industries - Design and operation of subsea production systems - Part 4: Subsea wellhead and tree equipment.
Dec. 2010, vol. NS-EN ISO 13628-4:2010.

[15] European Committee For Standardization,
ISO 13628-6: Petroleum and natural gas industries - Design and operation of subsea production systems - Part 6: Subsea production control systems. Sep. 2006, vol. ISO 13628-6:2006.

[16] The Petroleum Safety Authority Norway. "§54: Christmas tree and wellhead." (Jan. 2014), [Online]. Available:
`https://www.ptil.no/en/regulations/all-acts/the-facilities-regulations3/VIII/54/` (visited on May 19, 2023).

[17] Norwegian Electrotechnical Standard, *IEC 62619: Secondary cells and batteries containing alkaline or other non-acid electrolytes*. May 2022,
vol. NEK EN IEC 62619:2022, ISBN: 978-2-8322-2497-7.

[18] European Committee For Standardization,
ISO 13702: Petroleum and natural gas industries - Control and mitigation of fires and explosions on offshore production installations - Requirements and guidelines, 2nd ed. Geneva, Switzerland, Aug. 2015, vol. NS-EN ISO 13702:2015.

[19] NORSOK, *NORSOK S-003: Environmental care*, 2nd ed. Oct. 2017,
vol. NORSOK S-003:2017.

[20] International Electrotechnical Commission,
IEC 61508-1: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements, 2nd ed.
Geneva, Switzerland: Norwegian electrotechnical publication, Apr. 2010,
vol. NEK IEC 61508-1:2010.

[21] T. Henanger, G. Muller, and L. Picacia, "Managing installation tolerances through system modeling and tolerance budgeting,"
INCOSE International Symposium, vol. 26, no. 1, pp. 1176–1191, 2016, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.2334-5837.2016.00219.x, ISSN: 2334-5837.
DOI: 10.1002/j.2334-5837.2016.00219.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2334-5837.2016.00219.x (visited on Apr. 19, 2023).

[22] The Petroleum Safety Authority Norway. "§53: Equipment for completion and well flow." (Jan. 2014), [Online]. Available: https://www.ptil.no/en/regulations/all-acts/the-facilities-regulations3/VIII/53/ (visited on May 19, 2023).

[23] L. Wang, X. Wang, H. Lizhang, P. Jia, F. Yun, and H. Wang, "Design and reliability analysis of the electrical control system of the subsea control module," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 233, no. 6, pp. 720–733, Jul. 1, 2019, Publisher: IMECHE, ISSN: 0959-6518. DOI: 10.1177/0959651818821199. [Online]. Available: https://doi.org/10.1177/0959651818821199 (visited on Mar. 6, 2023).

[24] C. Mahler, M. Glaser, S. Schoch, *et al.*,
"Safety capability of an all-electric production system,"
in *Day 3 Wed, May 08, 2019*, Houston, Texas: OTC, Apr. 26, 2019, D031S041R003. DOI: 10.4043/29472-MS. [Online]. Available: https://onepetro.org/OTCONF/proceedings/19OTC/3-19OTC/Houston,%20Texas/181387 (visited on Apr. 12, 2023).

[25] International Electrotechnical Commission,
IEC 61508-3: Functional safety of electrical/electronic/ programmable electronic safety-related systems - Part 3: Software Requirements, 2nd ed.
Geneva, Switzerland: Norwegian electrotechnical publication, Apr. 2010, vol. NEK IEC 61508-3:2010.

[26] International Electrotechnical Commission,
IEC 61508-2: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: Requirements for electrical/electric/programmable electronic safety-related systems, 2nd ed.
Geneva, Switzerland: Norwegian electrotechnical publication, Apr. 2010, vol. NEK IEC 61508-2:2010.

[27] S. Imle, T. Winter, J. Popp, M. Glaser, and B. Bertsche,
"Safety and reliability analysis of an actuation system,"
in *Proceedings of the 29th European Safety and Reliability Conference (ESREL)*, Research Publishing Services, 2019, pp. 3675–3682, ISBN: 978-981-11-2724-3. DOI: 10.3850/978-981-11-2724-3_0885-cd. [Online]. Available: http://rpsonline.com.sg/proceedings/9789811127243/html/0885.xml (visited on Mar. 6, 2023).

[28] W. Li, M. Rentemeister, J. Badeda, D. Jöst, D. Schulte, and D. U. Sauer, "Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation," *Journal of Energy Storage*, vol. 30, p. 101 557, Aug. 1, 2020, ISSN: 2352-152X. DOI: 10.1016/j.est.2020.101557. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352152X20308495 (visited on Apr. 12, 2023).

[29] M. Köder, M. Loos, T. Winter, and M. Glaser, "Online large signal EIS to predict the LFP cell state of health," *Hochschule Aalen*, 2023.

[30] L. Lu, X. Han, J. Li, J. Hua, and M. Ouyang, "A review on the key issues for lithium-ion battery management in electric vehicles," *Journal of Power Sources*, vol. 226, pp. 272–288, Mar. 15, 2013, ISSN: 0378-7753. DOI: 10.1016/j.jpowsour.2012.10.060. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378775312016163 (visited on Apr. 13, 2023).

[31] M.-K. Tran and M. Fowler, "A review of lithium-ion battery fault diagnostic algorithms: Current progress and future challenges," *Algorithms*, vol. 13, no. 3, p. 62, Mar. 8, 2020, ISSN: 1999-4893. DOI: 10.3390/a13030062. [Online]. Available: https://www.mdpi.com/1999-4893/13/3/62 (visited on Apr. 13, 2023).

[32] A. Barré, B. Deguilhem, S. Grolleau, M. Gérard, F. Suard, and D. Riu, "A review on lithium-ion battery ageing mechanisms and estimations for automotive applications — elsevier enhanced reader," *Jornal of Power Sources*, Jun. 2013. DOI: 10.1016/j.jpowsour.2013.05.040. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0378775313008185?via%3Dihub (visited on Apr. 13, 2023).

[33] V. M. Bida, D. V. Samokhvalov, and F. S. Al-Mahturi, "PMSM vector control techniques — a survey," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Jan. 2018, pp. 577–581. DOI: 10.1109/EIConRus.2018.8317164.

[34] J. Lemmens, P. Vanassche, and J. Driesen, "PMSM drive current and voltage limiting as a constraint optimal control problem," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 3, no. 2, pp. 326–338, Jun. 2015, Conference Name: IEEE Journal of Emerging and Selected Topics in Power Electronics, ISSN: 2168-6785. DOI: 10.1109/JESTPE.2014.2321111.

[35] W. Young and N. G. Leveson, "An integrated approach to safety and security based on systems theory," *Communications of the ACM*, vol. 57, no. 2, pp. 31–35, Feb. 2014, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/2556938. [Online]. Available: https://dl.acm.org/doi/10.1145/2556938 (visited on Jan. 30, 2023).

[36] J. Thomas, J. Sgueglia, D. Suo, N. Leveson, M. Vernacchia, and P. Sundaram, "An integrated approach to requirements development and hazard analysis," presented at the SAE 2015 World Congress & Exhibition, Apr. 14, 2015, pp. 2015–01–0274. DOI: 10.4271/2015-01-0274. [Online]. Available: https://www.sae.org/content/2015-01-0274/ (visited on Feb. 15, 2023).

[37] N. Leveson, "STPA handbook," Mar. 2018.

[38] "Definition of LOSS," Merriam-Webster. (Apr. 9, 2023), [Online]. Available: https://www.merriam-webster.com/dictionary/loss (visited on Apr. 20, 2023).

[39] G. Howard, M. Butler, J. Colley, and V. Sassone, "A methodology for assuring the safety and security of critical infrastructure based on STPA and event-b," *Int. J. Critical Computer-Based Systems*, vol. 9, Mar. 2019, ISSN: 1757-8779.

[40] N. K. Singh, *Using Event-B for Critical Device Software Systems*. London: Springer London, 2013, ISBN: 978-1-4471-5259-0 978-1-4471-5260-6. DOI: 10.1007/978-1-4471-5260-6. [Online]. Available: https://link.springer.com/10.1007/978-1-4471-5260-6 (visited on Jan. 25, 2023).

[41] International Electrotechnical Commission, *IEC 61508-7: Functional safety of electrical/electronic/ programmable electronic safety-related systems - Part 7: Overview of techniques and measures*, 2nd ed. Geneva, Switzerland, Apr. 2010, vol. NEK IEC 61508-7:2010, ISBN: 978-2-88910-530-4.

