NTNU

Norwegian University of
Science and Technology

Specification of Requirements for Safety in the Early Development Phases - Misuse Case and HAZOP in the Concept Phase

Joshua Maringa
Thorbjørn Sæther

Master of Science in Computer Science
Submission date: July 2011
Supervisor: Tor Stålhane, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

In the Specialization Project (TDT4520) prior to this thesis we studied how hazard analyses are used to identify safety requirements, in particular software safety requirements, and if there was some way to improve this process. The result of a survey conducted on experts in the field showed us that the most important project phase for safety analysis improvement is the concept phase. Based on our understanding, we designed a procedure for early software safety requirement identification that included the methods Misuse Case and HAZOP. A case study was performed, and it showed that the procedure could have merit.

We will run an experiment comparing the procedure with the more commonly used Preliminary Hazard Analysis. The data collected will be analyzed, modifications to the procedure will be made, if needed, and advantages and disadvantages will be highlighted. We also need to validate our assumptions with an experienced analyst.

With all this data collected and analyzed, we will reach a conclusion whether our procedure has merit or not.

Assignment given: 15. January 2011

Supervisor: Tor Stålhane

Abstract

In the course TDT4520 - Specialization Project, the preparatory course to this thesis, we looked at several safety analysis methods and how they could be exploited to identify software hazards in the early stages of development. After our evaluation, and with the results from a survey conducted on experts in the field, we proposed a procedure to improve software hazard identification in the concept phase of projects. The procedure consisted of a Misuse Case analysis with a sub-sequential HAZOP analysis. Our case study showed that this procedure will indeed aid in the identification process. However, testing the procedure on others is needed to see if this is correct. That is the main theme for this thesis.

We performed an experiment with undergraduate students and an interview with an expert in the field. We use the results from the experiment to validate our assumptions and identify modifications that might be needed. The experiment gave us a good illustration of how the procedure would work in a real hazard analysis project, and the data collected showed us the differences between it and the more commonly used Preliminary Hazard Analysis. Our hypothesis was that the Misuse Case and HAZOP approach would improve the hazard identification with focus on software. The experiment resulted in no clear difference in non-software parts of the system, but a clear improvement on the software parts. Afterwards we conducted an interview with an expert in the field, in which we clarified many of our questions and assumptions, and aided us in modifying the procedure to the better.

Although the procedure still needs to be tested thoroughly with real projects in the industry to make a final decision on whether it has merit or not, our conclusion is that the procedure deserves further attention. Software hazard identification in the concept phase is difficult but based on our findings, the Misuse Case and HAZOP combination can improve this problem.

Preface

This master thesis was written as part of our *MSc* at the Norwegian University of Science and Technology (NTNU), Department of Computer and Information Science, spring 2011. It extends the work done in the preliminary project *CESAR - Specification of requirements for safety in the early development phases* carried out by the same authors in fall 2010.

We would like to thank our supervisor Professor Dr. Tor Stålhane for his encouragement and input during this thesis. His guidance and support in this thesis has been invaluable. We would also like to thank Frank Reichenbach from ABB for his cooperation in our interview, and the undergraduate students that participated in our experiment.

We would also like to give a special thanks to our friends and families, who has always supported us.

Trondheim, June 28, 2011

Joshua Maringa Thorbjørn Sæther

Contents

Appendices 95

List of Figures

List of Tables

CHAPTER 1

Introduction

This chapter gives an introduction to the rest of the report. First we present the problem definition, and then we describe the motivation behind exploring this problem and the problem's context. At the end of the chapter we present an outline of the rest of the report.

1.1 Problem Definition

In the Specialization Project (TDT4520) prior to this thesis we studied how hazard analyses are used to identify safety requirements, in particular software safety requirements, and if there was some way to improve this process. The result of a survey conducted on experts in the field showed us that the most important project phase for safety analysis improvement is the concept phase.

Based on our understanding, we designed a procedure for early software safety requirement identification that included the two methods Misuse Case and HAZOP.

The thesis is mainly on testing and evaluating the procedure to see if it has merit, if it needs further tuning or if it has no merit at all.

1.2 Motivation

The concept phase is the first phase in the IEC 61508 safety life cycle [1]. Often in the concept phase, the only things that exists are the problem that needs to be solved, an idea of how to solve it and the environment and users of the solution. It is in this phase that the stakeholders decide whether the project is cost-effective and if the technology needed is available. Anything that could help them make this choice will be much appreciated.

For safety-critical systems one has to see if the potential risks are too large, or if they force limitations in the design. The earlier this can be identified, the better.

1.2.1 Purpose

We aim to thoroughly test the procedure, evaluate the test results and see if modifications to the two methods needs to be made, and finally, see if the procedure will improve the hazard identification to make safety requirements in the concept phase.

1.2.2 Objective

We shall perform an experiment and an interview. The experiment will be performed with computer science undergraduate students at NTNU. When we have evaluated the results and made the necessary modifications to the procedure, we will perform an interview with an expert in the field. After another round of evaluation of results in regard to the findings in the interview and making necessary modifications, we will decide whether our procedure has merit.

1.3 Context

Our method combines two known hazard analysis methods to try improving the identification of hazards in the early development phases, with focus on software. HAZOP is normally performed in later stages, but we believe that it will compliment Misuse Cases in a good way also in the concept phase.

This thesis has been carried out with supervision from Professor Tor Stålhane, at the Department of Computer and Information Science at the Norwegian University of Science and Technology (NTNU).

1.4 Outline

The remainder of this thesis document is organized in the following parts:

- **Preliminary Studies and Preparation** contains background information of the field, state of the art and our evaluation of our approach before conducting the experiment and interview.
- **Experiment** contains the experiment; its preparation, execution and aftermath.
- **Discussion** contains an interview with an expert in the field and our justification of our approach.
- **Conclusion** contains concluding remarks and points for further work.
- **References and Appendix** contains the references used in the thesis, the documents used in the experiment and its result.

Part I

Preliminary Studies and Preparation

CHAPTER 2

Preliminary Studies

Most of the preliminary studies were conducted in the TDT4520 Specialization Report [2], but the most essential for understanding the content of this thesis will be repeated and expanded in this chapter.

2.1 Software Safety Goals

The increased use of software systems to control and operate complex and safety-critical systems lends to the need for ensuring safety of the system during the concept phase or in the early phases of acquisition or planning of a software system. Safety critical systems require monitoring and control and thus software safety increasingly needs to be improved during the development process. Flaws in the design and/or implementation normally lead to software failures thus making the software system incapable of handling Abnormal Conditions and Events (ACE). Software by itself is neither safe nor unsafe. It depends on the environment the software executes in how safe a software can be [3]. Software safety ensures that software systems failures which can lead to death or injury to personnel, system loss and/or damage are reduced as much as possible while maintaining its purpose.

2.1.1 Software Safety Definitions

The definitions shown below should be used a guidance by system analysts and developers during software safety analysis [4].

- Mishap - can be defined as unexpected event or series of events resulting in injury, death, occupational illness or damage of property equipment or the environment in which the system is operating in.
- Software Hazard - is an error in the software that would lead to a mishap
- Hazard probability - the overall probability of a certain event occurring that leads to a specific hazard.
- Hazard severity - is determined by the worst possible mishap that could occur due to a specific hazard within its operational environment.

- Risk - the probability that a software system will suffer due to and event of negative impact to the system considering the system's existing vulnerabilities multiplied by the costs that may result due to its occurrence to its environment.
- Threat - is the source of danger to the system that can lead to system failure hence resulting to a catastrophic end.
- Vulnerability - is the weaknesses to a software system that create loop holes for failures to occur. They can be as a result of poor requirement specifications, inadequate design and implementation procedures in the several components in the system. They are categorized into two types: Flaws which are design-level problems and Bugs, which are implementation-level problems.
- Software safety analysis - it is a systematic approach to identifying, analyzing, tracking, mitigating and controlling software hazards and hazardous functions to ensure safe operation within a system.
- Software hazard criticality matrix - is a measure of potential risks of a software component. It is used at the Computer Software Configuration Item level.
- Software Requirement Specification (SRS) - defines the software performance and requirements that the system must attain to be accepted and implemented as required.
- Safety Requirements (SRs) - are guidance to system analysts on which and how to evaluate the system functions and their safety-critical functions. Safety analysts analyse the safety requirements of a system and can be included in the development team to help in the generation of requirements, design and development until the testing phase.
- Security vs. Safety - Both of these two terms, if not achieved, can lead to conflict with important mission requirements. Security has its focus on privacy of systems and malicious activities that can compromise the intended use of the system as well as attack from internal and external environment. Software safety is concerned with events that lead to failure of a system.
- Reliability vs. Safety - Reliability describes the probability that a system performs its intended function for a specified duration of time with some set environmental conditions. Safety is the probability that conditions that can lead to a mishap do not occur. The main purpose of having safety and reliability analysis is to minimize the probability of failure occurring. Reliability is focuses on having a system which is failure free, whereas safety is concerned with making it mishap free.

Safety-critical software handles the hazards that are identifies by the system analysts during the safety analysis for the developers to make the system safe, risk-free and fail-safe[5]. The results from the faults in the critical systems can be catastrophic and result into injury or harming the people operating the system or the environment. Software failures occur due to logic or design errors which are a result of use of incorrect requirements during the design of the software or

coding errors that deviate from the requirements during the system development phase. They also occur when the delivered service no longer complies with the specifications as expected [6]. This occurs both to hardware and software system failures that are caused by the errors in the system that affect the delivery of service by system components. Errors can lead to a system failure due to failure of multiple components at the components interact with one another. Failure of one component might lead to a series of faults in a system [6]. There are different ways in which safety analysis techniques address these problems [7]; it depends on the methodology used and the target of their analysis. Below are some of the used to address the different aspects of a problem:

- Hazard identification
- Demonstrate the absence of specific hazards
- Determine the impact resulting from hazards
- Finding the root cause of hazards
- Evaluate the efficiency of hazard controls
- Identify safety design criteria that will eliminate, reduce, or control identified hazards.

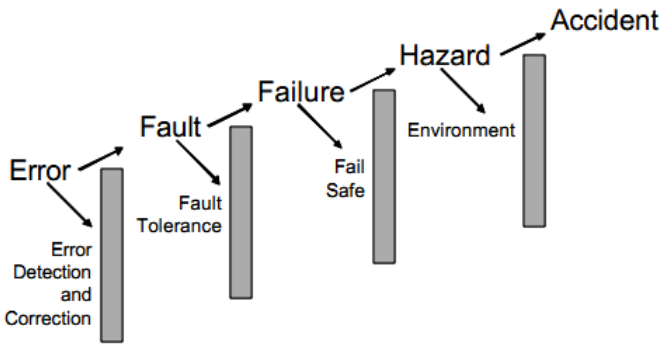


Figure 2.1: Accident Model [8].

2.1.2 Accident Model

Errors and mistakes made in requirements specification, design, development, or testing can be the root cause of an accident. These errors may arise anywhere in the project life cycle. By expectation, all errors should be detected and corrected before the system is released but in reality, some errors might not be detected and efforts to correct them might lead to new errors cropping up, resulting in defects in the final product [8]. However, even well designed, manufactured, and tested to the required standards, a perfectly functioning system may still contain logical errors if the requirements used to develop the system are incorrect. Other faults may arise due to transient events, such as single event upset, or physical failures of

correctly designed components. Fault tolerance design techniques may be used to contain the faults, but some faults may propagate to the next stage and result in system level failures - a loss of functionality. At this stage fail-safe design techniques may again halt the process, but some failures may not be contained and will place the system in a hazardous state - a state that has the potential to result in an accident. The final factor that determines whether or not an accident occurs is the condition of the surrounding environment. A hazardous condition coupled with "good" environmental conditions may not result in an accident, and is only an incident. If "bad" conditions are present the result will be an accident. Thus, an error may develop into a fault, a fault may result in a failure, a failure may place the system in a hazardous condition, and a hazardous condition may result in an accident.

2.1.3 Safety Processes and Software

Safety Processes and Software are concerned with identifying and controlling ways, in which systems may behave(fail) in order to be unsafe. The processes are normally structured around the idea of hazards which are situations that can lead to injury or loss of life [9]. The early phases of the safety process involve identification of hazards determining the associated risk. If the risk is considered unacceptable, then the design must be changed to reduce the effect of the hazards or to mitigate the consequences of the hazards involved. The results of such analysis are often referred to as derived safety requirements.

Once hazard analysis has been undertaken, design and implementation can continue, with the aim of producing a system which meets all its requirements. Once the design and implementation is complete and the system is integrated, analysis and testing is performed to validate that the system meets its specified requirements and the results demonstrate that the system is safe. The safety case complements the evidence by providing the arguments which show why the evidence is considered suficient to demonstrate safety of the system. Most safety standards are concerned with how engineered systems can fail and give rise to hazards. Software can only "fail" due to systematic causes, e.g. requirements or design errors, and the usual analysis processes do not apply [9]. It's is therefore necessary to have design processes which reduce the likelihood of introducing such flaws into software.

Software safety standards are different but on a closer look, there are many areas of commonality involved, for example, the concept of hazard, the aim to reduce risk, and mitigate the consequences [9]. An attempt has been made to rationalize these processes and to show that there is significant commonality in these standards. However, there are different views on how they treat systematic issues, and this has led to a number of authors questioning the notion of SILs and the soundness of the guidance in the standards [9].

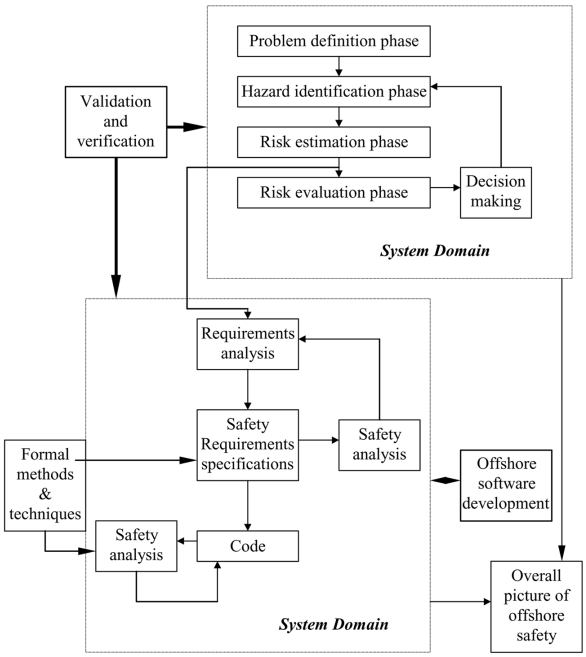


Figure 2.2: An example of processes involved in developing a Safety Critical Software System [10].

2.1.4 Software Safety System

Software System Safety optimizes system safety in the design, development, use, and maintenance of software systems and their integration with safety critical hardware systems in an operational environment [11].

- Safety critical systems are designed following the requirements specified in a timely and cost effective manner.
- All hazards that are associated with the software are identified, evaluated and resolved or reduced to an acceptable level thought out the system development lifecycle.
- The complexity of safety critical software components and interfaces are reduced so as to reduce the probability of human error occurring.
- Failure modes such as hardware, software, human and system are addressed in the design of the software.
- Safety issues are addressed at all levels of development and testing.
- Software safety systems are designed for ease of maintenance and modification or enhancement.

Safety rules should be considered during development of safety critical software to prevent events or actions that can initiate potential hazardous events to happen. The software system should be able to detect unsafe conditions or commands and originate functions or procedures to put the system in a safe state. There are guidelines and requirements that should be followed when performing the safety activities during the early development phases.

2.1.5 Software Concept and Initiation Phase

This phase involves system level requirements and design development. The phase is concentrated on the subsystem level and included the creation of important software documents and plans that determine how, what and when software products will be produced or activities will be performed. There are several documents involved:

- System safety plan - Include software as a subsystem to identify tasks.
- Software concepts document - Identify safety critical processes.
- Software management plan and software configuration management plan - Coordination with system safety tasks, flow down incorporation of safety requirements.
- Software security plan - Security of safety critical software
- Software quality assurance plan - Support of software safety, verification of software safety requirement, and safety participation in software reviews.

2.1.6 Software Requirement Phase

Identification and elimination of errors from the early beginning of software development saves a lot of resources involved in correction of software faults and errors. Hence it is important to implement the software requirements correctly from the initial stages. Software developers must develop complete and correct requirements and code and ensure that they develop fault-tolerant designs which are able to detect and mitigate faults on the software system to operate as desired. Software safety requirements can be top-down, that is, derived from system requirements, and/or bottom-up in which case they are derived from hazard analysis. Software safety requirements are derived from the system and subsystem safety requirements developed to mitigate hazards identified in the preliminary, system and subsystem hazard analysis [11].