[42] J.-R. Abrial, *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, May 13, 2010, 613 pp., Google-Books-ID: 23UgAwAAQBAJ, ISBN: 978-1-139-64397-9.

[43] E. Seligman, T. Schubert, and M. V. A. K. Kumar, "Chapter 1 - formal verification: From dreams to reality," in *Formal Verification*, E. Seligman, T. Schubert, and M. V. A. K. Kumar, Eds., Boston: Morgan Kaufmann, Jan. 1, 2015, pp. 1–22, ISBN: 978-0-12-800727-3. DOI: 10.1016/B978-0-12-800727-3.00001-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128007273000010 (visited on Jan. 25, 2023).

[44] D. Dghaym, T. S. Hoang, S. R. Turnock, M. Butler, J. Downes, and B. Pritchard, "An STPA-based formal composition framework for trustworthy autonomous maritime systems," *Safety Science*, vol. 136, p. 105 139, Apr. 1, 2021, ISSN: 0925-7535. DOI: 10.1016/j.ssci.2020.105139. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925753520305348 (visited on Jan. 16, 2023).

[45] E. M. Clarke and J. M. Wing, "Formal methods: State of the art and future directions," *ACM Computing Surveys*, vol. 28, no. 4, pp. 626–643, 1996, ISSN: 0360-0300. DOI: 10.1145/242223.242257. [Online]. Available: https://doi.org/10.1145/242223.242257 (visited on Jan. 23, 2023).

[46] A. Romanovsky and M. Thomas, Eds.,
Industrial deployment of system engineering methods, OCLC: ocn827083547, Berlin: Springer, 2013, 259 pp., ISBN: 978-3-642-33169-5.

[47] T. Lecomte, T. Servat, and G. Pouzancre, "Formal methods in safety-critical railway systems," *ClearSy*,

[48] W. Su, J.-R. Abrial, and H. Zhu, "Formalizing hybrid systems with event-b and the rodin platform," *Science of Computer Programming*, Abstract State Machines, Alloy, B, VDM, and Z, vol. 94, pp. 164–202, Nov. 15, 2014, ISSN: 0167-6423. DOI: 10.1016/j.scico.2014.04.015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642314002482 (visited on Feb. 9, 2023).

[49] "Event-b.org," Home of Event-B and the Rodin Platform. (2018), [Online]. Available: http://www.event-b.org/ (visited on Feb. 16, 2023).

[50] Roscoe, A.W, *The Theory and Practice of Concurrency*. Pearson, Apr. 2005.

[51] R. Westergaard, *utblåsning*, in *Store norske leksikon*, Sep. 19, 2022. [Online]. Available: https://snl.no/utbl%C3%A5sning (visited on Apr. 21, 2023).

[52] Norwegian Oil and Gas Association, *Application of IEC 61508 and IEC 61511 in the norwegian petroleum industry (recommended SIL requirements)*, Jun. 2018. (visited on Mar. 6, 2023).

[53] The Petroleum Safety Authority Norway. "§33: Emergency shutdown system." (Jan. 2018), [Online]. Available: https://www.ptil.no/en/regulations/all-acts/the-facilities-regulations3/V/33/ (visited on Mar. 29, 2023).

[54] European Committee For Standardization, *ISO 13849-1: Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design*. Dec. 2015, vol. NS-EN ISO 13849-1:2015.

[55] The Petroleum Safety Authority Norway. "§55: Planning." (Jan. 2011), [Online]. Available: https://www.ptil.no/en/regulations/all-acts/technical-and-operational-regulations3/VIII/55/ (visited on Mar. 29, 2023).

[56] The Petroleum Safety Authority Norway. "§15: Electrical installations." (Jan. 2020), [Online]. Available: https://www.ptil.no/en/regulations/all-acts/technical-and-operational-regulations3/III/15/ (visited on Mar. 29, 2023).

[57] The Petroleum Safety Authority Norway. "§38: Emergency power and emergency lightning." (Jan. 2020), [Online]. Available: https://www.ptil.no/en/regulations/all-acts/technical-and-operational-regulations3/V/38/ (visited on Jun. 29, 2023).

[58] The Petroleum Safety Authority Norway. "§8: Safety functions." (Jan. 2014), [Online]. Available: https://www.ptil.no/en/regulations/all-acts/the-facilities-regulations3/II/8/ (visited on Jun. 29, 2023).

[59] Event-B and Rodin Documentation Wiki. “B2latex - event-b.” (Jul. 2016), [Online]. Available: https://wiki.event-b.org/index.php/B2Latex (visited on May 25, 2023).

[60] M. A. Lundteigen and M. Rausand, “The effect of partial stroke testing on the reliability of safety valves,” *Department of Production and Quality Engieering*, Nov. 2007.

[61] S. J. Rai, “Control of a permanent magnet synchronous motor (PMSM) with constraints,” *Norwegian University of Science and Technology*, Jun. 2017.

[62] S. Chattopadhyay, M. Mitra, and S. Sengupta, “Clarke and park transform,” in *Electric Power Quality*, ser. Power Systems, S. Chattopadhyay, M. Mitra, and S. Sengupta, Eds., Dordrecht: Springer Netherlands, 2011, pp. 89–96, ISBN: 978-94-007-0635-4. DOI: 10.1007/978-94-007-0635-4_12. [Online]. Available: https://doi.org/10.1007/978-94-007-0635-4_12 (visited on Feb. 6, 2023).

[63] R. K. Sharma, V. Sanadhya, L. Behera, and S. Bhattacharya, “Vector control of a permanent magnet synchronous motor,” in *2008 Annual IEEE India Conference*, ISSN: 2325-9418, vol. 1, Dec. 2008, pp. 81–86. DOI: 10.1109/INDCON.2008.4768805.

[64] Microsemi, “Park, inverse park and clarke, inverse clarke transformations MSS software implementations user guide,” [Online]. Available: https://www.microsemi.com/document-portal/doc_view/132799-park-inverse-park-and-clarke-inverse-clarke-transformations-mss-software-implementation-user-guide.

[65] L. Wang, *PID and predictive control of electrical drives and power converters using MATLAB/Simulink*. Solaris South Tower, Singapore: IEEE, Wiley, 2015, 343 pp., ISBN: 978-1-118-33944-2.

[66] M. S. Merzoug and F. Naceri, “Comparison of field-oriented control and direct torque control for permanent magnet synchronous motor (PMSM),” vol. World Academy of Science, Engineering and Technology, no. 21, 2008.

[67] N. P. Quang and J.-A. Dittrich, *Vector control of three-phase AC machines: system development in the practice* (Power Systems), 2. ed. Berlin Heidelberg: Springer, 2015, 364 pp., ISBN: 978-3-662-46914-9.

[68] B. Adhavan, A. Kuppuswamy, G. Jayabaskaran, and V. Jagannathan, “Field oriented control of permanent magnet synchronous motor (PMSM) using fuzzy logic controller,” in *2011 IEEE Recent Advances in Intelligent Computational Systems*, Sep. 2011, pp. 587–592. DOI: 10.1109/RAICS.2011.6069379.

[69] A. Abele, *Developing a digital twin for safety demonstration*, Jan. 2023.

Appendix

A Emergency shutdown system

Emergency shutdown(ESD) description is retrieved from the specialization project "Using Safety Analysis to generate test scenarios of an All-Electric Control System in Subsea Wells" [1] in conjunction with the subject TTK4550. Note that this section was in actuality subsection 2.3. in the original report, but added here as a standalone chapter for convenience. This is because ESD is mentioned often throughout the thesis, and some sort of understanding of ESD is beneficial to the safety analysis, but not necessary to understand it in depth. Hence, it is just recommended reading from the previous specialization project.

Principles of safety design for offshore oil and gas facilities are described in NORSOK S-001 [10], and a proposed safety barrier mentioned in this standard is an Emergency Shutdown system, further referred to as an ESD system. ESD is a safety system or a procedure, which are thoroughly described in NOG 070 [11] as well. The purpose of an ESD system according to NORSOK S-001 is to prevent escalation of conditions which are not considered regular for the production into major hazardous events, and is widely used for offshore facilities and industry. The ESD system should also limit the extent of any such event if it occurs by shutting down the system, as mentioned in §33 [53]. An ESD system typically includes sensors, logic solvers, and actuators that can detect an emergency condition and automatically shut down the process in a safe and controlled manner.
The ESD system needs uninterruptible power supply, further referred to as UPS, such that it can be operational even upon loss of main power. In traditional valve systems, there needs to be a hydraulic supply including back-up hydraulic accumulators, which in the case of an all-electric system will be replaced with electric actuators.

The actions of the ESD system are defined by NORSOK S-001 [10] to be:

- shutdown of dry well tresses, riser valves and subsea well trees

- initiate stop of subsea pipeline and flow lines connected to the installation. The closure signal can be realized in the subsea control system
- shutdown and sectioning of the hydrocarbon process facilities. This has to be changed or revised to be fitting to an all-electric control module
- shutdown of main power generation
- ignition source isolation
- initiation of emergency depressurization
- shutdown of drilling, intervention and work-over equipment not required for well control
- start/stop of emergency power generator via voltage sensor

These are important barriers needed for the ESD system to work properly for oil and gas production, and are significant aspects to be included in an x-mas tree. A barrier is explained in NOG 070 [11] as:

> "*A measure intended to identify conditions that may lead to failure, hazard and accident situations, prevent an actual sequence of event occurring or developing, influence a sequence of events in a deliberate way, or limit damage and/or loss.* (PSA, 2017)"

To put the ESD system in a more understandable perspective in terms of safety barriers: an ESD system is considered to be a barrier system, whereas the safety valves and ESD push buttons, the activation button in the control room, may be considered to be barrier elements.

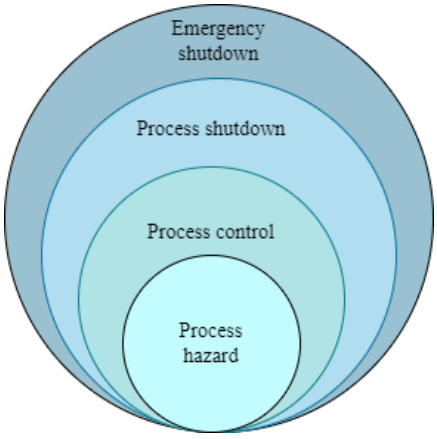


Figure A.1: Topside layers [4]

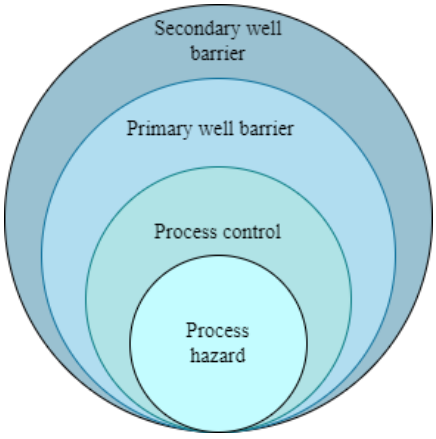


Figure A.2: X-mas tree layers [4]

There are different types of barrier isolation modes for subsea ESD isolation of a single subsea well. The primary well barrier isolation consists of PMV and PWV, which are the isolation valves in an x-mas tree. The second barrier isolation consists of closing the DHSV. All the ESD functionality starts at the unit where the isolation is initiated, most usually push button [11]. As seen in Figure A.1 and A.2, barriers for x-mas trees are based on topside layers of protection [4]. ESD and process shutdown (PSD) are still relevant for an x-mas tree, but the primary and secondary barriers will execute these. In the process control barrier, the PMV and the PWV is also included in addition to other elements such as a choke and other valves on the manifold. An important safety aspect to distinguish between topside and x-mas protection, is that for the x-mas tree, the first, and second barrier should never under any circumstance share valves, whereas this is acceptable for top side.

For specific safety strategy, the tree-structured hierarchy for ESD functions described in NORSOK S-001 [10] should be used. An edited version of the ESD hierarchy is seen in Figure A.3, where parts of the hierarchy found to be relevant for the all-electric safety valve system are highlighted. The structure in Figure A.3 is referred to as "hierarchy" because of the order in which different emergency shutdown systems are activated in response to an emergency. The hierarchy includes different levels of ESD systems, which are designed to handle different types of emergencies. It consists of the levels APS (Abandon platform shutdown), ESD1, ESD2 and PSD. A higher level ESD initiates lower level ESDs, including PSD, but a lower level signal should never initiate a higher level ESD. Also, the systems shall be able to perform intended functionality independently of other systems [11].

To contextualize the different levels of the hierarchy, one can look at the top level and the bottom level of the hierarchy. Abandon platform shutdown (APS) is the top level and is in the last line of defense. It is only activated when all the other levels have been activated and the situation on the platform cannot be safely managed. APS is only triggered when the situation on the platform is so severe that it is no longer safe for the personnel to remain.

The lowest level, the process shutdown (PSD), refers to a situation where the production process becomes unsafe, or when the process needs to be shut down for maintenance. PSD can be triggered if temperature or pressure exceeds thresholds. In case of a PSD, the process is shut down in a controlled manage to prevent further damage or injury by for example closing valves or stopping pumps. This helps to prevent the situation from escalating and to minimize the potential for damage or injury at an early stage.

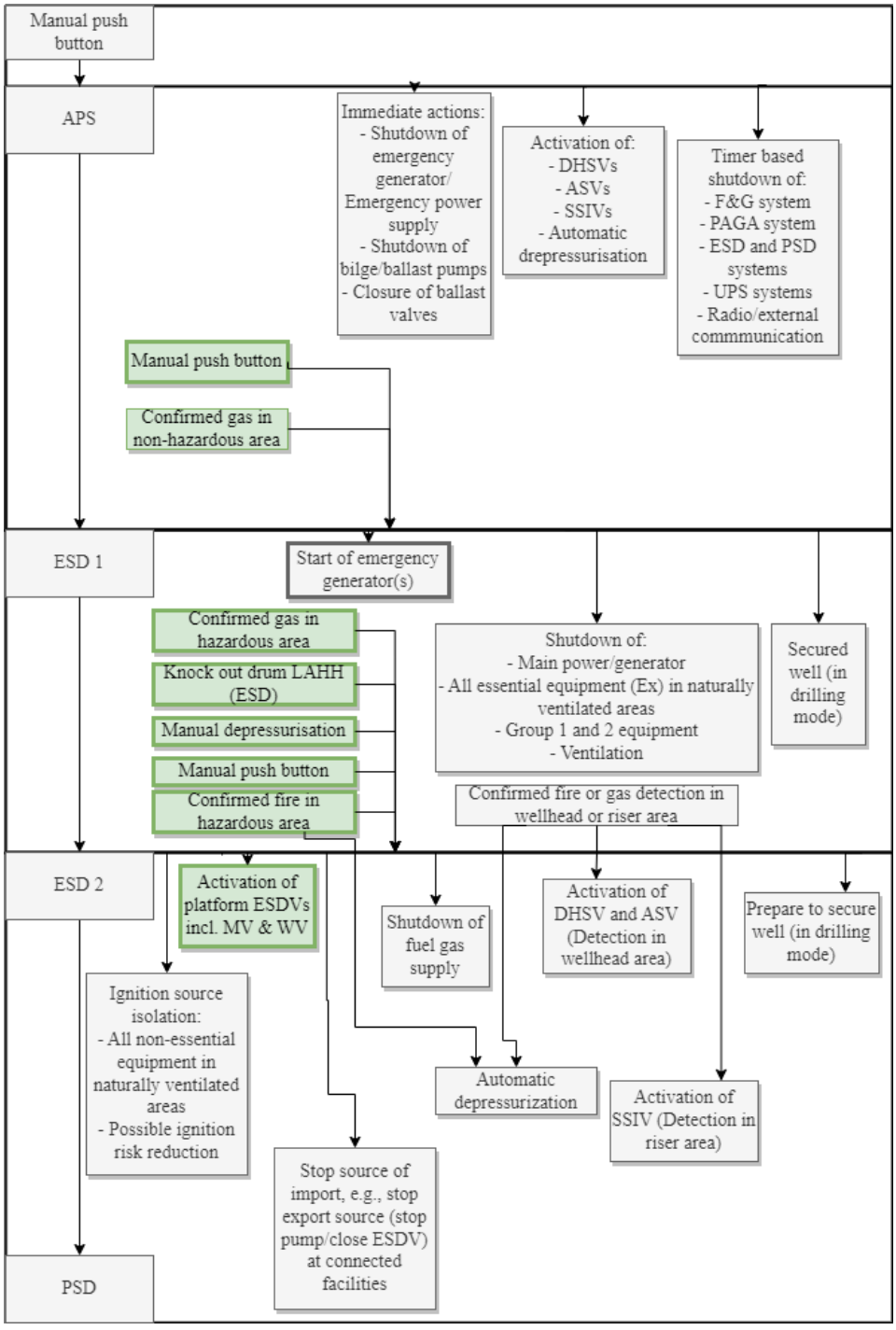


Figure A.3: Tree-structure hierarchy of ESD functions

B Event-B specification of ActuationSystem

The complete formal model for the all-electric actuation system is provided in this appendix. The model consists of both abstract and concrete machine specifications, and the two contexts, **as_ctx** and **as_ctx1** which were created for this project. This document is generated using the extension "B2Latex" in Rodin.