Thus software requirement document contains the basic requirements for the software system including the safety related requirements. Software Interface Specification identify, define and document interface requirements internal to the subsystem in which the software resides and between the hardware system and operator interface, subsystem, and program set components and operation procedures.

2.1.7 Development of Software System Safety Requirements - SSSR

SSSR are derived from several sources which are grouped into two:

- Generic software safety requirements are set of requirements that can be used in several software programs and environments and they solve similar kind of problems. They are found as standards and guidelines they should follow to achieve certain goals.

- Specific software safety requirements are unique functional capabilities of a software system that are derived from top-down analysis of system design requirement specifications to pin point the safety critical system functions. They can also be derived from the preliminary hazard analysis that looks into the system from the point of view of system hazards. The hazard control features are identified and specified as requirements. The last way of deriving specific software safety requirements is through bottom-up analysis of design data which includes data flow diagrams, charts, FMEAs, fault trees etc. to find the causes of hazards and their mitigation processes thereby specifying them as requirements.

2.1.8 Software Safety Lifecycle

Software safety lifecycle requirements in 61508 standard requires the developers to uphold an ongoing commitment to the users of the product to ensure the continued safe operation of desired system systems. The plan for validating the software safety considers details such as when the validation will take place and who shall carry out the validation. Relevant modules of the EUC operations are identified, and the technical strategy for the validation is also considered. The measures and procedures that shall be used for confirming that each safety function conforms with the specified requirements for the software safety functions and the specified requirements for software safety integrity are also specified [1]. Development of a component of a safety system requires commitment not only to stipulated development methods, but also to a thorough and careful approach from the initial stages of development all the way to the maintenance stage. Developers are expected to follow a lifecycle similar to the one illustrated below when using or designing the components of a safety system, beginning with a risk analysis to determine the Safety Integrity Level (SIL) required.

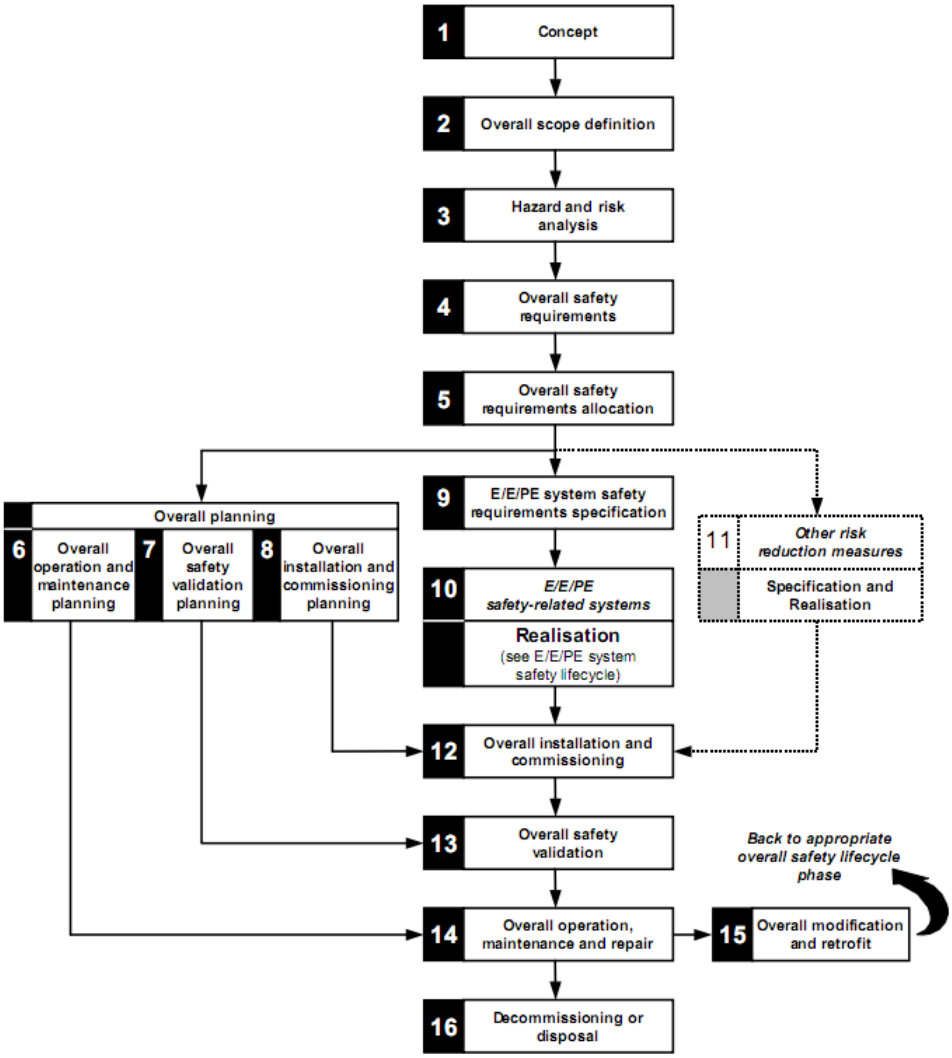


Figure 2.3: Software Safety Lifecycle [1]

2.2 State of the Art - Safety Analysis Methods

This section describes some of the most known safety analysis methods. The two methods included in the procedure will be described in the next chapter.

2.2.1 Preliminary Hazard Analysis (PHA)

The PHA is performed in the concept phase of the project to identify safety-critical areas, identify and evaluate hazards, and identify the safety design and operation requirements needed in the concept phase. [12]. This method is a semi-quantitative analysis that is performed to identify the potential hazards and failures that lead to an accident by ranking them in order of their severity and defining the mitigation process and follow-up actions [13]. Since PHA is performed early in the process, it is high level method. Problem solving of the hazards discovered is not discussed at this point, but high level suggestions are proposed with the main focus being on

identification of hazards [3]. This process uses a general overview of components that will be used in the final product but some of the components are known prior to the design of the system and need to be analyzed in order to find the hazards that they may cause to the system. The method is used as the initial step for a detailed risk study and analysis in the early phases of a project. PHA identifies the hazards to a software system and there after perform more rigorous than the risk analysis process.

The system analysts have to consider the hazardous components with the safety related elements and the constraints that surround them. The main steps involved in PHA are hazard identification, consequence and frequency estimation followed by ranking of the risks and their follow-up actions [13]. PHA is applicable to all type of systems and subsystems where high level analysis is carried out with the suggestions on how to deal with the detected hazards, which are later converted to requirements. This method enables system analysts to perform hazard analysis in the early phases of system development without necessarily requiring much expertise and resources. Thus, design can be altered depending on the results of this analysis. The downside of this method is lack of detailed analysis of specific components because the team of system analysts only have a general view of the components and their dependencies. There is a likelihood of overlooking some events which might seem insignificant in the initial stages of system development and turn out to be the top event. Thus, this method should not be used alone.

The results of the PHA are usually recorded in a PHA worksheet or a computer program. Below is a sample PHA worksheet.

Preliminary Hazard Analysis (Pressure Cooker)

Hazard	Cause	Effect	Probability of Accident due to Hazard	Corrective or Preventive Measures
Shock	Faulty wire insulation creates circuit to ground through operator when operator touches cord.	Mild shock to electrocution depending on the overall resistance to current flow through the person's body. The overall resistance would depend on factors such as the resistance of the person's shoes, whether or not his or her fingers were wet, and the condition of the insulation.	Remote	Use insulation that is very resistant to deterioration. Use a grounded cord (3-prong plug). Only plug the pressure cooker into outlets that are equipped with a ground-fault circuit interrupter.
Fire	Sparks are generated near a flammable material when current passes from the cord to another object at a point where the insulation is faulty.	Significant damage to system and surroundings.	Extremely remote (A fault must be present in the insulation, sparks must be generated, and a flammable material must be located very close to the cord. The probability that all of these conditions will exist simultaneously is very low.)	Same three used for shock. Keep flammable materials away from system.
Burn	Person touches hot pressure cooker surface or hot materials inside pressure cooker. Steam from safety valve burns person.	First or second degree burns depending on how long the person's skin is in contact with the hot surface or material.	Reasonably probable	Use hot pads if the pressure cooker must be touched. Keep pressure cooker out of the reach of children. Put a cover on the safety valve to spread the steam out so that it is not concentrated enough to burn the skin
Explosion	Thermostat and safety valve fail, and no one notices that the pressure gage indicates "danger."	Sever injuries or fatalities. Loss of system. Damage to surroundings.	Remote	Use only high quality thermostats and safety valves. Use more redundancies. (Example: Two safety valves)

Figure 2.4: Example of Preliminary Hazard Analysis of a pressure cooker [14].

2.2.2 Failure Mode and Effects Analysis (FMEA)

This method is used for hazard analysis during product development and operations to determine how failures occur, their frequencies and how likely it is that they can be detected. The FMEA method is focused on individual components in the system; how they can fail, the effects of these failures and how they can be handled. Rather than focusing on what can be done after the failure has occurred, FMEA encourages the developer to focus on how to prevent the failure from happening in the early phases of system development.

FMEA determines the effects of component failure on the operations of a system and categorizes the failures into levels according to their severity and probability. Usually, the failures are given a number from 1 to 10. The product of these two values equals the risk priority number (RPN) for that failure mode. The RPN is used to prioritize the failure modes. Failure modes are the possible events of errors occurring to a process or an item and have a harmful effect to the system, producing an undesired result while effect analysis is the process of analyzing the consequences and results of the failures and how they are mitigated [15].

This method helps the system analysts identify potential failures of a system based of previous experiences from similar systems. This process aids in design and development of safety critical systems. System FMEA focuses on global system functions, design FMEA focuses on components and subsystems, process FMEA focuses on manufacturing and assembly processes, service FMEA focuses on service functions and software FMEA which focuses on software functions [15].

2.2.3 Fault Tree Analysis (FTA)

FTA is a failure analysis that uses Boolean logic to analyze the undesired states of a system using graphical representation of failure analysis. It models and analyzes each failure event to find its root causes and provide risk assessment using cutsets (qualitative) and probability (quantitative) in the process of safety systems. A FTA is composed of logical diagrams that display the structure of the system and provides a logical framework for expressing combination of component failures, undesired events, unintended events or states that can lead to system failure [16].

This method is mainly used in analyzing engineering processes where hazards are evaluated in a top-down approach starting wit the potential event at the top and followed by an analysis which determines how the top event is caused by individual or a group of low level failure events. AND, OR and M or N logic gates are used to connect the top event with its cause events which form a tree with a hierarchical structure below the Top Event [17]. However, FTA can be a costly and cumbersome process, hence it is better implemented in subsystems to reduce the errors involved with the work and to reduce the amount of time used. One of the main limitations of FTA is the expense involved with finding the causes of the failures.

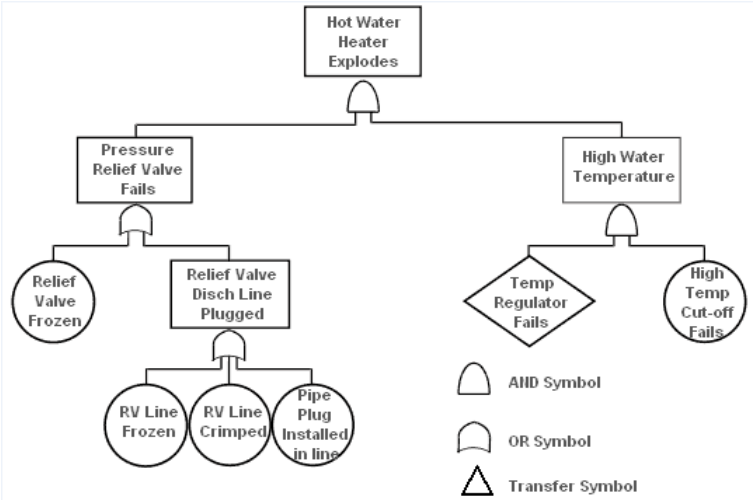


Figure 2.5: Example of a fault tree diagram [16].

2.2.4 Intent Specification

This method adress specification design in psychological principles of how humans use specifications to solve problems, as well as on basic system engineering principle [18]. It integrates the formal and informal aspects of software development and enhances their interactions. Intent specifications integrate human-centered automation into the system requirements specification. System specifications are organized along a vertical dimension using the intent abstraction and two horizontal dimensions using two types of abstraction: refinement and decomposition abstractions [18]. Horizontal dimension allows analysts to focus on the intent levels of the model. It uses two types of abstraction mechanisms: Parallel decomposition which is used to separate units into components of the same type and Refinement abstraction which is used to break down functions into more detailed steps.

This method has a couple of benefits: the ability to solve problems and support system development processes by helping the understanding of the requirements of the system for the designers and developers. The abstraction forms enable the integration of formal and informal aspects of requirement specification which are geared to development of high quality and evolvable safety critical systems. Intent specifications emphasizes the traceability issues between user requirements and the decisions made during the design process, thus offering an opportunity to change and upgrade the software product. However, this method requires knowledge about the system architecture and insufficient traceability in the specification requirements makes the system upgrading process tedious and in large systems almost impossible.

Horizontal dimension allows users to focus on the intent levels of the model. It uses two types of abstraction mechanisms:

- Parallel decomposition - to separate units into components of the same type.
- Refinement - breaks down functions into more detailed steps.

Intent specification on horizontal dimension is further broken down into four parts [19]:

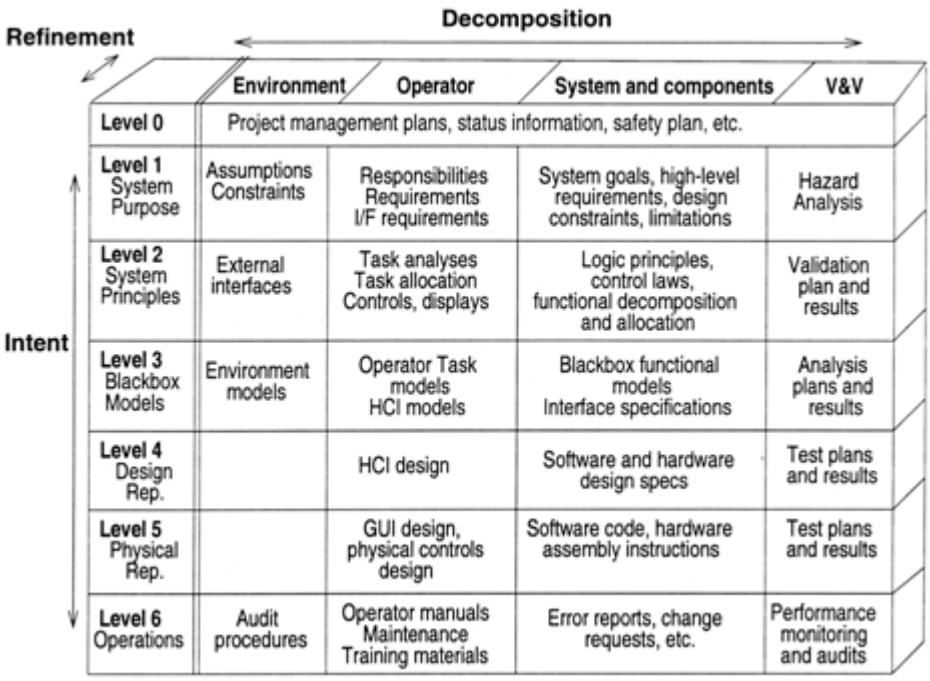


Figure 2.6: Intent Specification: Mapping between levels provide relational information necessary to reason across hierarchical levels [19] [2].

- Environment - factors that affect attainment of system goals and specifications to be used in hazard analysis. It contains information about the characteristics of the environment that affect attainment of goals and design constraints.
- Operator - these are defined system users that are considered in human-computer design. This help the system designers to focus not only on the technical aspects but also give the same attention to the human design factors.
- Systems components - system decomposition into subsystems or components.
- Verification and validation - this level evaluates the system as whole.

The vertical dimension (Intent dimension) has five levels which provide the reason for the information level below each one of them, thereby providing traceability of high-level system requirements and constraints to the basic building blocks of the system [19] [2].

- System purpose - enables engineers to determine system-level goals, constraints, priorities and trade-offs. Goals have to be refined into testable and achievable high-level requirements. Design constraints have to be considered when creating the requirement specification since they are restrictions on how the system can achieve its purpose. Environmental requirements and constraints may lead to restrictions on the use of the system to determine that the requirements hold for the system being designed. Assumptions must be clarified at all levels of the intent specifications to determine the basis for the design. The functional

operation accuracy of the system depends on the assumptions made in the design and hazard analysis processes. Limitations have also to be considered in relation to the assumptions made since they are hazards that could not be done away with during the design process. Thus they are considered as accepted risks to the system.

- System design principles - used to guide designers to reason the physical principles along which the design is implemented. These principles are linked to the related higher level requirements, constraints, assumptions, limitations and hazard analysis. Assumptions used in defining the design principles are specified at this level.

- Blackbox behavior - enables designers to understand the logical design of the whole system and interaction between the system components. This level specifies the system components and their interfaces, including the operators of the system. Each system component behavioral description and each interface is refined along the horizontal dimensions. The system environments description includes all assumptions of all external components, interfaces between the system and the external components made in the first level. Boundaries of the system are abstractions depending on the external components that interact with the system, hence, they are mostly placed between the system and its environments. The behavioral descriptions are inputs, outputs of all the components and their relationship depending on the kind of objects, variables and functions they share. This level also describes the operator requirements, interface requirements and testing requirements for the system functionality.

- Design representation - this level focuses on individual component design and implementation and contains the design information. This is where specifications include the physical and logical implementation of components. It contains the design packages used and the pseudo code of the design. Other necessary information contained in this level includes hardware description, user manuals, verification requirements, human-computer interface design specifications.

- Physical representation - the lowest level which entails the physical implementation of the levels above.

CHAPTER 3

Evaluation of the Procedure

This chapter describes the two methods used in our procedure, and a discussion of how best to combine them to achive the desired effect.

3.1 Misuse Cases

Misuse cases apply the concept of negative scenarios for a situation that the system's owner does not want to occur in a use case context [20]. Misuse cases ensure that safety and security requirements are represented in the same manner as the functional requirements, thus enabling the designers to focus on safety issues from the initial stages and raises the propability of capturing threats that can occur to the system.

Misuse cases can be used to model software safety scenarios in the development cycle of a software product. Use case diagrams are used in eliciting system requirements, preferably used for functional requirements, but they also offer some support for non-functional requirements such as safety and security threats to the system [21]. Problems with use case based approaches to requirements engineering include oversimplified assumptions about the problem domain and premature design decisions [22].

Use cases are used to determine, communicate, specify and document requirements [21]. A use case generally describes behavior that the system owner wants the system to show [22]. They help in defining the architecture of the safety-critical system and the components interrelation. It is done by looking for possible threats to a software system's functions, processes, data, transaction and scope thus they become misuse cases. Actor and use cases are defined for the better understanding of the system's functionality and misuse cases are also established to indicate the hazards that can affect the system.

This further leads to construction of use case model and creating of textual misuse case tables that give more detailed information about the use case and its corresponding misuse case. These threats can be caused by component failure or user error so they should all be modeled as misuse cases with relationship to the use cases they threaten using words like 'threatens' in the relationship link. Finally, use cases for mitigating the misuse cases should be created to counter the objective of the misuse component.

Textual misuse cases are more detailed compared to graphical misuse cases and are in two forms [2]:

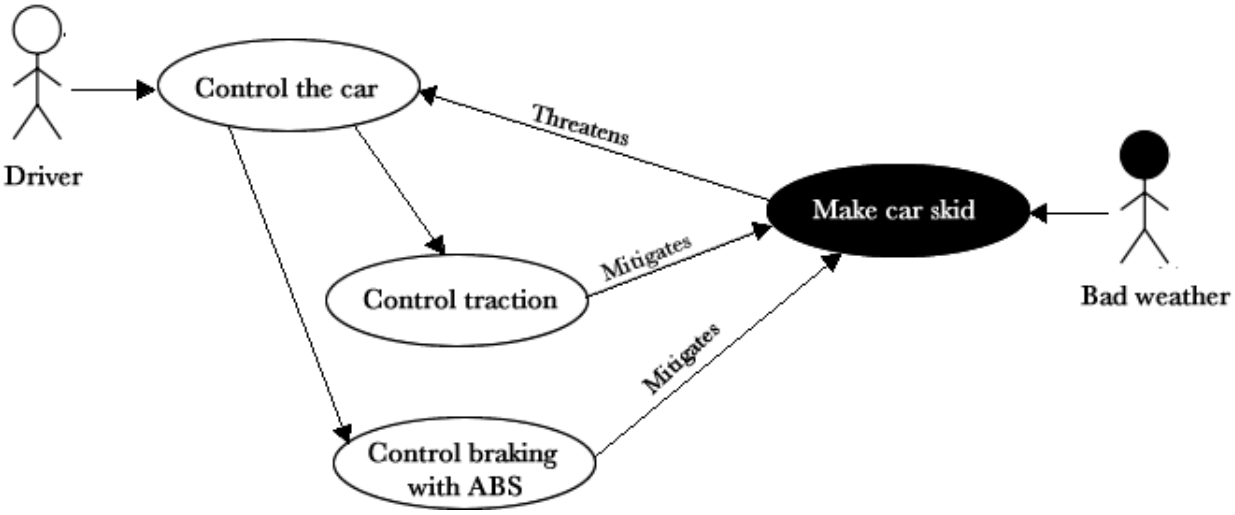


Figure 3.1: An example of misuse case diagram for car safety. From Alexander(2002) [23].

- Lightweight misuse case - takes the approach of embedding the description of the misuse case within a regular use-case template and have an extension field called threats [22]. It is represented by a table for user intention, system response and safety threats contain a simple description of the activities that cause threats to the system.
- Extensive misuse cases - support detailed description and analysis of safety threats extensively. In an extensive misuse case description, the fields Name, Summary, Author, and Date retain the same meaning as in regular use cases [22]. The Basic and Alternative path fields describe the events that occur to limit the safety of the proposed system.

3.2 HAZOP

Hazard and Operational analysis is a systematic investigation of design representations which are conventional, descriptive models of a system's design [24]. The method focuses on deviations from design intent. It is not a top-down or a bottom-up technique as many other methods. The analysists identify a set of study nodes in the design; focus points to which they apply guide-words to the system intentions. With the guide-words, the HAZOP team identify the possible deviations in the system. The guide-words are as follows [25]:

- NO OR NOT
- MORE
- LESS
- AS WELL AS

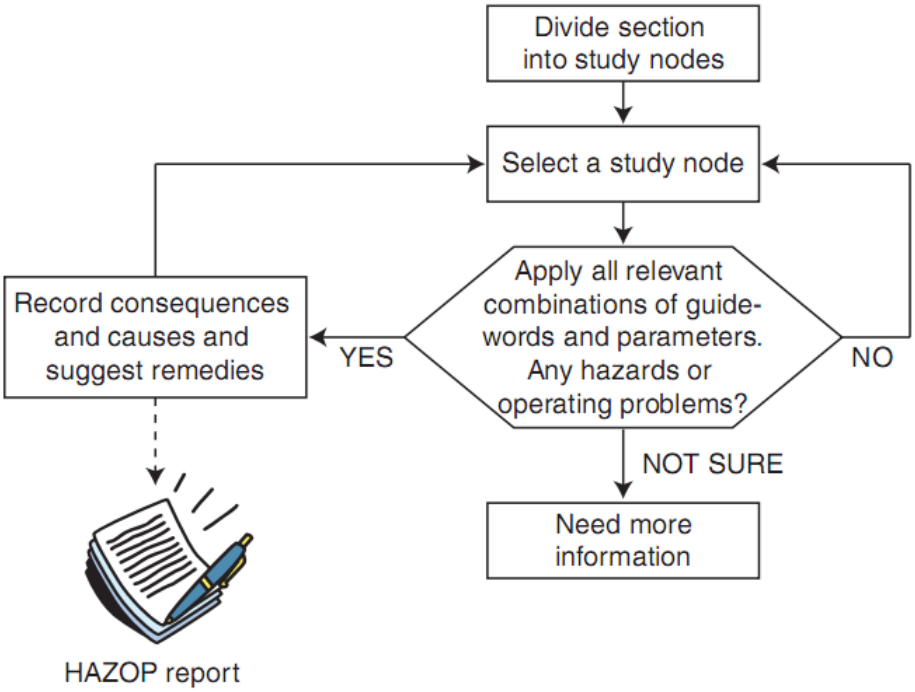


Figure 3.2: An illustration of the HAZOP process. From Rausand(2005) [26].

- PART OF
- REVERSE
- OTHER THAN
- EARLY
- LATE
- BEFORE
- AFTER

Marius et al [24] lists the following steps in a HAZOP study:

- Identify each entity in the design representation
- Identify attributes (physical or logical properties for each entity)
- Investigate deviations from design intent by applying guide words to attributes
- Investigate, for each deviation, the causes and consequences.

For use in software safety analysis, the standard guide-words are often not applicable and a different set of guide-words are often used. The following guide-words are more suited as they reflect software behaviour. ??INSERT REF??

- TOO LATE

- NEVER
- UNEXPECTED
- SPORADIC
- TOO OFTEN
- INCOMPLETE
- INCORRECT
- UNCHANGING

3.2.1 Definitions

- Hazard - is an operation that can cause a catastrophic accident, to a system or its users and can result in injury or death.
- Operability - any operation inside the designed system that would cause a shutdown that could possibly lead to a violation of environmental, health or safety regulations or negatively impact operability.
- Harm - Physical injury or damage to the health of people or damage to property or the environment. Harm is the consequence of a hazard occurring and may take many forms: patient or user safety, employee safety, business risks, regulatory risks, environmental risks, etc [27].
- Risk - Combination of probability of occurrence of harm and the severity of that harm. In a strict sense, "risk" is not always explicitly identified in HAZOP studies since the core methodology does not require identification of the probability or severity of harm. However, risk assessment teams may choose to rate these factors in order to further quantify and prioritize risks if needed [27].

HAZOP is a structured and systematic technique for hazard analysis and risk management. It identifies potential hazards in a system as well as operability problems in a systematic approach. Operability problems should be identified to the extent that they have the potential to lead to hazards. Although hazard identification is the main focus, operability problems should be identified to the extent that they have the potential to lead to hazards, result in harming the environmental or have a negative impact on operability of a system.

3.2.2 Usage of HAZOP

HAZOP is commonly used when assessing hazards in equipment, facilities and processes that are safety critical and required to operate in optimum safe states at all times to avoid failures that result to injury or harming humans and the environment they operate in. HAZOP is able to:

- Asses design of a system and capability to meet user specifications and safety standards.
- Asses the environment in which a particular system operates in to ensure it is situated appropriately, supported with all the required components, serviced and contained regularly.
- Asses operational controls, sequence of events, procedural controls where human interaction is involved.
- Assessing operational modes including start-up, standby, normal operation, steady and unsteady states, normal shutdown and emergency shutdown.

Some advantages of using HAZOP include:

- Useful when addressing hazards that are difficult to quantify for example hazards related to human error, hazards that are difficult to detect, analyze, isolate and predict.
- It uses brainstorming method where the team members are from several fields of expertise.
- More simple and intuitive than other commonly used risk management tools.

This method has also some limitations associated with it, they include:

- It is difficult to identify hazards that are found during the interaction between different parts of a system or when different processes are interacting together.
- There is no ranking or prioritization of risks that are identified, but the team may choose to do so if it is desired.
- There is no means of determining that the selected mitigation for hazards identified will actually work or be effective, hence it is necessary to use HAZOP with other risk management tools.

3.2.3 HAZOP Process

The HAZOP team is made up of individuals with different backgrounds, and with different expertise that bring collective brainstorming during HAZOP sessions. This aids in bringing in new ideas during the thorough review of the desired system under evaluation. The HAZOP team focuses on specific portions of the process called "nodes" that are identified before the study begins. A process parameter is identified: for instance the sensing of heat level by a heat sensor in a boiler system and an intention is created for the node under consideration. Thereafter, a series of guidewords are combined with the parameter "temperature" to create deviations [28]. For example, the guideword "NOT" is combined with the parameter heat sensor to give the deviation "heat sensor not working". The HAZOP team then focuses on listing all the credible causes of a "heat sensor not working " deviation beginning with the cause that can result in the worst possible consequence the team can think of at the time. Once the causes are recorded, the team lists the

consequences, safeguards and any recommendations that should be implemented in the system [28]. The process is repeated for the next deviation and so on until completion of the node then the team moves on to the next node and repeats the process.

3.2.4 Selection of Guidewords, Parameters and Deviations

The HAZOP process lists the possible failures that can lead to a hazard. A list of guidewords is combined with the parameters to produce deviation. However, not all combinations are useful since some of them are not meaningful or applicable. The application of parameters will depend on the type of process being considered, the equipment in the process and the process intent. In some instances, a parameter is evaluated for every node and consequently, a node should be evaluated for the selected parameters and only recorded when there is any association with the parameters. The most common parameters include: Flow, temperature, pressure, composition, phase, level, relief, instrumentation, sampling, contamination and many more.

Deviations are the also referred to as consequences and should be considered if there is a relevant cause that made the deviation occur. The team should be able to make good judgment when determining what events have a low probability of occurring so that credible causes are not overlooked [28]. Deviations have three main causes according to [28] :

- Human error which is acts of omission or commission by an operator, designer, constructor or other person creating a hazard that could possibly result in a release of hazardous or flammable material.
- Equipment failure in which a mechanical, structural or operating failure results in the release of hazardous or flammable material.
- External events in which items outside the system being reviewed affect the operation of the system to the extent that the release of hazardous or flammable material is possible. External events include upsets on adjacent units affecting the safe operation of the unit (or node) being studied, loss of utilities and exposure from weather and seismic activity [28].

Risks to a system can be ranked in order of their occurrence to help in determining if their consequences result in a hazard or operability problem. If the team concludes from the consequences that a particular cause of a deviation results in an operability problem only, they note it down and move to the next cause. If the cause results in a hazardous effect, then ways to mitigate the cause should be considered and noted down. The desired system should be designed to prevent catastrophic events from occurring. The system that is desired should be able to detect and give an alarm to the operators in case there is initiation of the causes of the hazard. Finally, the systems should be able to mitigate the consequences that lead to hazards and operability problem [28].

3.3 Result from the Specilization Project

The following quote is the main results from the specialization project [2] done as a preproject for this thesis.

> The main objective of this project was to analyze methods for early identification of safety requirements in safety-critical software systems. This was achieved by combining two safety analysis methods; Misuse cases and HAZOP to improve the process of identifying safety threats to a steam boiler system. We performed a thorough familiarity process of the steam boiler using the functional requirements written by Tor Stålhane and Tormod Wien which was the high end system requirements that were agreed upon by ABB who were the among the stakeholders. We were able to identify the control functions of the boiler system together with its physical environments where it operates to understand the relationship among all the components involved.
>
> Through the use of safety analysis methods, we identified hazard to the boiler system and events that can lead to failure of the system by determining their causes and consequences, and conditions under which they can occur. Hazard and risk analysis provided use with a platform for producing safety requirements for the system that were allocated to components of the boiler system. We also sought to find other risk reduction measure that could be implemented in the system to make it more fault tolerant. The number of barriers we recommended in order to prevent risks from occurring and when they occur, barriers could prevent or reduce their consequences and effects. We also used the guidance IEC 61508 [1] on getting information about safety regulations when developing safety-critical systems.
>
> Overall, we were able to achieve the major objective in our case study: Identify the safety requirements early to enable software developers and designers have a clear understanding of the safety issues surrounding the boiler system. Hence, they would be a reliable software for controlling the system with all the hazards under control.

With our case study, we explored the idea that the results from a misuse case analysis could be enhanced by a subsequental HAZOP analysis in order to identify safety requirements. How best to combine these methods will be discussed in the next section.

3.4 Combinding the Methods

In short, our procedure consists of starting with a misuse case analysis to find threats to the system functions and new system functions that mitigates these threats, then use the system functions as study nodes for the HAZOP. This section describes an in-depth evaluation of how to best combine the two methods, based on the case study in the specialization project and our thoughts on improvement.

3.4.1 Misuse Cases

A key factor to misuse cases is that they identify both functional and non-functional requirements, and represent them in the same manner. This enables the analysists to capture safety requirements from both worlds.

Graphical misuse case illustrates a sequence of actions, including variants that a system or other entity can perform [29]. It contains a sequence of actions and activities performed by a misuser that can cause harm if the sequence is allowed to finish. This is represented through an illustration showing the actors who specify a role played by a user or a part of a system that interacts with the subject [29]. A misuse case further describes a set of actions that are performed by a user or system resulting to observable results. The relationship called "threats" targets a use case and the relationship "mitigates" is a solution to the misuse case[22], [30].