CONTEXT as_ctx
SETS
MotorStates
ValveStates
BatteryStates
StemMovements
TemperatureSensor
PressureSensor
TopsideSupply
CONSTANTS
Stopped
Working
Open
Closed
FullyCharged
Depleted
Charging
Providing
Retracting
Extracting
Inactive
Acceptable
Unacceptable
Safe
Unsafe
Available
Unavailable
AXIOMS
axm1: $partition(MotorStates, \{Stopped\}, \{Working\})$
axm2: $partition(ValveStates, \{Open\}, \{Closed\})$
axm3: $partition(BatteryStates, \{FullyCharged\}, \{Depleted\}, \{Charging\}, \{Providing\})$

axm4: $partition(StemMovements, \{Retracting\}, \{Extracting\}, \{Inactive\})$
axm5: $partition(TemperatureSensor, \{Acceptable\}, \{Unacceptable\})$
axm6: $partition(PressureSensor, \{Safe\}, \{Unsafe\})$
axm7: $partition(TopsideSupply, \{Available\}, \{Unavailable\})$
END

CONTEXT as_ctx1
EXTENDS as_ctx
SETS
SafetyController
CONSTANTS
CloseValve
OpenValve
OverrideBMS
Idle
AXIOMS
axm1: $partition(SafetyController, \{CloseValve\}, \{OpenValve\}, \{OverrideBMS\}, \{Idle\})$

END

MACHINE as_mac
SEES as_ctx
VARIABLES
battery_state
topside_supply
battery_level
INVARIANTS
inv4: $battery\_state \subseteq BatteryStates$
inv5: $topside\_supply \in TopsideSupply$
inv6: $0 < battery\_level \land battery\_level \leq 100$
inv7: $topside\_supply = Unavailable \Rightarrow Providing \in battery\_state$
EVENTS
Initialisation
begin
act1: $battery\_state := \{Charging\}$
act2: $topside\_supply := Available$
act3: $battery\_level := 100$
end
Event SetBatteryState ⟨ordinary⟩ $\widehat{=}$
any
new_battery_state
where
grd1: $Charging \in battery\_state \land battery\_level = 100 \Rightarrow FullyCharged \in new\_battery\_state$
grd2: $topside\_supply = Unavailable \Rightarrow new\_battery\_state = \{Providing\}$
grd3: $topside\_supply = Available \land battery\_level < 100 \Rightarrow new\_battery\_state = \{Charging\}$
grd4: $0 \leq battery\_level \land battery\_level \leq 20 \Rightarrow new\_battery\_state = \{Depleted\}$
grd5: $topside\_supply = Unavailable \land battery\_level < 100 \Rightarrow battery\_state = \{Providing, Charging\}$
grd6: $topside\_supply = Available \land battery\_level = 100 \Rightarrow new\_battery\_state = \{FullyCharged\}$
grd7: $battery\_state = \{Depleted\} \land (battery\_level < 100 \land battery\_level > 20) \Rightarrow new\_battery\_state = \{Charging\}$
then
act1: $battery\_state := new\_battery\_state$
end
END

MACHINE as_mac1
REFINES as_mac
SEES as_ctx
VARIABLES
battery_state
topside_supply
battery_level
pressure_level
ESD_initiated
INVARIANTS
inv1: $pressure\_level \in PressureSensor$
inv2: $ESD\_initiated \in BOOL$
EVENTS
Initialisation ⟨extended⟩
begin
act1: $battery\_state := \{Charging\}$
act2: $topside\_supply := Available$
act3: $battery\_level := 100$
act4: $pressure\_level := Safe$
act5: $ESD\_initiated := FALSE$
end
Event SetBatteryState ⟨ordinary⟩ $\widehat{=}$
extends SetBatteryState
any
$new\_battery\_state$
where
grd1: $Charging \in battery\_state \wedge battery\_level = 100 \Rightarrow FullyCharged \in new\_battery\_state$
grd2: $topside\_supply = Unavailable \Rightarrow new\_battery\_state = \{Providing\}$
grd3: $topside\_supply = Available \wedge battery\_level < 100 \Rightarrow new\_battery\_state = \{Charging\}$
grd4: $0 \leq battery\_level \wedge battery\_level \leq 20 \Rightarrow new\_battery\_state = \{Depleted\}$
grd5: $topside\_supply = Unavailable \wedge battery\_level < 100 \Rightarrow battery\_state = \{Providing, Charging\}$
grd6: $topside\_supply = Available \wedge battery\_level = 100 \Rightarrow new\_battery\_state = \{FullyCharged\}$
grd7: $battery\_state = \{Depleted\} \wedge (battery\_level < 100 \wedge battery\_level > 20) \Rightarrow new\_battery\_state = \{Charging\}$
then
act1: $battery\_state := new\_battery\_state$
end
Event InitiateESD ⟨ordinary⟩ $\widehat{=}$
any
new_ESD_state
where

grd1: $pressure\_level = Unsafe \wedge ESD\_initiated = FALSE \Rightarrow new\_ESD\_state = TRUE$

grd2: $pressure\_level = Safe \wedge ESD\_initiated = TRUE \Rightarrow new\_ESD\_state = FALSE$

then

act1: $ESD\_initiated := new\_ESD\_state$

end

END

MACHINE as_mac2
REFINES as_mac1
SEES as_ctx1
VARIABLES

battery_state
topside_supply
battery_level
pressure_level
ESD_initiated
safety_command
valve_position

INVARIANTS

inv1: $safety\_command \subseteq SafetyController$
inv2: $ESD\_initiated = TRUE \Rightarrow CloseValve \in safety\_command$
inv3: $valve\_position \in ValveStates$
inv4: $OverrideBMS \in safety\_command \Rightarrow Providing \in battery\_state$
inv5: $ESD\_initiated = FALSE \Rightarrow CloseValve \notin safety\_command$

EVENTS

Initialisation ⟨extended⟩

begin

act1: $battery\_state := \{Charging\}$
act2: $topside\_supply := Available$
act3: $battery\_level := 100$
act4: $pressure\_level := Safe$
act5: $ESD\_initiated := FALSE$
act6: $safety\_command := \{Idle\}$
act7: $valve\_position := Open$

end

Event SetBatteryState ⟨ordinary⟩ $\widehat{=}$
refines SetBatteryState

any

new_battery_state

where

grd1: $Charging \in battery\_state \land battery\_level = 100 \Rightarrow FullyCharged \in new\_battery\_state$
grd2: $topside\_supply = Unavailable \Rightarrow new\_battery\_state = \{Providing\}$
grd3: $topside\_supply = Available \land battery\_level < 100 \Rightarrow new\_battery\_state = \{Charging\}$
grd4: $0 \leq battery\_level \land battery\_level \leq 20 \Rightarrow new\_battery\_state = \{Depleted\}$
grd5: $topside\_supply = Unavailable \land battery\_level < 100 \Rightarrow battery\_state = \{Providing, Charging\}$
grd6: $topside\_supply = Available \land battery\_level = 100 \Rightarrow new\_battery\_state = \{FullyCharged\}$
grd7: $battery\_state = \{Depleted\} \land (battery\_level < 100 \land battery\_level > 20) \Rightarrow new\_battery\_state = \{Charging\}$

grd8: $Providing \notin battery\_state \wedge OverrideBMS \in safety\_command \Rightarrow new\_battery\_state = \{Providing\}$
grd9: $Providing \notin battery\_state \wedge OverrideBMS \in safety\_command \wedge battery\_level < 100 \Rightarrow new\_battery\_state = \{Providing, Charging\}$
grd10: $OverrideBMS \in safety\_command \Rightarrow new\_battery\_state = battery\_state \cup \{Providing\}$