Textual misuse case contains the details of use cases which are usually captured in the associated textual templates. Templates are important because they encourage developers to write clear and simple action sequences [30]. Like ordinary use cases, misuse cases may be described textually using misuse case templates. Two ways of expressing misuse cases textually have been suggested: lightweight descriptions and extensive descriptions [31]. A lightweight description is embedded in an ordinary use case template and extends it with additional entries for threat by the misuser [22] while extensive description gives a more detailed approach to capturing of information required by developers. It supports detailed description and analysis of security threats. New fields are added in the textual misuse case to enhance the description of misuse cases of a system into a more detailed way [22].

3.4.2 From Misuse Cases to HAZOP

Our procedure includes both misuse case diagrams and textual misuse cases. In the textual misuse cases there is a column with threat decriptions to the use cases. Thus, the textual misuse cases take the misuse case diagrams one step further, and also helps the transition to the HAZOP analysis easier. The HAZOP will be performed on the "System Response" column, how the system respond to each use case. The threat descriptions will then aid the HAZOP analysis to identify what can go wrong. After the HAZOP is performed, the analysists should go back to see if there are any threats in the textual misuse cases that have not been analized.

When performing hazard analysis using the two methods, we begin with graphical misuse cases to identify the hazards that are present in the proposed system. The misuse cases are extended from the normal use case diagrams where they are viewed as threats to the systems functions but are mitigated with proposed solutions to main hazards. We move a step further to create textual misuse cases which have a more detailed description of the system and can be in lightweight form or extensive form where they describe the hazards in to more detail. In our case, we used the light weight textual misuse case which has four columns including name of use case, system response, threats and mitigation. This allows us to analyze the hazard further identifying the threat and how the system responds. However, this method is not sufficient since it misses the finer details of the hazards that are posed to the system. We combine HAZOP analysis with misuse cases to increase the chance of identifying the hazards in the system. There are a set of guide words that will aid

us in performing this analysis where we apply all the guide words to the system responses of the textual misuse cases.

There are two sets of guide words, one for software and one for mechanical hazards that will aid us in performing this analysis where we apply all the guide words to the items in the system response column of the textual misuse cases. In the HAZOP table, there are columns which include: guide word, consequence, cause, hazard and possible solution. The table is filled in for all the applicable guide words and the team performing the analysis has to ensure that they have exhausted all the possible options of the systems response being investigated. This allows the team to move to the next item on the system response list and the same procedure is repeated throughout until all the hazards have been analyzed.

The final results of combining these two methods is a well described and detailed hazard analysis that enables the system developers to design a system that is fault tolerant, since safety critical systems require hazard analysis to prevent the events that could lead to failure. The use of the two methods also helps us to investigate the hazards level by level, therefore giving a wide range of capturing the hazards that might threaten the system.

3.4.3 Improvements From Specialization Project

One of the issues we found when performing the case study, was the columns *Threats* and *Mitigation* in the textual misuse cases. Since we used the items in the *System Response* column as study nodes in the HAZOP analysis, these columns was outside the main process. It was not formalized in the procedure, and we wrote some safety requirements independantly of the HAZOP analysis.

We are now aware of the issue and have tried to incorporate it into the procedure. One alternative is to write safety requirements from the threats that are left out of the HAZOP analysis. After the HAZOP analysis is performed, look at the textual misuse cases and see if there are some threats that are not covered, and then write the safety requirements.

The second alternative is to perform another HAZOP on these study nodes with the uncovered threats in mind. For example, in the specialization project [2], table 4.2 on page 55, there is a study node named *Check pressure level sensor* with two threats, *Pressure level sensor failure* and *Software failure*. In this case, we could have run two HAZOP analyses on this node, and then have all the threats covered. In our case study, this was hardly an issue. Since the HAZOP analysis almost always covered all the threats in the textual misuse cases, and running a HAZOP for each of the threats would be reduntant and time consuming. We therefore choose alternative one as the best option, and leave it up to analysists to consider wether to run another HAZOP or not. We just have to make sure the analysists are aware of the issue, and to notify them to look at the textual misuse cases after the HAZOP to see if something is uncovered.

Another alternative is to remove the textual misuse cases altogether. The system responses, which are the study nodes in the HAZOP, can easily be identified from the diagrams. As identified in our case study, the threats and mitigation columns was mostly redundant with the results from the HAZOP. The key word here, though, is *mostly*. Since the most important aspect identified in the conducted survey was

hazard coverage rate, the procedure should identify as many hazards as possible, we can thus argue that the extra time with the textual misuse cases are well spent.

We will have this as one of the focus points in the test and interview and discuss what should be done.

3.4.4 Challenges

We have identified several challenges with using this procedure. See section *5.4 Limitations of the Approach* in the specialization project report [2] for a more extensive description of these. We will repeat the main points here.

- The approach is time consuming and has a complex work structure. Making the diagrams, then making the textual misuse cases, then the HAZOP, then, finally, checking if all the threats in the misuse cases have been covered by the HAZOP. This is a lot more extensive than for example a preliminary hazard analysis, which is the method commonly used in the concept phase.
- The procedure focuses on single event failures. Hazards that ocurrs due to multiple events are hard to identify.
- There are no priorities among the identified hazards, although, most likely, some are much more important to handle than others.
- The misuse case diagrams can become quite messy considering the many arrows going back and fourth.
- The HAZOP can produce very detailed mitigations to the hazards, so detailed that they can be irrelevant in this phase of development.

The last point is only our opinion. We will discuss this point with the experts in the field to see if there is anything that can be too detailed at any phase of development when it comes to safety requirements.

3.5 A Walkthrough

This section provides a walkthrough of the procedure, using a small example. The example concept is the same scenario as shown in figure 3.1, taken from Alexander(2002) [23].

3.5.1 Misuse Case Diagram

Consider a car, in which the basic function is to get the driver from point A to point B safely. Figure 3.3 shows how you can loose control over the car by skidding because of bad weather. The misuse case is therefore "Bad weather", and the threat is "Make car skid". How can you help the driver maintain control over the car in bad weather conditions?

Figure 3.4 shows how two new functions are introduced to help the driver; traction control and an anti-lock braking system. We say these functions *mitigate* the threat.

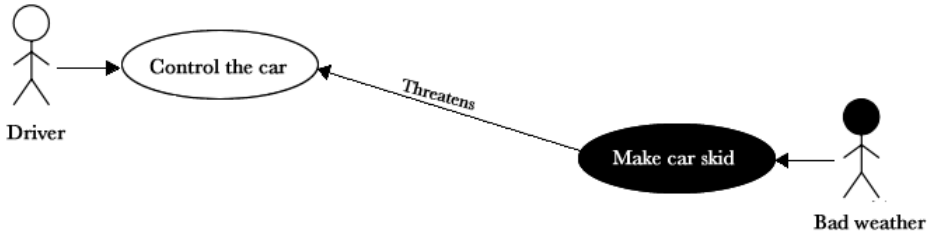


Figure 3.3: Loosing control

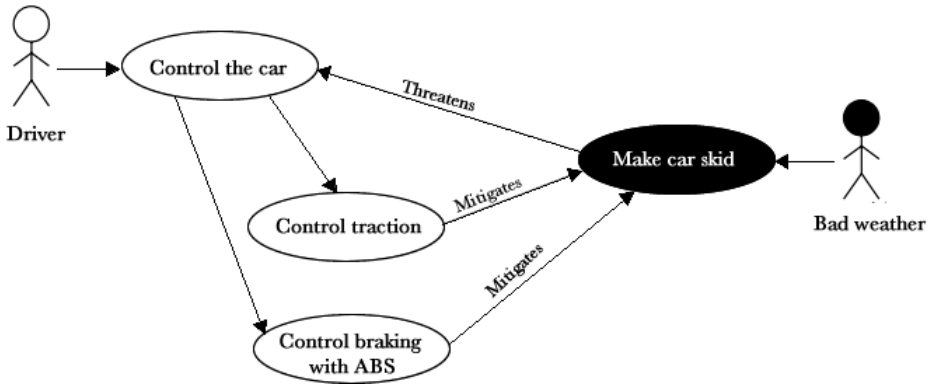


Figure 3.4: Threat mitigated

Threats can then be introduced to the new mitigation use cases as well. For example, a threat could be "ABS breaks too softly". A new use case could be introduced to adress this issue, and so fourth. You could go on endlessly, but this would cause the system to be overly expensive and complicated. It is up to the analysists to evaluate when they have made the system safe enough, and that further expanding the system safety is either too costly or nonessential.

3.5.2 Textual Misuse Case

Following the misuse case diagrams above, the textual misuse cases, see figure 13.1, expands the the system overview to include lower level behaviours.

The threats column should only include situations where there actually could be a risk of danger involved. If you have a chainsaw, "Prevents chainsaw from starting", although an undesired effect, is not dangerous.

3.5.3 Using HAZOP on the Misuse Case Results

Moving on to HAZOP, you apply all the guide words to the system responses. There are two sets of guide words, one for software and one for mechanical hazards. The software version is used whereever software is controlling the process. Preventing the wheels from locking is controlled by software, as it monitors the wheels speed and increases and decreases the brake pressure accordingly. Manual braking, for example, has no involvement with software. Braking is only resulting from the driver pushing the brake pedal.

The following is an example of a HAZOP table on *Prevent wheels from locking*

Table 3.1: Textual misuse case: Car Control

Textual misuse case: Car Control			
Use case	**System Response**	**Threats**	**Migitation**
Control car	Prevent car from crashing	Skidding	Traction control ABS
Control traction	Prevent excessive throttle being applied on bad condition surface	System prevents any throttle applied to the wheels Software error Car component failure	Power the brake lights Backup functionalities Exception handling Component redundancy
ABS braking	Prevent wheels from locking while braking	Speed sensor failure Hydraulic failure Software error	Notify driver and disable system Backup functionalities Component redundancy

while braking. Not all guide words may be applicable everywhere. If a guide word does not make sense for that particular item, skip it.

Based on the results from the HAZOP analysis, the analysts need to evaluate what they need to translate into safety requirements. Some of the hazards identified have too low propability to ocurr, or are too expensive to fix scompared to the gain. From table 3.2 one could argue that adding redundancy, an extra set of hydraulic components, would not be cost-effective, but a robust sanity check in the software would be highly recommended.

Table 3.2: HAZOP: Prevent wheels from locking while braking

Prevent wheels from locking while braking				
Guideword	**Consequences**	**Cause**	**Hazard**	**Possible solutions**
TOO LATE	ABS reduces pressure on the brakes too late, wheels lock	Sensor failure Hydrolic failure	Yes	Add redundancy
NEVER	ABS fails entirely, wheels lock when braking	Software error Sensor failure Hydrolic failure	Yes	Sanity check Add redundancy
UNEXPECTED	ABS releases brakes suddenly and when not needed	Software error	Yes	Sanity check
SPORADIC	ABS releases and applies pressure on the brakes sporadicly and unintentionally	Software error	Yes	Self-test Sanity check
TOO OFTEN	ABS activates at too high wheel speed	Sensor failure	Yes	Self-test Sanity check
INCOMPLETE	Speed value message incomprehensible	Software failure	Yes	Self-test Sanity check
INCORRECT	Wheel speed registered in ABS not same as real wheel speed	Sensor failure Software failure	Yes	Sanity check Add redundancy
UNCHANGING	Wheel speed value does not change	Sensor failure Software failure	Yes	Self-test Alert driver for repair

Part II

Experiment

CHAPTER 4

Experiment - Strategy

??Check for consistency later?? The purpose of this chapter is to define the overall approach for experimenting the procedure. The strategy highlights roles, schedule, risks and how to handle the experiment results. At the end of the chapter we describe the concept used for the experiment [32].

4.1 Experiment Management

This section describes how we organize the experiment. We define experiment roles, schedules and priorities, and describe what to do with the results.

4.1.1 Experiment Leaders

The experiment leaders will be Joshua Maringa and Thorbjørn Sæther. They will assist the participants, helping them understand the procedure and answer questions they might have. They will also record time spent, problems raised by the participants and how easy they found the procedure.

4.1.2 Participants

The experiment subjects consists of undergraduate students at NTNU. The students are currently at their second year, and has the necessary background to understand the basics of the experiment content. They have been taught about use cases, ways software can fail and to some extent how software can cause hazards. See also section 8.3 where we discuss the validity of using these students. They will be rewarded with a wage corresponding to NTNU's policy for participating in such events.

4.1.3 Experiment Schedule

The experiment is performed 10:15 at March 31st, in auditorium S2 at the Norwegian University of Science and Technology. The auditorium has room for 250 people, well above the number of test participants. We reserve 90 minutes, but expect the students to be finished before that.

4.1.4 Risks

There are risks that threaten the validity of the experiment, and these will be evaluated. Inexperience, motivation issues etc. from the test subjects are examples of such threats. Also, the experiment itself might not be valid. The latter is up to us to mitigate to the best of our knowledge. The former we can try to reduce as much as possible, but can never be sure we succeded to the level we expected. In the evaluation chapter after the experiment we will include a section where we consider the validity of it.

4.1.5 Results

The experiment results will be the type and number of hazards the participants have identified. These will be deducted from the PHA, misuse case and HAZOP tables produced during the experiment. The quality and quantity of these hazards will be the basis of our evaluation of the procedure. Other results include time used, problems that occurred and how user friendly the participants found the procedure. This will be recorded by the experiment leaders.

4.1.6 Experiment Result Priorities

There are different levels of priorities of the results. The primary objective of the procedure is hazard coverage rate; how many of the potential hazards of the system are identified.

Time used, occurring problems and easiness to learn are initially considered as lower priorities, but may get a higher priority if found to be having a larger impact than expected.

CHAPTER 5

Experiment - System Description

To test our procedure in an experiment we need a system concept in which there are possibilities of hazards to occur. The system we will use is a steam boiler system, an ABB pilot application which is being carried out by Tor Stålhane at the Norwegian University of Science and Technology and Tormod Wien from ABB Norway. The documents handed out to the participants can be found in appendix A and B.

In the following section we give a short description of the system, while in the next section we give a detailed description.

5.1 Overview

As illustrated in figure 5.1, the system includes two control units and several mechanical parts. Water is fed to a tank through a non-return valve by a pump, a heating element heats the water inside the tank and the steam generated is delivered to the industrial process through a valve. A safety release valve is included in case of too high steam pressure. Two control units, using several sensors, controls the process in order to have the right amount of water, heat and steam pressure in the tank.

5.2 System Detailed

This section describes the concept system in detail. See figure 5.1 for an illustration of how the components are connected.

5.2.1 Functional Requirements

The steam boiler has several functional requirements that ensure that the system performs all the operations required to deliver steam to the industrial process and all the components function as expected. They are listed as follows:

- The steam boiler shall deliver steam at a predefined, constant pressure to an industrial process.
- Steam is produced by heating water using an electric heating element.

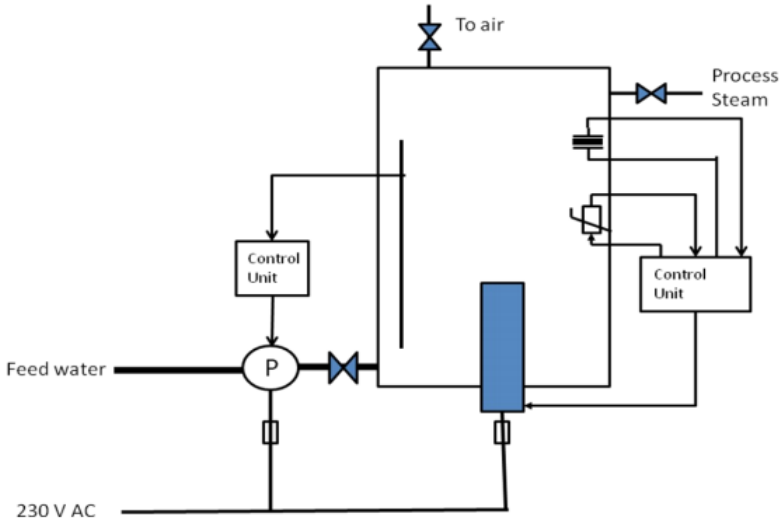


Figure 5.1: Steam boiler concept.

- The steam pressure is controlled by regulating the temperature setting on the heating element thermostat.
- The water level in the tank is controlled by a feeding pump which pumps water into the tank via a non-return valve.
- The safety of the steam boiler is taken care of by a safety valve that opens to air. The release pressure for the safety valve is fixed, based on the strength of the boiler.
- The system shall be SIL2 certifiable.

5.2.2 Components

This section lists the components and their purpose in the system.

- Tank: A water tank designed to withstand high pressure.
- Water pipe: Leads the water from storage to the tank.
- Water pump: Regulates the water flow to the tank.
- Non-return valve: Prevent reverse water flow.
- Heating element: Provides heat to transform water into steam.
- Steam valve: Releases steam from tank to the steam pipe.
- Steam pipe: Delivers steam to the industrial process.
- Emergency release steam valve: In case of too high steam pressure, this valve opens.

- Water level sensor: Measures the water level and sends the value to the control unit.
- Water level control unit: Controls the water pump.
- Temperature sensor: Measures the temperature in the tank.
- Steam pressure sensor: Measures the steam pressure in the tank.
- Steam pressure control unit: Controls the heating element.
- Wiring: Wires connecting the components to each other and to electric power.
- Electricity: Power is provided from a 230 V AC outlet.

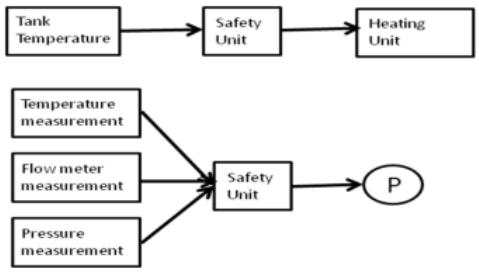


Figure 5.2: Control loops for water level and steam pressure.

5.2.3 Detailed System description

The main objective of this project is to supply steam at a predefined pressure to an industrial process in a way that is reliable and constant as required. The boiler system uses water that is boiled in the tank to produce steam hence this constituted to a safety critical system that contains hazards that pose a risk to the operation of the system. Below we are going to highlight the components and how they relate to each other during the process of producing steam.

- Tank - is the main component of the boiler system that converts water into steam through the heating by the heating element. Water is fed by the water pipe which is supported by the pump to the tank. Once in the tank, heating process starts and it is regulated by pressure, water level and temperature sensors to ensure that the correct ratio of the three is maintained for a constant supply of steam to the industrial process. The tank should be able to withstand pressure that is required by the industrial process and at the same time have the right capacity of water to hold water at the required level for the process to flow.
- Pump - is responsible for supplying water to the tank at the required amounts and at the specified intervals to allow constant supply of steam to the industrial system. The pump gets water from a water source which has to have a steady supply of water at all times and when required, hence water storage tank is

recommended for this case. The pump ensure that water gets into the tank with a higher pressure since the water entry point is at the lower level of the tank and hence it uses the non-return valve to counter the pressure in tank. It works hand in hand with the water level control unit to keep the water level between the predefined maximum and minimum level.

- Water pipe - leads water from the storage tank to the boiler tank via the pump to ensure that water is supplied to the tank. It works together with the non-return valve that ensures water gets into the tank. It should be able to withstand the pressure of water that is caused by the pump since it varies from the storage tank to when it gets in the tank.

- Non-return valve - ensures that water flows only in one direction - towards the tank. It should open to let the water enter the tank and close when no water is getting in. The valve should also be strong enough to block the water since there is pressure applied to it from both ends and if weak, it can get destroyed.

- Steam valve - opens to releases steam to the industrial process via the steam pipe. It should open when there is the required pressure in the tank to go to the process. It opens in one direction towards the process to prevent condensed water in the process from getting back to the tank.

- Steam pipe - delivers steam at predefined pressure to the process, hence it has to be of the right diameter to maintain the required pressure. The pipe should be able to withstand pressure to avoid it from rupturing.

- Emergency release valve - is a regulator of pressure in tank and opens when the pressure in tank exceeds the amount that the tank can hold. It prevents the tank from exploding due to high pressure, hence it has to be functional at all times and open only when the pressure exceeds the maximum pressure and close when normal pressure level us achieved.

- Heating element - functions by supplying heat to the boiler tank, hence heating the water, transforming it into steam. This process has to be regulated at all times to prevent too much heat or little heat in the tank. It works hand in hand with the steam pressure control system that gets the reading of the pressure indicator and temperature indicator for it to control the amount of heat in the tank using the thermal start connected to it.

- Electricity source - supplies the water pump and the heating element with power to pump and heat water respectively. It has 230 volts of alternating current that is required to run all the components of the boiler system that require power for them to run. The source of electricity should be reliable in terms of correct voltage and uninterrupted to ensure the process to run as expected.

- Water level sensor - is responsible for getting the reading of the water level in the tank and communicating with the water level control unit that keeps water level between the predefined maximum and minimum level.

- Pressure sensor - reads the pressure in tank for the control unit to determine if it is within the required range, hence ensuring pressure is regulated to the correct amount.
- Temperature sensor - indicates the temperature in tank and communicated to the control unit hence heat is regulated by turning on/off the heating element.
- Steam pressure control unit - controls the heating element through the data it receives from the temperature sensor and pressure sensor. It is important in the operations of the boiler system since it ensures that the required pressure and temperature is maintained at all times to prevent undesired results.
- Water level control unit - is connected to the water level sensor and the pump to ensure that the required water level is maintained in the tank. It is responsible for controlling the amount of water getting in the tank and at the right intervals.
- Wiring - The electrical components are connected through wires to each other and to an electrical source. Should these wires not be good enough isolated, or have their isolation broken, water could cause hazards should they get into contact. Loose or disconnected wires could also cause components to fail, while other components proceeds as usual, potentially causing hazards.

The boiler system has to coordinate all the components for it to achieve the objective of delivering steam at the predefined pressure to the industrial process, thus it is important to ensure that the components do not fail. To achieve this, a rigorous hazard analysis has to be carried out to ensure the system does not fail, and if it does, ensure the failure can be tolerated. This is a safety critical system, hence it requires safety analysis to be performed before it is implemented.

CHAPTER 6

Experiment - Preparation and Execution

In order to get a basis for evaluating the performance of the misuse case and HAZOP approach we carried out an experiment. This experiment involved computer science undergraduates in their second year of studies as the main test subjects, and we used the system concept described in chapter 5 as the problem to be analyzed.

6.1 Experiment Goals

Our overall goal for this experiment was to get a comparisson of how the misuse case and HAZOP approach would perform compared to the commonly used PHA when analyzing a system concept.

The experiment will also highlight any problems and timesinks that the participants might encounter. This will help us to make the appropiate adjustements in order to make the new method as polished and effecient as possible.

In section 3.4.4 we mentioned some challenges that we have identified with the approach. We will consider these challenges continually through the test phases, take notes of them should we notice any of them occurring, and it will hopefully help us clarify some of the issues.

6.1.1 Overview

- **See how the combined Misuse Cases and HAZOP results compares to those of PHA.**
- **See if there are any substantial difference in time to learn the methods, and if there are any recurring problems.**
- **Compare the time spent to finish the analyses.**

6.1.2 Goal Descriptions

This is a detailed description of the goals and how we will judge the results.

See how the combined Misuse Cases and HAZOP results compares to those of PHA

As mentioned in 4.1.6, the highest priority goal is the comparisson of the results of the two approaches. Our hypothesis is that the misuse cases and HAZOP will provide better results in the identification of safety requirements, with focus on software safety, than the PHA. For the system concept that the experiment participants shall use, we belive that the misuse case and HAZOP groups will have a higher identification rate when analyzing the control unit components, and, to some extent, the components connected to them (sensors, pump and heating element) as well. We will compare the results from each of the two approaches, and, hopefully, the results are substantial enough to make valid conclusions.

The metrics for this goal are the number of unique hazards identified by each group.

See if there are any substantial difference in time to learn the methods, and if there are any recurring problems

Comparing PHA and our approach, we are quite certain that our approach needs more time and explaination to learn. PHA has a simple structure. Our approach has several steps in which the participants need to change focus. From misuse case diagrams to textual misuse cases, and then on to HAZOP. The question is how much more learning and problems this extra complexity adds. As mentioned earlier, one of the reasons we chose misuse cases was because of its close resemblense to use cases. Since use cases are widely known and used in the industry, this would make the approach less dificult to learn. We hope the use cases will provide an easy and fast understanding of the system. Once the guide words in the HAZOP stage are understood, we also hope they will help the participants to identify more hazards.

We will record when the groups go from learning the methods to actually using them. This is, however, no real metric for this goal. Since everything is mostly new to them, they will most likely learn by doing, and it would therefore be wrong to just use this timestamp as the metric. There is no metric for the recurring problems either. We will have to evaluate the severity and importance of these problems separately. For this goal, we will use our own judgement on the results to decide how the methods compare to each other.

Compare the time spent to finish the analyses

Although not the most important when analyzing safety, the time spent is still important to consider.

We will record when the individual groups consider themselves finished with the analysis. We estimate that, considering the extra complexity in the misuse case and HAZOP approach, that the PHA groups will finish earlier, but by much. The metric for this goal is the time the groups spend analyzing.

Other Goals

Another goal that we will also focus on is seeing if the textual misuse cases are used properly (the "Threats" and "Mitigation" column). See section 3.4.3. During the

test we will observe and ask the participants if they understand it correctly, and we will also analyze the experiment results to see if this was the case.

6.2 Handout

It was important to make the handouts as understandable and informative as possible. The students did not know much about hazard analysis on beforehand, and especially not about the methods used. Thus, we needed to include an introduction with examples. The handouts were made together with our supervisor. All the most essential material needed for the students to understand the experiment's purpose and goal was included.

There were two different handouts, one for the PHA groups and one for the misuse case and HAZOP approach - see appendix A and B. The challenge in making them was to keep them short enough so the students would not lose track, but still informative and clear enough to make them understand the subject and task. We went through several drafts with different amount of explanations and examples, until we landed on the ones referenced above. They gave enough background information to hazard analysis, and the students would know how to approach the task. The examples gave a good walkthrough and information that the students could use throughout the analysis as a basis for what they were supposed to do.

One of the more impacting descissions we had to make was wether to include the example of the hazard checklist, see figure 6.1. This checklist could make the students only considering the hazards included here. Still, since we knew they had limited experience in hazard analysis, we decided to include it. It would provide help for achieving the correct mindset for the task. It could also help them identify other hazards as well. During the experiment introduction we made it clear that this sheet did not include all kinds of hazards, and that they should consider other possibilities too.

6.3 Experiment Run

This section describes how we prepared the experiment and how the experiment progressed.

6.3.1 Preparation

We prepared a presentation where we described the hazard analysis concept and the system they should analyze. The presentation is included in appendix C. The presentation was short and only intended to get the students on the right track, and give them an introduction on the subject at hand. It was not meant as a thourough explanation of hazard analysis, as this would take too much time away of the experiment itself.

We also printed enough of the handouts so the students would not lack any material during the experiment.

Common Causes
___Utility Outages
___ Moisture/Humidity
___ Temperature Extremes
___ Seismic Disturbance/Impact
___ Vibration
___ Flooding
___ Dust/Dirt
___ Faulty Calibration
___ Fire
___ Single-Operator Coupling
___ Location
___ Radiation
___ Wear-Out
___ Maintenance Error
___ Vermin/Varmints/Mud Daubers

Software Errors
__ Overflow {Too much work for the CPU}
__Wrong values
__ Wrong timing of message send/receive
__ Loss of data
__ Infinite loops
__ Errors in handling or interpreting data
__ Calculation errors

Control Systems
___ Power Outage
___ Interferences (EMI/ESI)
___ Moisture
___ Sneak Circuit
___ Sneak Software
___ Lightning Strike
___ Grounding Failure
___ Inadvertent Activation

Leaks/Spills (Material Conditions)
___ Liquids/Cryogens
___ Gases/Vapors
___ Dusts - Irritating
___ Radiation Sources
___ Flammable
___ Toxic
___ Reactive

Explosives (Initiators)
___ Heat
___ Friction
___ Impact/Shock
___ Vibration
___ Electrostatic Discharge
___ Chemical Contamination
___ Lightning
___ Welding (Stray Current/Sparks)

Figure 6.1: Hazards Checklists, adapted from Mohr(1993) [14].

6.3.2 Progression

All in all, 53 students turned up. Based on feedback from the students, the introductory presentation seemed to work as intended, the students were not confused, and it was easy for them to understand their task at hand. They were divided into groups of four or five members. There werewas a total of 12 groups, six groups using the PHA and six groups using the misuse case and HAZOP approach.

During the process, we, the experiment leaders, observed and took note of the issues we had identified beforehand. This included time spent and feedback questions we asked the students. We had a few questions from the students, but there were only a handfull. This was less than expected, and something we need to consider in chapter 8.3.

One thing we wanted to prevent was that the groups that finished early would leave before the rest was finished. This might have pressured the rest to want to finish and leave as well, so we kept all the students in the auditorium until everybody was finished. The groups that finished early stayed at their places and chatted within the group until the end. See also 8.3.

Overall, the test progressed in a fine manner. At no point did we feel we had lost control of the situation, or missed out taking notes of what we had planned beforehand. This should make a good starting point for our discussion.

CHAPTER 7

Experiment - Results

This chapter presents the results from the experiment. We evaluate the results in chapter 8. Of the 53 persons that participated in the experiment, 49 provided material of sufficient quality to be used in the analysis, five PHA analyses and six misuse case and HAZOP analyses. See section 8.3 for the reason why the results from one group was dropped.

7.1 Result Overview

This section provides a short description of the results of the goals in section 6.1.1 and an analysis and presentation of the results.

- **See how the combined Misuse Cases and HAZOP results compares to those of PHA:** The most interesting result was that the misuse case and HAZOP groups had much higher identification rate for the sensors and control units. For the other components, there was no clear pattern for where one approach have better result than the other one.
- **See if there are any substantial difference in time to learn the methods, and if there are any recurring problems:** All of the PHA experimenting groups progressed from reading the handouts to evaluating the system before the misuse case and HAZOP teams started. The latter also needed more time to discuss within the team of how to aproach the task and the meaning of the guide words. On average, the misuse case and HAZOP teams spent five minutes longer than the PHA groups before starting on the experiment itself.
- **Compare the time spent to finish the analyses:** The PHA groups finished approximatily after 55 minutes, the Misuse Case and HAZOP groups finished approximatily after 1 hour and 15 minutes, a difference of 20 minutes.

7.2 Results Detailed

This section contains a in-depth description of the experiment results.

7.2.1 Summary

By analyzing the material, and observing the columns in 7.3, we set the following results.

- **A paired t-test show that the observed differences are statisticly significant.** The procedures does not produce random results.
- **For software components:** The misuse case and HAZOP approach has a larger identification rate.
- **For mechanical components:** Neither of the two approaches seem to have any significant advantage over the other.

7.2.2 Analysis of the Hazard Identification Rate

As mentioned earlier, the most important goal with this new approach is to improve the hazard identification, with focus on software hazards, early in the development process.

First we looked at all the groups and the hazards the experiment participants had identified, and divided them into results for each of the two approaches. For each of the two approaches we took note of how many hazards that were identified for each component in the system. Since the HAZOP analysis would provide many slightly different hazards because of the guide words, we only counted the hazards where the differences were significant. For example, if the water pump never starts, or starts too late, both the cause, consequence and solution would often be the same for them both. We therefore did not distinguish these as two different hazards.

The tables below show the number of hazards identified for each component by each of the two approaches. Table 7.1 show the results for the PHA groups, and table 7.2 show the results for the misuse case and HAZOP groups. The first column contains the ID which corresponds to the X-axis in figure 7.1 and 7.2. The second column is the component's name. The third column contains the number of hazards identified for this component by all groups combined. The fourth column shows the number of hazards for this component identified by each group on the average. This number is calculated by taking the total number of hazards identified (column three) divided by the number of groups, in the case of PHA, five, and six for misuse case and HAZOP.

ID	Component	Number of hazards	Number of hazards per group
1	Tank	5	1
2	Water pipe	2	0,4
3	Water pump	3	0,6
4	Non-return valve	3	0,6
5	Heating element	2	0,4
6	Steam valve	3	0,6
7	Steam pipe	2	0,4
8	Emergency release steam valve	4	0,8
9	Water level sensor	5	1
10	Water level control unit	3	0,6
11	Temperature sensor	1	0,2
12	Steam pressure sensor	4	0,8
13	Steam pressure control unit	2	0,4
14	Wiring	2	0,4
15	Electricity	3	0,6

Table 7.1: Results from the PHA groups.