then

act1: $battery\_state := new\_battery\_state$

end

Event InitiateESD_safety ⟨ordinary⟩ $\widehat{=}$

refines InitiateESD

any

new_ESD_state

where

grd1: $pressure\_level = Unsafe \wedge ESD\_initiated = FALSE \Rightarrow new\_ESD\_state = TRUE \wedge safety\_command = \{CloseValve, OverrideBMS\}$
grd2: $pressure\_level = Safe \wedge ESD\_initiated = TRUE \wedge CloseValve \in safety\_command \Rightarrow new\_ESD\_state = FALSE$
grd3: $pressure\_level = Safe \wedge ESD\_initiated = FALSE \Rightarrow new\_ESD\_state = FALSE$
grd4: $ESD\_initiated = FALSE \Rightarrow safety\_command = safety\_command \setminus \{CloseValve\}$
grd5: $new\_ESD\_state = TRUE \Rightarrow safety\_command = safety\_command \cup \{CloseValve\}$
grd6: $new\_ESD\_state = TRUE \Rightarrow CloseValve \in safety\_command$
grd7: $ESD\_initiated = TRUE \wedge CloseValve \in safety\_command \Rightarrow new\_ESD\_state = TRUE$

then

act1: $ESD\_initiated := new\_ESD\_state$

end

Event SetSafetyCommand ⟨ordinary⟩ $\widehat{=}$

any

new_safety_command

where

grd1: $ESD\_initiated = TRUE \wedge valve\_position = Open \Rightarrow new\_safety\_command = \{CloseValve\}$
grd2: $ESD\_initiated = TRUE \Rightarrow new\_safety\_command = \{CloseValve\}$
grd3: ⟨theorem⟩ $topside\_supply = Unavailable \wedge Providing \notin battery\_state \Rightarrow OverrideBMS \in new\_safety\_command$
grd4: $ESD\_initiated = TRUE \wedge valve\_position = Closed \Rightarrow new\_safety\_command = \{Idle\}$
grd5: $ESD\_initiated = FALSE \wedge valve\_position = Closed \Rightarrow new\_safety\_command = \{OpenValve\}$
grd6: $ESD\_initiated = FALSE \wedge valve\_position = Open \Rightarrow new\_safety\_command = \{Idle\}$
grd7: $ESD\_initiated = FALSE \Rightarrow new\_safety\_command = \{Idle\}$
grd8: $OverrideBMS \in new\_safety\_command \Rightarrow Providing \in battery\_state$

then

act1: $safety\_command := new\_safety\_command$

end

Event SetValvePosition ⟨ordinary⟩ $\widehat{=}$

any

new_valve_position

where

grd1: $pressure\_level = Safe \wedge OpenValve \in safety\_command \wedge ESD\_initiated = FALSE \Rightarrow new\_valve\_position = Open$

grd2: $pressure\_level = Unsafe \wedge CloseValve \in safety\_command \Rightarrow new\_valve\_position = Closed$

then

act1: $valve\_position := new\_valve\_position$

end

END

MACHINE as_mac3
REFINES as_mac2
SEES as_ctx1
VARIABLES
valve_position
battery_state
topside_supply
battery_level
pressure_level
ESD_initiated
safety_command
stem_movement
INVARIANTS
inv1: $stem\_movement \in StemMovements$
EVENTS
Initialisation ⟨extended⟩
begin
act1: $battery\_state := \{Charging\}$
act2: $topside\_supply := Available$
act3: $battery\_level := 100$
act4: $pressure\_level := Safe$
act5: $ESD\_initiated := FALSE$
act6: $safety\_command := \{Idle\}$
act7: $valve\_position := Open$
act8: $stem\_movement := Inactive$
end
Event SetBatteryState ⟨ordinary⟩ $\widehat{=}$
extends SetBatteryState
any
$new\_battery\_state$
where
grd1: $Charging \in battery\_state \wedge battery\_level = 100 \Rightarrow FullyCharged \in new\_battery\_state$
grd2: $topside\_supply = Unavailable \Rightarrow new\_battery\_state = \{Providing\}$
grd3: $topside\_supply = Available \wedge battery\_level < 100 \Rightarrow new\_battery\_state = \{Charging\}$
grd4: $0 \leq battery\_level \wedge battery\_level \leq 20 \Rightarrow new\_battery\_state = \{Depleted\}$
grd5: $topside\_supply = Unavailable \wedge battery\_level < 100 \Rightarrow battery\_state = \{Providing, Charging\}$
grd6: $topside\_supply = Available \wedge battery\_level = 100 \Rightarrow new\_battery\_state = \{FullyCharged\}$
grd7: $battery\_state = \{Depleted\} \wedge (battery\_level < 100 \wedge battery\_level > 20) \Rightarrow new\_battery\_state = \{Charging\}$
grd8: $Providing \notin battery\_state \wedge OverrideBMS \in safety\_command \Rightarrow new\_battery\_state = \{Providing\}$

grd9: $Providing \notin battery\_state \wedge OverrideBMS \in safety\_command \wedge battery\_level < 100 \Rightarrow new\_battery\_state = \{Providing, Charging\}$
grd10: $OverrideBMS \in safety\_command \Rightarrow new\_battery\_state = battery\_state \cup \{Providing\}$

then

act1: $battery\_state := new\_battery\_state$

end

Event InitiateESD_safety ⟨ordinary⟩ $\widehat{=}$

extends InitiateESD_safety

any

new_ESD_state

where

grd1: $pressure\_level = Unsafe \wedge ESD\_initiated = FALSE \Rightarrow new\_ESD\_state = TRUE \wedge safety\_command = \{CloseValve, OverrideBMS\}$
grd2: $pressure\_level = Safe \wedge ESD\_initiated = TRUE \wedge CloseValve \in safety\_command \Rightarrow new\_ESD\_state = FALSE$
grd3: $pressure\_level = Safe \wedge ESD\_initiated = FALSE \Rightarrow new\_ESD\_state = FALSE$
grd4: $ESD\_initiated = FALSE \Rightarrow safety\_command = safety\_command \setminus \{CloseValve\}$
grd5: $new\_ESD\_state = TRUE \Rightarrow safety\_command = safety\_command \cup \{CloseValve\}$
grd6: $new\_ESD\_state = TRUE \Rightarrow CloseValve \in safety\_command$
grd7: $ESD\_initiated = TRUE \wedge CloseValve \in safety\_command \Rightarrow new\_ESD\_state = TRUE$

then

act1: $ESD\_initiated := new\_ESD\_state$

end

Event SetSafetyCommand ⟨ordinary⟩ $\widehat{=}$

extends SetSafetyCommand

any

$new\_safety\_command$

where

grd1: $ESD\_initiated = TRUE \wedge valve\_position = Open \Rightarrow new\_safety\_command = \{CloseValve\}$
grd2: $ESD\_initiated = TRUE \Rightarrow new\_safety\_command = \{CloseValve\}$
grd3: ⟨theorem⟩ $topside\_supply = Unavailable \wedge Providing \notin battery\_state \Rightarrow OverrideBMS \in new\_safety\_command$
grd4: $ESD\_initiated = TRUE \wedge valve\_position = Closed \Rightarrow new\_safety\_command = \{Idle\}$
grd5: $ESD\_initiated = FALSE \wedge valve\_position = Closed \Rightarrow new\_safety\_command = \{OpenValve\}$
grd6: $ESD\_initiated = FALSE \wedge valve\_position = Open \Rightarrow new\_safety\_command = \{Idle\}$
grd7: $ESD\_initiated = FALSE \Rightarrow new\_safety\_command = \{Idle\}$
grd8: $OverrideBMS \in new\_safety\_command \Rightarrow Providing \in battery\_state$

then

act1: $safety\_command := new\_safety\_command$

end

Event SetStemMovement ⟨ordinary⟩ $\widehat{=}$

any

new_stem_movement

where

grd1: $valve\_position = Open \wedge CloseValve \in safety\_command \Rightarrow new\_stem\_movement = Extracting$

grd2: $valve\_position = Closed \wedge OpenValve \in safety\_command \Rightarrow new\_stem\_movement = Retracting$

grd3: $valve\_position = Closed \wedge CloseValve \in safety\_command \Rightarrow new\_stem\_movement = Inactive$