ID	Component	Number of hazards	Number of hazards per group
1	Tank	2	0,33
2	Water pipe	4	0,67
3	Water pump	4	0,67
4	Non-return valve	6	1
5	Heating element	4	0,67
6	Steam valve	6	1
7	Steam pipe	2	0,33
8	Emergency release steam valve	3	0,5
9	Water level sensor	13	2,17
10	Water level control unit	14	2,33
11	Temperature sensor	11	1,83
12	Steam pressure sensor	9	1,5
13	Steam pressure control unit	12	2
14	Wiring	1	0,17
15	Electricity	1	0,17

Table 7.2: Results from the misuse case and HAZOP groups.

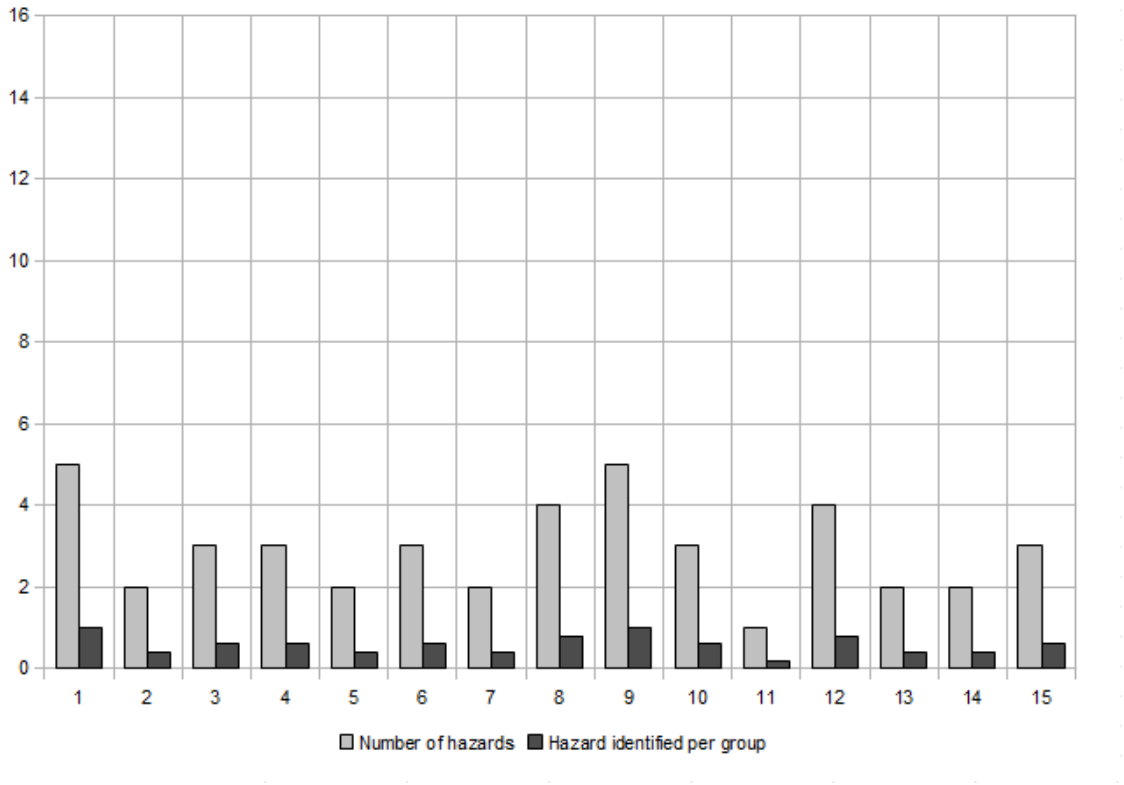


Figure 7.1: Results from the PHA groups. The light grey columns show the total number of hazards identified for the component, dark grey columns show the number of hazards identified for this component by each group.

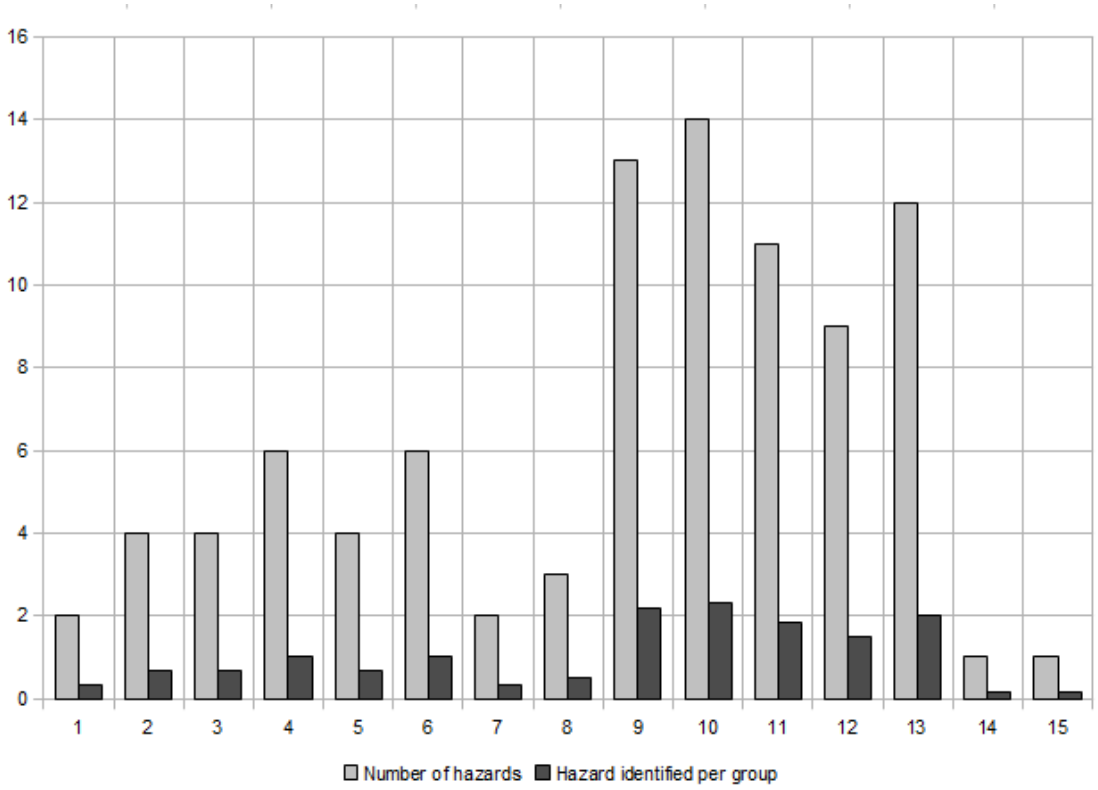


Figure 7.2: Results from the misuse case and HAZOP groups. The light grey columns show the total number of hazards identified for the component, dark grey columns show the number of hazards identified for this component by each group.

The first thing that draws the attention is the differences of the light grey columns for the sensors and control units. However, since the number of groups for each approach was different, looking at these alone would not give the correct impression. The dark gray columns give a better impression of the results. These columns are shown in table 7.3 and figure 7.3.

ID	Component	PHA	Misuse case + HAZOP
1	Tank	1	0,33
2	Water pipe	0,4	0,67
3	Water pump	0,6	0,67
4	Non-return valve	0,6	1
5	Heating element	0,44	0,67
6	Steam valve	0,6	1
7	Steam pipe	0,4	0,33
8	Emergency release steam valve	0,8	0,5
9	Water level sensor	1	2,17
10	Water level control unit	0,6	2,33
11	Temperature sensor	0,2	1,83
12	Steam pressure sensor	0,8	1,5
13	Steam pressure control unit	0,4	2
14	Wiring	0,4	0,17
15	Electricity	0,6	0,17

Table 7.3: Hazards identified per component per group for both the approaches.

The results for individual components are illustrated in appendix D.

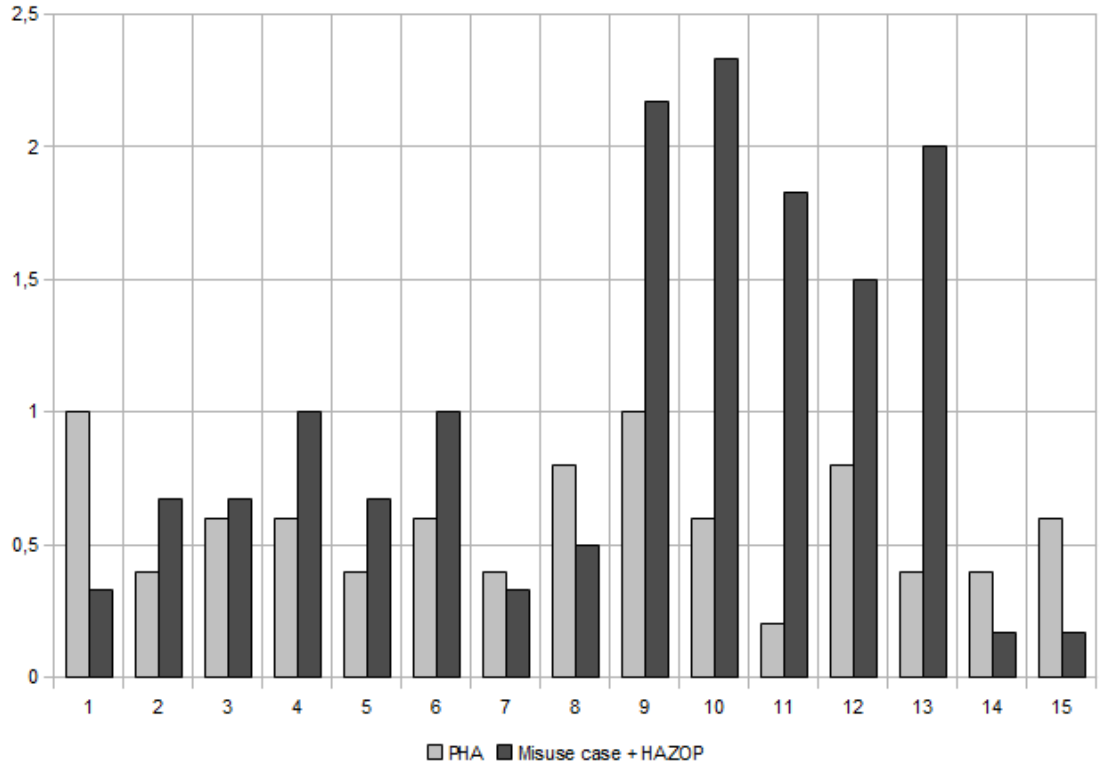


Figure 7.3: A comparrisson of the PHA results and the misuse case and HAZOP results. The number of total identified hazards by all groups divided by the number of groups. The X-axis is the components in the system, see table 7.1 or 7.2 for ID number.

7.2.3 Paired t-test

We have obtained two sets of quantitative data, where the sample sets are related. This opts for the use of a paired t-test to compare them. Each component has a pair of data, one for the identified hazards using PHA and one for the identified hazards using misuse cases and HAZOP. A more general test would be the sign test, which tests the difference in medians for paired samples, but it lacks the statistical power of the paired t-test. [1]

The null hypothesis, H_0, was that the difference between the two approaches was zero. The mean scores was calculated by adding the number of hazards for each component for each approach, and then dividing on the number of groups. These scores are the last column in table 7.1 and 7.2. We chose a significance level of 95%.

The results are shown in table 7.4.

[1] `http://en.wikipedia.org/wiki/Sign_test`

	PHA	Misuse Case + HAZOP	Difference
Mean	0,587	1,023	-0,436
St. Deviation (σ)	0,233	0,752	0,778
St. Error Mean	0,060	0,194	0,201

Table 7.4: Paired t-test.

The t-value is -2,17, and the p-value is 0,048. The difference mean was calculated to -0,436 and a 95% confidence interval from this difference was [-0,867, -0,005]. The results are statisticly significant, and we therefore reject the null hypothesis, and conclude that the test show that the two approaches produce different results. The difference in the mean values of the two groups is greater than would be expected by chance; there is a statistically significant difference between the input groups. Thus, on the average, the misuse case and HAZOP approach is better than PHA to identify hazards.

7.2.4 Other Observations

Figure 7.3 illustrates the number of hazards identified per group per component, with the results from each approach side by side. The results for individual components are illustrated in appendix D.

One of the assumptions made was that the misuse case and HAZOP approach would perform better on the following components: control units, sensors and components that the control units controls, with primarily focus on the control units. From figure 7.3 we see that the identification rate is higher for the control units and sensors, but for the pump and heating element, which are controlled by the control units, the identification rate is almost the same. For the pump, the identification rate was 0,6 for the PHA groups, and 0,67 for the misuse case and HAZOP groups. For the heating element, the identification rate was 0,4 for the PHA groups, and 0,67 for the misuse case and HAZOP groups. Even though it was for the control units we had expected the largest differences in identifications, we had also expected more differences in the water pump and heating element than what was the result. Looking at the data, there were no software hazards identified for these components for any of the groups but one. We wonder if the experiment participants had managed to consider the sensors as part of the software part of the system, but for some reason not the components on which the software executes. In other words, input components has been considered, but seemingly not output components. A closer inspection of the data reveals that the output has been considered in the control units themselves, meaning that the participants had identified some of the hazards we had expected, but they had identified them in another place. In any case, the most important aspect for us in this experiment was to check the identification rate on the control units and the interaction with connected components. The data show that this has been the case.

7.3 Disregarded Material

In addition to the many good analyses and results, there were a few indications that some participants had not understood the system concept completely. All of these indications involved connecting components that are not connected in the system drawing. One group had for example analyzed what would happen if the communication between the temperature contol unit and the steam pressure release valve had an error, although the steam pressure release valve is an independent component with purely mechanical functions. Since these kinds of error would never happen in the real system, the errors did not affect the analysis result and the errors were not attributed to any one of the two approaches, these results were disregarded.

CHAPTER 8

Experiment - Evaluation

This chapter describes our evaluation of the experiment. We discuss wether our experiment goals were fulfilled, whether the experiment results are useful and valid and what we need to act on based on this.

8.1 Summary

The results from the experiment were interesting, and provided valuable data for exploring the possibilities for the misuse case and HAZOP approach. The goals we had set with this experiment, and a thorough analysis of the validity of it shows that the result data are trustworthy. Based on these results we consider some modifications to the approach, and also prepare for the next phase of the thesis.

8.2 Goal Fulfillment

In this section we evaluate if the results from the experiment are satisfactory compared with our expectations. The emphisized comments are from section 7.1.

See how the combined Misuse Cases and HAZOP results compares to those of PHA

The most interesting result is that the misuse case and HAZOP groups had much higher identification rate for the sensors and control units. For the other components, there was no clear pattern for where one approach had better result than the other one.

The paired t-test, see section 7.2.3, showed that the results for the two methods were statistically significantly different. From figure 7.3 we can also see that the misuse case and HAZOP approach has some clearly better identification rates. A closer interpretation of this will be discussed in section 8.4, but it was encouraging to see that the results are in line with our hypothesis; the misuse case and HAZOP approach have a higher identification rate for software hazards than PHA.

See if there are any substantial difference in time to learn the methods, and if there are any recurring problems

All of the PHA experimenting groups progressed from reading the handouts to evaluating the system before the misuse case and HAZOP teams started. The latter also needed more time to discuss within the team of how to aproach the task and the meaning of the guide words. On average, the misuse case and HAZOP teams spent five minutes longer than the PHA groups before starting on the experiment itself.

Although PHA was clearly easier to learn and understand, we did not see a substantial time difference. Within the context of this experiment, a substantial time difference would be 15-20 minutes longer. Five minutes was well within acceptable limits. In a more proffesional environment, with participants that might demand total clarification before starting, this time could increase, but we doubt it still would be considered substantial in such a context. Therefore, we consider this goal fulfilled.

Compare the time spent to finish the analyses

The PHA groups finished approximatily after 55 minutes, the Misuse Case and HAZOP groups finished approximatily after 1 hour and 15 minutes, a difference of 20 minutes.

This time difference is acceptble. As mentioned, the misuse case and HAZOP approach is more complex and has more forms that needs to be completed. Therefore we expected our approach to be more time consuming than PHA, but still not too long. Considering the PHA is easier to learn, this time difference might even be shorter for experienced users. We consider the experiment fulfilled this goal.

8.3 Threats to validity

Ideally we would have performed this experiment on subjects that are familiar with safety analysis and that knows and works with such systems on a regular basis. Since that was not possible, and we had to use students instead, we thought it crucial to perform an extensive analysis of the threats to the validity of the experiment. We tried to identify anything that could make the experiment results invalid and evaluate wether the threat ocurred, did not ocurr or was avoided.

8.3.1 Inexperience

The first threat is the participants lack of experience in dealing with identification of hazards in a system. In order to minimize the effects of this threat, we used much time to make the experiment handouts as good as possible. They could not be too long so that it would cause confusion over details, but it needed to contain enough specifications for them to perform well. Several drafts were made, where we extended and shortened paragraphs, added and removed content, to a finished handout we were confident would give the experiment participants a good introduction to hazard analysis and the system concept, and that they would give us valid results.

One of the most discussed content of the handout were the hazard checklist example that we included. We feared that it would cause the students to only focus

on the list and that they would just go through each item on the checklist and stop after this. However, if we would not include it, would the students manage to come up with relevant hazards themselves? In the end we chose, in colabiration with our supervisor, to include it. The checklist just listed some hazard examples, and not the cause and consequence, and the students would have to evaluate the system and its potential hazards anyway. In addition, the students most likely needed examples to help them to get started. After analyzing these results, we conclude that even though some groups obviously used the checklist examples viguorosly, they had understood the purpose and techniques to a sufficient degree. We did not expect them to become experts in the short time they were given, but the quantitative and qualitative results they produced gave us confidence that they are valid, and that this threat was avoided.

8.3.2 Too Small Differences Between the Procedures

Should both approaches produce similar results, it would be difficult for us to argue one way or the other. It could mean that both procedures are equal in hazard identification, or, more likely, the students were not valid experiment subjects because of their lack of knowledge. As shown in chapter 7, and as mentioned in section 8.3.1, the experiment participants understood their job and gave us valid data for analysis. As shown by the paired t-test in section 7.2.3, the samples gave statistically significantly different results, so the two approaches clearly had their differences and this threat did not ocurr.

8.3.3 Motivation

The students agreed to participate in this experiment so that they would get money for their excursion trip abroad next year. They did not join just out of interest or to help us in our thesis work. This meant that their motivation for this experiment was, as a minimum, to show up and do an adequete job without really putting their mind to it (if they had just shown up and did nothing, they would not get paid). Our experiment, however, was but the first they would paricipate in. Should they get a bad rumour, other in need of experiment subjects would be reluctant to contact them. They were therefore motivated to do as good job as they could, and thereby ensuring that others would not have any second thoughts of asking them to participate in their experiements. At least, this was the case for the vast majority. There was one group of four students that diferred from the rest. All they handed in was a painting. Needless to say, this groups results were disregarded in our analysis. This action was also approved by our supervisor. Fortunately, the rest of the groups did a good job and gave us a large sample, so we conclude that this threat did not ocurr.

8.3.4 Questions During the Experiment

Considering that this was their first experience in hazard analysis, we had expected that we, the experiment leaders, would have to walk around assisting the students with the procedures and answering questions. This turned out to not be the case.

There were less than a dozen questions in total during the experiment. We wondered why, and decided to ask the students if they had any problems, instead of waiting for them to come to us. They answered that they did not need any help, and that they got what they needed from the handouts. This was good news to us, but we still wondered if they really had understood it or if they just did not care. As mentioned in section 8.3.3, their motivation seemed to be on top, and the results pointed in the same direction. Therefore, we conclude that this threat was sufficiently avoided.

8.3.5 Late Arrivals

Earlier experience has shown that late arrivals can cause disturbance enough to be an issue. Being late they would not have heard the introduction and the rest of their designated group would have to spend valuable time explaining it to them. A late arrival could also disrupt the process if the group had already started analyzing, thereby getting them off track. In our experiment there were four late arrivals. All arriving after the introduction, but before the rest of the students had finished reading the handouts. We did not see this causing any real harm to the experimenting.

8.3.6 Groups Finished Early Puts Pressure on Rest

One threat we identified early on was the fact that people finishing ealry and leaving would put pressure on the rest for leaving too. Earlier experience told us that this is a real threat, and would compromise the results. We therefore kept everyone at their seats before letting everybode leave at the same time. The groups that were finished first we let sit and chat within the group.

8.3.7 Conclusion

By some of the threats not ocurring, and the rest of them avoided, we feel that the experiment results are valid for our use and analysis. We can safely use the results to evaluate our procedure in comparisson to PHA. Again, we would have had a better basis had we used experts, but that was outside the financial borders of this thesis.

8.4 Discussion

The experiment gave us much valuable data on how our approach is compared to PHA, and what aspects that works well or not. One of the reasons we chose misuse cases as part of our approach was that it should provide an easy and fast understanding of the system at hand. We observed this to be true, at least to some degree. It was hard to say it definitively did, since no participants tried both approaches. However, based on our observation of their understanding and the hazards identified, it seemed that the misuse case and HAZOP groups had a better understanding of the system functions. We were unable to tell if it provided a faster understanding of the system, since this would be hard to measure and we did not want to disturb the process to much.

The misuse case and HAZOP procedure itself, as predicted, was not as easy to learn as PHA. Misuse case and HAZOP has several formalized steps that needed more thought to get the bearing of it. For an unexperienced student this was expected, but we are not so sure it would be so predominant with more experienced participants. Certainly, an expert in HAZOP would be much more efficient teaching the rest of the group the method rather than some text on some paper. Still, this would also be the case with PHA. In any case, the time to learn the methods are not the most important aspect. It is much more important that they are used properly once learned.

The hazard identification results were the most interesting and important part of this experiment. As shown in the previous chapter, the misuse case and HAZOP approach seem to have an avantage when it comes to identifying software hazards. Improving the software hazard identification was our main goal with our procedure from the start. Still, there is one issue with the hazards from the misuse case and HAZOP groups. The level of detail some of the hazards consisted of seemed to be too high considering that this procedure is meant to be performed in the concept phase. None the less, even if those few highly detailed hazards were removed, the misuse case and HAZOP approach would still provide better results.

For the non-software parts of the system, there were no obvious differences. For some components, one approach was better, for others, the other approach was better. The differences were either too small, or the total number of hazards identified was too small to decide whether this was real differences or just coincidence.

Based on the results and their validity, we conclude that the experiment was a success. However, as mentioned earlier, the approach might provide different results with a more experienced group, either better or worse, so we cannot conclude wether the approach has merit or not. For a final verdict, the approach should be used in several projects by experts in order to decide, but that is beyond the scope of this thesis.

8.5 Modifications to the Procedure

Based on the experiment results and feedback and questions from the participants during the experiment, we discussede several possible modifications.

The first possible modification was one that was highlighted in section 3.4.3, wether we should remove the textual misuse cases or not. We were worried that the *Threats* and *Mitigation* columns would not be used in the HAZOP, and would thus not serve any purpose, or worse, forget some threats that would not be handled in the HAZOP analysis. The experiment participants did not encounter this problem. They used the textual misuse cases vigorously, and some even said that it in some cases was important for identifying hazards that would not have been considered had they moved directly from misuse case diagrams to HAZOP. This alone was a sufficient argument for us to keep the textual misuse cases in place. However, should there later be conducted experiments with experts in the field, this issue should be revisited.

One of the challenges identified in section 3.4.4 was the lack of priorities in the procedure. Some hazards are much more important to handle than others, and considering the amount of tables produced in the HAZOP, some kind of priorities

should be introduced. We discussed two alternatives for this. The first one was to simply add a column, *Priority*, where one could define the importance of the hazard. Either High, Normal or Low, or a number from one to ten are most commonly used. The other alternative was to replace the column *Hazards* with a priority column. Instead of Yes/No, an item that is no hazard would get a dash, or simply left empty, in stead of having a priority. This would combine the two and still keep the function of both. We did, however, consider the need of a dedicated column for wether an item is a hazard or not for easy reference. We decided to bring this issue up during the interview in the next phase.

Part III

Discussion

CHAPTER 9

Interview - Preparation and Execution

9.1 Introduction

The main goal of the interview was to get an expert opinion on the misuse case and HAZOP approach; pros and cons, his thoughts on our concerns and a comparison of how the misuse case and HAZOP approach would perform compared to the commonly used PHA analyzing a system concept.

9.2 Planning

The stakeholders of the interview were Joshua Maringa, Thorbjørn Sæther of NTNU and Frank Reichenbach of ABB Norway. The main objective of the interview was to have an experienced hazard analysts help us explore aspects that we were uncertain about, and also determine the efficiency of misuse cases combined with the HAZOP approach compared to other methods of specifying safety requirements in the early phases of system development. The interview was prepared in a semi-structure form so that we would access more detailed information from the interviewee, only one person was to be interviewed and the theme of the questions to be covered was already set.

The questions were designed as open end since they would be influenced by the interviewee's response. This would allow for a good flow of the conversation in cases where the interviewee brings up issues that were not prepared for in the questions. This also gives the interviewee the opportunity to discuss in detail, issues that we raise and also have an opportunity to introduce issues that they think are relevant to the interview. Semi-structured interviews allows the interviewee to speak their minds and offer an in-depth investigation, especially interviews aimed at exploring personal accounts and feelings [33].

Below is the list of questions we had prepared for the interview and were used as guidance towards achieving the main objective of the interview.

- What are the methods used to evaluate software safety when capturing safety requirements that you consider effective in your organization? What are the most important pros and cons of the methods to you?
- In your experience, what is the major problem facing the most common hazard analysis methods used to evaluate and improve safety in software-controlled

safety-critical systems?

- How would you combine standards and regulations for safety-critical systems with the technically specific requirements to a project and its operating environments to ensure maximum safety in a software product?
- In the concept phase, do you use any other methods than PHA? Is PHA providing satisfactory results? Are there hazards often discovered in later phases that you wish were found in the concept phase? If so, if they had been discovered in the concept phase, would it have changed the design ?
- If the software functions as specified, does this give rise to any Hazardous Failure Conditions?
- When performing Hazard analysis in the concept phase, do you wish to identify as many hazards as possible, or just the most important ones? Do you wish to identify the most critical hazards?
- Do you have any experience with misuse cases? (Or use cases) Do you find them easy to use? Are the diagrams too messy? Have you used it in any project together with non-developers?
- Part of our procedure includes performing a HAZOP analysis. It is generally thought that it should be performed when the design is finished. Do you think it is too early to use it on a concept (for example, the boiler system)? If so, what more is needed before a HAZOP can be used?
- One of our main concerns is that this method produces results that might be too detailed to be of any practical use this early in the project. What are your thoughts on this? Do such things as "too detailed too early" exist?
- How would you consider combining misuse cases and HAZOP methods in eliciting safety requirements of a safety critical system?
- Do you think user friendliness would be a problem compared to PHA?
- Do you think that textual misuse cases are needed in addition graphical misuse cases, or are they just doing the same thing twice over?
- What is the relationship between hardware controls and software controls in the boiler system application that would improve the safety of the system?
- What hazard severity level should be allowed in the boiler system to ensure that it operates in a safe state and there is a low probability of hazards occurring and if so, their consequence can be tolerated.
- Are there plausible failure modes of the software or of the underlying computing hardware which are not contained by the software, which can give rise to Hazarders Failure Conditions?

9.3 Preparation

During the preparation of the interview, we gathered background information of our interviewee who is responsible for safety controls in the corporate center in ABB Oslo. He studied MSc in Information Technology and has a PhD in wireless sensor networks specializing in localization of sensors. This kind of information helped us in formulating the questions we wanted to raise to our interviewee and also helped us to establish credibility in our study area to him so that he was willing to open up and share information with us. It also helped us in assessing the accuracy of the information given to us.

9.4 Recording

Recording was to be done using of field notes and audio recording. We had note books for taking down points from the interview which would later help us in compiling the report. We had in mind that writing notes would not provide a complete record of everything said, hence it was necessary to have an audio recording. We got permission from the interviewee on conditions it was not be distributed without his authorization and would only be used for its intended purpose.

The audio recording was to help later during the transcribing process since it is much easier to search through and analyze the data once it is in written form. During the transcription, the interview is brought back to life and refreshes the main ideas discussed at the moment it was happening. It is important to add up your own information on the side, for example the atmosphere of the interview, thoughts of what a comment might have meant. This played a big role in guiding us through the writing of the findings and the results.

9.5 The Interview

We started the interview by introducing ourselves and giving a short description of our background and our studies. We introduced our case study and the area that it covered. Thereafter we explained the purpose of the interview and assured him of confidentiality of the information collected and its proper use. The interview itself started by discussing our research area: Specification of requirements for safety in the early development phases, which in our case is done using misuse cases and HAZOP analysis methods combined to increase the coverage of failures that might occur in information systems. During the initial stages of the interview, we started by describing how we came up with the idea of combining the two methods and the procedures we used to combine them.

The first part of the interview was characterized by a series of questions from the interviewee rather than what was expected, since he wanted to understand our project well and get an idea of all processes and functionality of the boiler system, for instance "What kind of analysis have you figured out on how to find all kinds of hazards in the software system. What are your likes and dislikes and what would you propose?" This gave us a perspective on how to restructure the interview and start dealing with the information we had at hand before going ahead with the questions

intended for the interview. However, the interview proceeded as planned and we were able to discuss several issues regarding the case study that was the basis of our research and we were able to achieve lots of valuable information.

The interview took 50 minutes and afterwards, we had some extra 10 minutes of having a general talk with the interviewee to thank him for availing himself to talk to us and share information that was of great help in our study field.

CHAPTER 10

Interview - Results

10.1 Review of the information gathered

During the interview, we were able to get answers to most of the questions that we had listed and discussed them in detail. There are some major key findings that he recommended us to consider when using the two hazard analysis methods:

- Identify the basic functions of the system in the concept phase and identify the errors that can occur during the operability of the system.
- If possible, implement both a top-down approach and a bottom-up approache since they give a wider coverage of the hazards and risks during the early development phase of sefety requirement specification. This is time consuming and some organizations might find it a waste of time, but it increases the scope of the coverage and reduces the amount of hazards in a software safety system.
- Consider all the operational modes of the boiler system and identify the threats that can occur thereafeter, find the mitigations to the hazards.
- Think of different hazards that can occur to a system and in what modes they can occur and also their causes. Thereafter, make a hierarchy of the findings and identify ways in which the hazards can be mitigated.
- Increase the coverage of the threats in software systems by using a structured approach where several operational modes of the system are covered. This shoulld be compared with our approach of implementing the both methods.
- Misuse cases might not be able to cover all the threats, hence they will work best when supported by other methods like in our case.
- All components have to be covered in the hazard analysis for us to produce a safety critical system. He emphasized the identification of where various components lay, either software supported or mechanical parts. HAZOP analysis should be performed on each component to find the threats they may pose to the system.
- Consistency should be emphasized when performing hazard analysis since the team members might change during the hazard analysis process or throughout

the development lifecycle of the software product, Hence it is necessary to maintain the same structure and procedures during the the entire phase.

- Hazards should be prioritized according to the level of threat and magnitude of their aftereffects.
- Detailed information for mitigating hazards during the concept phase should not be ignored but instead stored for later phases during the development of the safety critical system. This will make the design phase and development easier when the right time for their use comes.
- There should be a list of components where each component corresponds to the hazop tables that have analyzed this component. This will make it easier for the reader to reffer to a certain components and have information of which table(s) to check.

The interviewee encouraged us to think in terms of safety functions when looking at the hazards and risks to the system. The system can be viewed in terms of sub-components and as a whole system view thereby we will be able to create the necessary safety functions. The safety functions should be the core of the analysis - for example: check water level function which should be viewed in terms of the components and sub-components that substitute the function. The inputs and outputs parameters and the execution process should be analyzed to find the risks and hazards involved with them before reviewing them as a one whole component. The safety functions should be connected to the SIL levels where the main objective is to achieve software safety and the greater the importance to safety of the system whose SIL is under consideration, the lower the rate of unsafe failures should be. IEC 61568 [1] recommends that hazards that are posed by Equipment Under Control (EUC) and its control system should be identified and analyzed and that a risk assessment should be carried out. Each risk is then tested against tolerability criteria to determine whether it should be reduced. If risks are reduced by redesign of the EUC we are back to the starting point and hazard identification and analysis and risk assessment should again be carried out. The hazards that contribute to the risks posed by the EUC must be mitigated until their risk is considered tolerable.

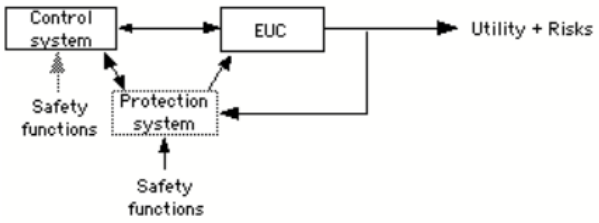


Figure 10.1: IEC 61508 system model [1]

CHAPTER 11

Interview - Evaluation

Throughout the interview session, we were able to gather much information regarding software safety issues and requirement specification during the early development phases. This was based on an intensive discussion with the interviewee on the key issues that influence the specification of safety requirements. The key issues discussed will help us to come up with a better way to integrate the misuse cases and HAZOP methods with the aim of detecting possible failure events in a safety critical system. Some of the findings below are relevant to our case study.