grd4: $valve\_position = Open \wedge OpenValve \in safety\_command \Rightarrow new\_stem\_movement = Inactive$

grd5: $stem\_movement = Retracting \wedge valve\_position = Open \Rightarrow new\_stem\_movement = Inactive$

grd6: $stem\_movement = Extracting \wedge valve\_position = Closed \Rightarrow new\_stem\_movement = Inactive$

grd7: $safety\_command = \{Idle\} \Rightarrow new\_stem\_movement = Inactive$

then

act1: $stem\_movement := new\_stem\_movement$

end

Event SetValvePosition ⟨ordinary⟩ $\widehat{=}$

refines SetValvePosition

any

new_valve_position

where

grd1: $pressure\_level = Safe \wedge OpenValve \in safety\_command \wedge ESD\_initiated = FALSE \Rightarrow new\_valve\_position = Open$

grd2: $pressure\_level = Unsafe \wedge CloseValve \in safety\_command \Rightarrow new\_valve\_position = Closed$

grd3: $valve\_position = Closed \wedge stem\_movement = Retracting \Rightarrow new\_valve\_position = Open$

grd4: $valve\_position = Open \wedge stem\_movement = Extracting \Rightarrow new\_valve\_position = Closed$

then

act1: $valve\_position := new\_valve\_position$

end

END

MACHINE as_mac4
REFINES as_mac3
SEES as_ctx1
VARIABLES
valve_position
battery_state
topside_supply
battery_level
pressure_level
ESD_initiated
safety_command
stem_movement
motor_state
temperature
INVARIANTS
inv1: $motor\_state \in MotorStates$
inv2: $temperature \in TemperatureSensor$
inv3: $temperature = Unacceptable \Rightarrow motor\_state \neq Working$
inv4: $motor\_state = Stopped \Rightarrow stem\_movement = Inactive$
EVENTS
Initialisation ⟨extended⟩
begin
act1: $battery\_state := \{Charging\}$
act2: $topside\_supply := Available$
act3: $battery\_level := 100$
act4: $pressure\_level := Safe$
act5: $ESD\_initiated := FALSE$
act6: $safety\_command := \{Idle\}$
act7: $valve\_position := Open$
act8: $stem\_movement := Inactive$
act9: $motor\_state := Stopped$
act10: $temperature := Acceptable$
end
Event SetBatteryState ⟨ordinary⟩ $\widehat{=}$
extends SetBatteryState
any
$new\_battery\_state$
where
grd1: $Charging \in battery\_state \wedge battery\_level = 100 \Rightarrow FullyCharged \in new\_battery\_state$
grd2: $topside\_supply = Unavailable \Rightarrow new\_battery\_state = \{Providing\}$
grd3: $topside\_supply = Available \wedge battery\_level < 100 \Rightarrow new\_battery\_state = \{Charging\}$
grd4: $0 \leq battery\_level \wedge battery\_level \leq 20 \Rightarrow new\_battery\_state = \{Depleted\}$

grd5: $topside\_supply = Unavailable \land battery\_level < 100 \Rightarrow battery\_state = \{Providing, Charging\}$
grd6: $topside\_supply = Available \land battery\_level = 100 \Rightarrow new\_battery\_state = \{FullyCharged\}$
grd7: $battery\_state = \{Depleted\} \land (battery\_level < 100 \land battery\_level > 20) \Rightarrow new\_battery\_state = \{Charging\}$
grd8: $Providing \notin battery\_state \land OverrideBMS \in safety\_command \Rightarrow new\_battery\_state = \{Providing\}$
grd9: $Providing \notin battery\_state \land OverrideBMS \in safety\_command \land battery\_level < 100 \Rightarrow new\_battery\_state = \{Providing, Charging\}$
grd10: $OverrideBMS \in safety\_command \Rightarrow new\_battery\_state = battery\_state \cup \{Providing\}$

then

act1: $battery\_state := new\_battery\_state$

end

Event InitiateESD_safety ⟨ordinary⟩ $\widehat{=}$

extends InitiateESD_safety

any

new_ESD_state

where

grd1: $pressure\_level = Unsafe \land ESD\_initiated = FALSE \Rightarrow new\_ESD\_state = TRUE \land safety\_command = \{CloseValve, OverrideBMS\}$
grd2: $pressure\_level = Safe \land ESD\_initiated = TRUE \land CloseValve \in safety\_command \Rightarrow new\_ESD\_state = FALSE$
grd3: $pressure\_level = Safe \land ESD\_initiated = FALSE \Rightarrow new\_ESD\_state = FALSE$
grd4: $ESD\_initiated = FALSE \Rightarrow safety\_command = safety\_command \setminus \{CloseValve\}$
grd5: $new\_ESD\_state = TRUE \Rightarrow safety\_command = safety\_command \cup \{CloseValve\}$
grd6: $new\_ESD\_state = TRUE \Rightarrow CloseValve \in safety\_command$
grd7: $ESD\_initiated = TRUE \land CloseValve \in safety\_command \Rightarrow new\_ESD\_state = TRUE$

then

act1: $ESD\_initiated := new\_ESD\_state$

end

Event SetSafetyCommand ⟨ordinary⟩ $\widehat{=}$

extends SetSafetyCommand

any

$new\_safety\_command$

where

grd1: $ESD\_initiated = TRUE \land valve\_position = Open \Rightarrow new\_safety\_command = \{CloseValve\}$
grd2: $ESD\_initiated = TRUE \Rightarrow new\_safety\_command = \{CloseValve\}$
grd3: ⟨theorem⟩ $topside\_supply = Unavailable \land Providing \notin battery\_state \Rightarrow OverrideBMS \in new\_safety\_command$
grd4: $ESD\_initiated = TRUE \land valve\_position = Closed \Rightarrow new\_safety\_command = \{Idle\}$

grd5: $ESD\_initiated = FALSE \wedge valve\_position = Closed \Rightarrow new\_safety\_command = \{OpenValve\}$
grd6: $ESD\_initiated = FALSE \wedge valve\_position = Open \Rightarrow new\_safety\_command = \{Idle\}$
grd7: $ESD\_initiated = FALSE \Rightarrow new\_safety\_command = \{Idle\}$
grd8: $OverrideBMS \in new\_safety\_command \Rightarrow Providing \in battery\_state$

then

act1: $safety\_command := new\_safety\_command$

end

Event SetStemMovement ⟨ordinary⟩ $\widehat{=}$

refines SetStemMovement

any

new_stem_movement

where

grd1: $valve\_position = Open \wedge CloseValve \in safety\_command \Rightarrow new\_stem\_movement = Extracting$
grd2: $valve\_position = Closed \wedge OpenValve \in safety\_command \Rightarrow new\_stem\_movement = Retracting$
grd3: $valve\_position = Closed \wedge CloseValve \in safety\_command \Rightarrow new\_stem\_movement = Inactive$
grd4: $valve\_position = Open \wedge OpenValve \in safety\_command \Rightarrow new\_stem\_movement = Inactive$
grd5: $stem\_movement = Retracting \wedge valve\_position = Open \Rightarrow new\_stem\_movement = Inactive$
grd6: $stem\_movement = Extracting \wedge valve\_position = Closed \Rightarrow new\_stem\_movement = Inactive$
grd7: $safety\_command = \{Idle\} \Rightarrow new\_stem\_movement = Inactive$
grd8: $motor\_state = Stopped \Rightarrow new\_stem\_movement = Inactive$

then

act1: $stem\_movement := new\_stem\_movement$

end

Event SetMotorState ⟨ordinary⟩ $\widehat{=}$

any

new_motor_state

where

grd2: $(stem\_movement = Retracting \vee stem\_movement = Extracting) \wedge temperature = Acceptable \Rightarrow new\_motor\_state = Working$
grd3: $Providing \notin battery\_state \wedge topside\_supply = Unavailable \Rightarrow new\_motor\_state = Stopped$
grd4: $temperature = Unacceptable \Rightarrow new\_motor\_state = Stopped$
grd5: $stem\_movement \neq Inactive \Rightarrow new\_motor\_state = Working$

then

act1: $motor\_state := new\_motor\_state$

end

Event SetValvePosition ⟨ordinary⟩ $\widehat{=}$

extends SetValvePosition

any

$new\_valve\_position$

where

grd1: $pressure\_level = Safe \wedge OpenValve \in safety\_command \wedge ESD\_initiated = FALSE \Rightarrow new\_valve\_position = Open$

grd2: $pressure\_level = Unsafe \wedge CloseValve \in safety\_command \Rightarrow new\_valve\_position = Closed$

grd3: $valve\_position = Closed \wedge stem\_movement = Retracting \Rightarrow new\_valve\_position = Open$

grd4: $valve\_position = Open \wedge stem\_movement = Extracting \Rightarrow new\_valve\_position = Closed$

then

act1: $valve\_position := new\_valve\_position$

end

END

C Proposed motor model

This section of the appendix introduces the PMSM motor which is the proposed motor for the all-electric actuation system in this thesis. The motor model is chosen based on the details provided in 2.6, and each part included in both the motor and the motor controller is presented here. The reason for an added appendix is that the details of how the motor is controlled and what the motor model consists of is not necessary for the safety analysis, but it is recommended to have some sort of understanding of it for a better perspective of the total system.

C.1 Motor model

A PMSM uses an AC supply to create a rotating magnetic field which causes the rotor to rotate synchronously with the field, hence converting electrical energy to mechanical energy [61]. A three-phase system is advantageous for the PMSM due to the fact that three-phase systems are less costly and more efficient than a single-phase system, it takes up less space and has constant power.

The voltages and currents for the PMSM is given by the following equations for a balanced three-phase system [61]:

$$X_a(t) = Acos(\omega t + \phi) = A\angle\phi \tag{C.1}$$

$$X_b(t) = Acos(\omega t + \phi - 120^\circ) = X_a(t)(1\angle - 120^\circ \tag{C.2}$$

$$X_c(t) = Acos(\omega t + \phi + 120^\circ) = X_a(t)(1\angle 120^\circ \tag{C.3}$$

$$\tag{C.4}$$

where X is the signal, A is the amplitude, ω is the frequency and ϕ is the initial angle.

C.2 Clarke/Park transformations

The motor which is driven by a three-phase system as described in the equations C.4 is usually simplified to avoid controlling the motor studying sinusoidal waveforms. To simplify the motor model, Clarke and Park transformations, also known as $\alpha\beta$-transformation and $d-q-0$-transformations, are common. Clarke and Park transformations are used in three power system analysis, and are commonly used in field-oriented control [62]. The Clarke transform converts time domain components of a three-phase system to two components in an orthogonal stationary frame, also known as the $\alpha\beta$-frame. Two-phase components are then transformed into DC quantities by Park transform in an orthogonal rotating reference frame, known as the dq-frame. This is because it is ideal to have DC motor control in PMSM control due to the fact that DC motor control is simple. It is simple because all controlled quantities are DC values in steady state, and current phase/angle is controlled by a mechanical commutator [63]. Further on, the system can be controlled using DC signals rather than sinusoidal waveforms. The Clarke/Park transformations will be a part of the digital twin depicted in Figure 2.6. Some details are advantageous for the analysis in this system, and are presented in Appendix C.

The different reference frames which are constructed from the Clarke and Park transformations can be seen in Figure C.1

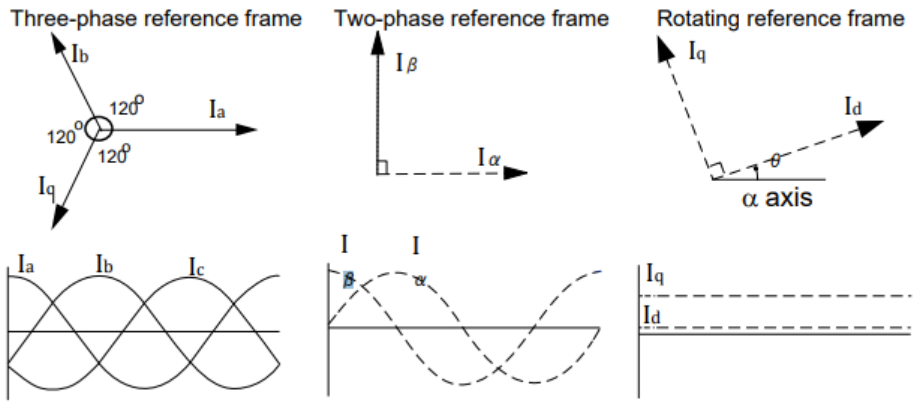


Figure C.1: Reference frames [64]

The Clarke Transformation is expressed by the following equations [64]:

$$I_\alpha = \frac{2}{3}(I_a) - \frac{1}{3}(I_b - I_c) \tag{C.5}$$

$$I_\beta = \frac{2}{\sqrt{3}}(I_b - I_c) \tag{C.6}$$

I_a, I_b, I_c are three-phase quantities, whereas I_α and I_β are stationary orthogonal reference frame quantities. When $I_a + I_b + I_c = 0$ and I_α is superposed as I_a the three-phase quantities can be transformed as:

$$I_\alpha = I_a \tag{C.7}$$

$$I_\beta = \frac{1}{\sqrt{3}}(I_a + 2I_b) \tag{C.8}$$

To transform back to three-phase, Inverse Clarke Transformation can be used by the following equations:

$$V_a = V_\alpha \tag{C.9}$$

$$V_b = \frac{-V_\alpha + \sqrt{3} * V_\beta}{2} \tag{C.10}$$

$$V_c = \frac{-V_\alpha - \sqrt{3} * V_\beta}{2} \tag{C.11}$$

To further transform from two-axis orthogonal stationary reference frames to a rotating reference frame, Park transformation is used by the following equations:

$$I_d = I_\alpha * cos(\theta) + I_\beta * sin(\theta) \tag{C.12}$$

$$I_q = I_\beta * cos(\theta) - I_\alpha * sin(\theta) \tag{C.13}$$

and the quantities are returned to two-axis with Inverse Park Transformation:

$$V_\alpha = V_d * cos(\theta) - V_q * sin(\theta) \tag{C.14}$$
$$V_\beta = V_q * cos(\theta) + V_d * sin(\theta) \tag{C.15}$$

Combining Clarke transformation with Park transformation gives Park-Clarke transformation from three-phase values to their $d-q$ reference frames, such that current and voltage signals no longer are sinusoidal signals, but DC signals. The combination can be seen as [65]:

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos\theta_e & \cos\theta_e - \frac{2\pi}{3} & \cos\theta_e - \frac{4\pi}{3} \\ -\sin\theta_e & -\sin\theta_e - \frac{2\pi}{3} & -\sin\theta_e - \frac{4\pi}{3} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{C.16}$$

C.3 Field Oriented Control

Field Oriented Control (FOC) is a control method based on field orientation. The magnetic flux generated from the PM motor is fixed in relation to the rotor shaft position, so the flux position can be determined by the shaft position [66]. The rotor flux is only produced in the q-axis, whereas the current vector is generated in the FOC-axis [67]. The motor torque is linearly proportional to the q-axis current since the d-axis rotor flux is constant, one can achieve the maximum torque per ampere. The q-axis component of the stator flux contributes to torque generation, and the d-axis component only magnetizes the machine. This results in repulsive forces generating a torque on the rotor when the magnetic poles of the stator are not aligned with their opposite counter-part.

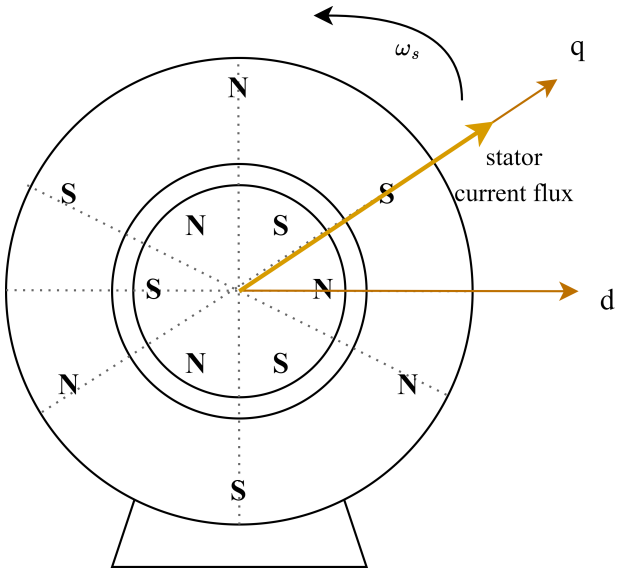


Figure C.2: Maximum torque alignment of flux vector

Figure C.2 shows the q and d-axis components and how the maximum torque, ω_s is generated when the flux vector is aligned to the q-axis. If the flux vector were aligned to the d-axis, there would be zero torque generation.

The control of the stator currents can simply be done by two separate PI controllers for q-axis and d-axis, followed by basic operations to compute PWM duty cycles, such as space vector pulse width modulation [67]. FOC usually generates voltage references that the pulse width modulation modulator, known as PWM Generator, transforms into signals for a voltage source inverter. The PWM generator is also seen in Figure 2.6.

C.4 Space vector pulse width modulation

Space vector pulse width modulation, also known as SVPWM, is used on the outputs of the controller to generate a changing frequency, and is a common technique in FOC for PMSM. SVPM is a technique for generating a sine wave which provides a higher voltage to the motor and lower total harmonic distortion [68].

The general idea behind SVPWM is to use constellations from a three-phase inverter circuit in a space vector representation to generate a PWM signal [69]. The three-phase inverter has six switches, where each switching configuration results in a specific voltage applied to the motor terminals. The inverter circuit is connected to the stator windings of the motor. There is a total of eight constellations, where six of them, $V_1 - V_6$ are considered active. The two remaining vectors, V_0 and V_7 are zero vectors. The vectors can be seen in Figure C.3.

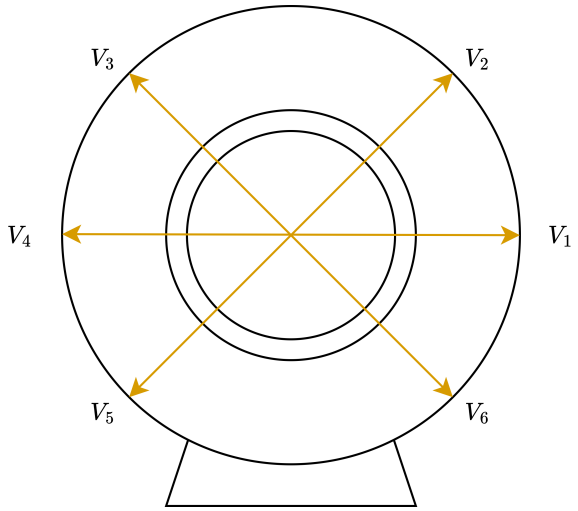


Figure C.3: Space vector modulations

The switching table for the three-phase inverter circuit can then be written as [69]:

Vector	V_a	V_b	V_c
V_0	0	0	0
V_1	1	0	0
V_2	1	1	0
V_3	0	1	0
V_4	0	1	1
V_5	0	0	1
V_6	1	0	1
V_7	1	1	1

Table C.1: Table for switching vectors

One of the configurations of switches can be illustrated as seen in Figure

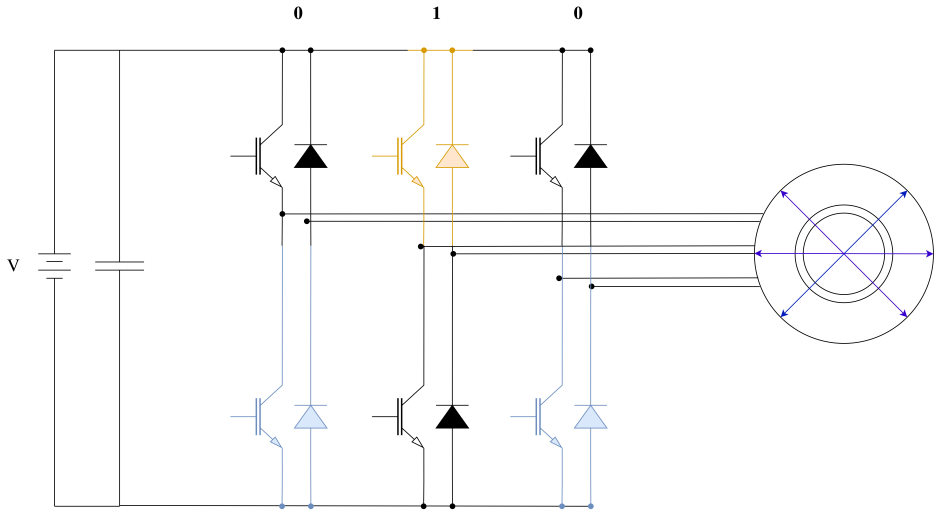
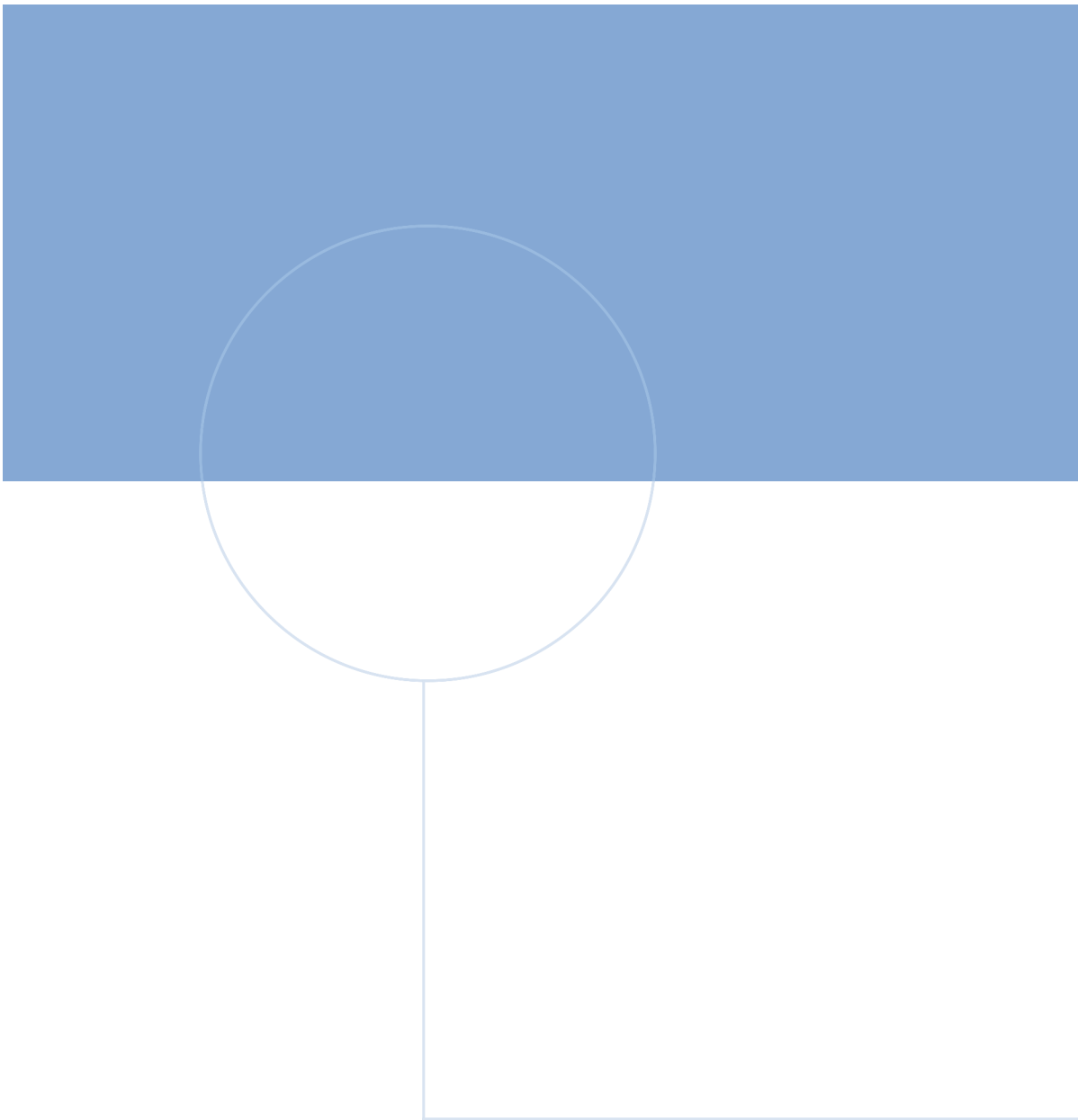


Figure C.4: Three-phase inverter connected to motor windings

The three-phase inverter generates the desired frequency and voltage for the motor by using the PWM pulses generated from the controller through the inverter. The total control structure is depicted in Figure 2.6, and will not be given more elaboration than these main components needed to understand the motor controller for further analysis.



NTNU

Norwegian University of
Science and Technology