- Identifying the basic operation functions of a safety system during the concept phase will enable us to capture as many threats as possible since we are able to look at the system in terms of function for each module. By using misuse cases, we are able to perform hazard analysis on each function.

- Both top-down approach and bottom-up approach are effective ways of detecting threats in a system and the interviewee advocated the use of both since it will maximize the rate of capturing threats. In our case study, we feel in that it would be of great importance to have both approaches implemented but the limitation lies with the methods we are using hence we will consider them in the further work section.

- By considering all the operational model of the boiler system, we are able to identify the hazards and ways to mitigate them through a systematic procedure. The interviewee advised us to focus on both hardware and software faults to increase the coverage of the hazards and thus decreasing the probability of the system failure in operation. A hierarchy of the findings should be noted down and applied in HAZOP to find their mitigations.

- Implementing a structured approach will aid in increasing the coverage of the possible threats to the system, hence achieving the desired characteristics of a safety critical system.

- We agreed with the interviewee regarding the use of misuse cases in conjunction with HAZOP for the purpose of increased coverage of threats to the system hence enabling the development of a software safe system.

- All components in the system should be listed to be sure that they will are considered during hazard analysis. For instance, the pressure valve should

be analyzed to determine what threats it faces and whether it will be controlled by software or mechanically. In the analysis, we should have a clear distinction between the system components and the activities performed by the components.

- Iteration should be emphasizes in the hazard analysis process since it contributes to the broader coverage of system. During the concept phase, not all information regarding the system is know and new threats are detected throughout the analysis period.

- Prioritization of hazards is essential in knowing the impact that the hazards can cause to the system and their level of tolerance can be determined. Developing of the systems can also be influenced by the SIL level requirements hence the analysis team should list all the hazards and arrange then according to their level of severity.

- It saves much of the system development time when the hazard analysis team gather more information since the information is useful during other phases such as design and development of the system.

In general, it is wise to increase the coverage of threats to a safety critical system since it lowers the risk involved and also the probability of failure events occurring.

11.1 Threats to validity

There were few if any misunderstandings in the interview that we conducted with Frank Reichenbach from ABB since we had similar level of understanding of the safety issues in developing software systems. However, there was a slight delay in the beginning of the interview since we spent some time explaining our research and the interviewee had another perspective of what we were doing than we expecting but we managed to get on track. Different language skills could lead to misunderstandings or misinterpretations in an interview where either of the parties was not fluent or did not have a better understanding of the language in use but in our case we did not have any communication problems.

We used open end questions to allow a good flow of conversation with the interviewee to allow him to bring up issues that were not included the interview plan but were still important. This is a characteristic of semi-structured interview but it could in the discussion.

The environment for conducting the interview was quiet and allowed for a comfortable interaction with enough space for all the materials needed to conduct the interview. The room was equipped with a projector screen and a whiteboard that we could use for making illustrations and displaying images. With this kind of setting, we eliminated any chances of not having proper response from the interviewee and the environment encouraged him to share more of his views.

Using a recorder could make him uncomfortable if we had not requested for permission from the interviewee but he allowed us to record the interview for the purpose of transcribing after the interview and use the details for writing the report.

However, he did not want the recording to be hosted on public social sites or used for non-academic purposes.

CHAPTER 12

Summary and Final Verdict

After evaluating the results from both the experiment and the interview, we can summarize our evaluation of the procedure and the changes we have made, and make a final verdict.

12.1 Summary of Validation

For veryfying the misuse case and HAZOP procedure's usefulness, we have conducted an experiment with students. A PHA was used for comparisson. Both methods were used on a concept system. In addition, we interviewed an expert in the field. This gave us valuable feedback that allowed us both to both confirme many of our assumtions and also highlight areas that needed modification. Even though there were threats to the validity of the student experiment, we came to the conclusion that the experiment and interview gave us valid data.

12.2 Summary of Changes

After conducting the experiment and the interview, we came up with some changes that would improve the usefulness and effectiveness of the two methods when identifying hazards. The first proposal was to add a hazard priority column to the right of the HAZOP tables. The priority column would have either numerical values of 1 - 10 or high, medium or low priority levels. It would help in identifying the most critical hazards in the system and alert the developers to give them more consideration. We also came up with the issue of having a list of components of the system which is matched with the specific HAZOP tables where they have been analyzed, thus creating a quick reference point to all the system components amidst the many HAZOP tables. This can also be written in the *Related Tables* item now introduced, but a digital spreadsheet is advisable. The final draft for the HAZOP table is shown below in table 12.1. Note that we ended up not changing the misuse case, but highlighted that should this approach be the target of further experimenting, the *Threats* and *Mitigation* columns should be revisited. Read more in section 8.5.

Table 12.1: Updated HAZOP table.

ID:	
Related Tables:	
Component(s):	
System Function:	

Guideword	**Consequences**	**Cause**	**Hazard**	**Possible Solutions**	**Priority**
TOO LATE					
NEVER					
UNEXPECTED					
SPORADIC					
TOO OFTEN					
INCOMPLETE					
INCORRECT					
UNCHANGING					

12.3 Summary of Evaluation

After a thoughrough analysis and case study in the Specialization Project [2], and in this thesis, an experiment with students and an interview with an expert, we have had a lot of valuable input for evaluating the advantages and disadvantages for the misuse case and HAZOP approach.

The result of the case study in the Specialization Project [2] confirmed that this procedure, although it had its flaws, should be explored further. The results from the student experiment gave us a comparrisson with PHA, and also highlighted some of the issues we knew existed.

The students, although inexperienced, seemed to grasp the subject quickly and gave us good data. The identification of software hazards in the experiment confirmed our hypothesis about the procedure, and further strengthens it. The t-test shows that the data are statistically significant.

The interview gave us the opportunity to discuss our procedure with an expert in the field, and we got a better understanding of how hazard analysis is performed in real projects. We also got important feedback to our questions, and our confidence in the final version of the procedure is strong.

12.4 Final Verdict

Based on the student experiment reuslts, the higher software hazard identification, the t-test that show the statistical significanse of the results, and the feedback from the interviewee, we have come to the conclusion that the misuse case and HAZOP procedure has merit, and that it should be tested in real projects with experienced analysists to see if this is true or not.

CHAPTER 13

Walkthrough

This chapter contains a walkthrough of the final version of the misuse case and HAZOP procedure. Much of it is the same as in section 3.5, but updated with the changes made after the experiment and interview. It is assumed that the reader has a basic knowledge of use cases and hazard analysis.

13.1 Misuse Case Diagram

Consider a car, in which the basic function is to get the driver from point A to point B safely. Figure 13.1 shows how you can loose control over the car by skidding because of bad weather. The misuse case is therefore "Bad weather", and the threat is "Make car skid". How can you help the driver maintain control over the car in bad weather conditions?

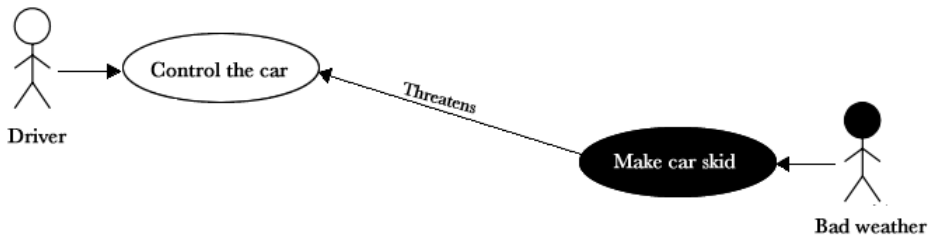


Figure 13.1: Loosing control

The analysis team will have to consider options to how to keep the car from skidding, or lowering the effect of it somehow. Figure 13.2 shows how two new functions are introduced to help the driver; traction control and an anti-lock braking system. We say these functions *mitigate* the threat. It is up to the team to consider when the misuse cases has been sufficiently mitigated.

Threats can then be introduced to the new mitigation use cases as well. For example, a threat could be "ABS breaks too softly". A new use case could be introduced to address this issue, and so fourth. You could go on endlessly, but this would cause the system to be overly expensive and complex. It is up to the analysists to evaluate when they have made the system safe enough, and that further expanding the system safety is either too costly or non-essential.

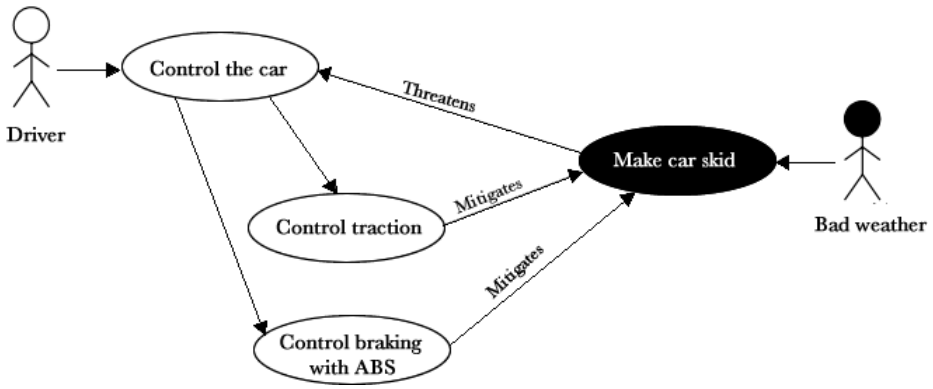


Figure 13.2: Threat mitigated

13.2 Textual Misuse Case

Following the misuse case diagrams above, the textual misuse cases, see figure 13.1, expands the the system overview to include lower level behaviours. This is were the analysis team take a more detailed look at how the system operates and functions. All the use cases from the diagrams are listed in the first column, and the team agrees how this use case is realized in the system, and writes it in the "System Response" column. As seen in table 13.1, these can be of different levels. "Prevent car from crashing" describes a high level function, and "Prevent wheels from locking while braking" is a more detailed function involving named components. The team also identifies threats, and possible mitigations to these threats, to the system responses.

Table 13.1: Textual misuse case: Car Control

Textual misuse case: Car Control			
Use case	**System Response**	**Threats**	**Migitation**
Control car	Prevent car from crashing	Skidding	Traction control ABS
Control traction	Prevent excessive throttle being applied on bad condition surface	System prevents any throttle applied to the wheels Software error Car component failure	Power the brake lights Backup functionalities Exception handling Component redundancy
ABS braking	Prevent wheels from locking while braking	Speed sensor failure Hydraulic failure Software error	Notify driver and disable system Backup functionalities Component redundancy

The threats column should only include situations where there actually could be a risk of danger involved. If you have a chainsaw, "Prevents chainsaw from starting", although an undesired effect, is not dangerous.

13.3 Using HAZOP on the Misuse Case Results

Moving on to HAZOP, you apply all the guide words to the items in the "System Response" column. There are two sets of guide words, one for software hazards and one for mechanical hazards. The software version is used whereever software is controlling the process. Preventing the wheels from locking is controlled by software, as it monitors the wheels speed and increases and decreases the brake pressure accordingly. Manual braking, for example, has no involvement with software. Braking is only resulting from the driver pushing the brake pedal.

The following is an example of a HAZOP table on *Prevent wheels from locking while braking*. Not all guide words may be applicable everywhere. If a guide word does not make sense for that particular item, skip it.

Table 13.2: HAZOP: Prevent wheels from locking while braking

ID:	1
Related Tables:	
Component(s):	ABS
System Function:	Prevent wheels from locking while braking

Guideword	**Consequences**	**Cause**	**Hazard**	**Possible solutions**	**Priority**
TOO LATE	ABS reduces pressure on the brakes too late, wheels lock	Sensor failure Hydrolic failure	Yes	Add redundancy	High
NEVER	ABS fails entirely, wheels lock when braking	Software error Sensor failure Hydrolic failure	Yes	Sanity check Add redundancy	High
UNEXPECTED	ABS releases brakes suddenly and when not needed	Software error	Yes	Sanity check	High
SPORADIC	ABS releases and applies pressure on the brakes sporadicly and unintentionally	Software error	Yes	Self-test Sanity check	High
TOO OFTEN	ABS activates at too high wheel speed	Sensor failure	Yes	Self-test Sanity check	Medium
INCOMPLETE	Speed value message incomprehensible	Software failure	Yes	Self-test Sanity check	High
INCORRECT	Wheel speed registered in ABS not same as real wheel speed	Sensor failure Software failure	Yes	Sanity check Add redundancy	High
UNCHANGING	Wheel speed value does not change	Sensor failure Software failure	Yes	Self-test Alert driver for repair	High

Based on the results from the HAZOP analysis, the analysts need to evaluate what they need to translate into safety requirements. Some of the hazards identified have too low propability to ocurr, or are too expensive to fix compared to the gain.

From table 13.2 one could argue that adding redundancy, an extra set of hydraulic components, would not be cost-effective, but a robust sanity check in the software would be highly recommended.

Part IV

Conclusion

CHAPTER 14

Conclusion and Further Work

This chapter concludes our study and also present a description of further work.

14.1 Conclusion

During the start of the master's thesis, our main objective was to test the misuse case and HAZOP methods combined to achieve high hazard identification rate as compared to the rate achieved using the Preliminary Hazard Analysis (PHA) method. The misuse case and HAZOP, when combined, increase the rate of software hazard identification and widen the coverage of components in the system as compared to the PHA method. Although it takes longer to use the procedure, its benefits are better than those of PHA and it improves efficiency in software hazard identification. The interviewee was impressed by the idea of the procedure and gave us some guidelines regarding hazard prioritization, using a structured approach in the procedure considering all the mechanical and software guided components in the system. The paired t-test performed shows that the experiment results are statistically significant and the procedure does not produce random results, thus making the data collected valid. The procedure ensures that most of the hazards are identifies, ranked in their hazard priority level and the mitigation of the hazards is proposed. We are therefore satisfied with the outcome of the procedure and would like to further test it on a real project and confirm its practicability.

In short, we belive the reason this procedure is better at software hazard identification is because of the more detailed focus the guide words in HAZOP provide, and the misuse cases highlight the system properties so that HAZOP is easier to perform this early in the project. In a PHA you just look at the concept diagram, and perhaps some early requirements, where as in our procedure you get a more systematic approach that improve the software hazard identification.

14.2 Further Work

The most important thing to do as a continuation is to test the procedure on experienced analysts, preferably in real projects with a multi-disciplinary team. Their assessment and evaluation will be the final judgement of whether this procedure has merit or not. There are a number of criterias that will have to be considered:

- **Number of hazards identified, and their relevance and importance.**

- **Amount of documents and data produced, and their usefulness.**
- **Amount of time spent learning the procedure.**
- **Amount of time spent analyzing in total.**
- **Evaluate the need of textual misuse cases, especially the *Threats* and *Mitigation* columns**.

The analysts will have to compare this to their earlier experience, and weighing the pros and cons, decide wether this procedure has merit or not. Alternatively, they could do as we did, have different groups working on the same concept with either the misuse case and HAZOP approach or PHA, or any other approach used in the field, and compare the results. The analysists should also, however, take their own experience into consideration when evaluating the results.

Part V

References and Appendix

References

[1] IEC61508, "Iec 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems," 2008.

[2] J. Maringa and T. Sæther, "Cesar - specification of requirements for safety in the early development phases," Master's thesis, Norwegian University of Science and Technology, 2010.

[3] C. A. Ericson, *Hazard Analysis Techniques for System Safety.* John Wiley and Sons, 2005.

[4] S. Oleson and M. Johnson, "System methodology for analysis, review and test (smart) for system software safety analysis (sssa)," *CSC Papers*, 2008.

[5] E. Tran, "Verification/validation/certification," *Dependable Embedded Systems*, pp. 18–849b, 1999.

[6] J. C. Laprie, "Dependability: Basic concepts and terminology," *Springer-Verlag, Wein, New York.*, 1992.

[7] N. Leveson, "Safety and hazard analysis." Talk on Software Safety and Reliability.

[8] *IEEE Std. 610.12-1990, Standard Glossary of Software Engineering Terminology.*

[9] J. McDermid, "Software hazard and safety analysis," *FTRTFT '02 Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems: Co-sponsored by IFIP WG 2.2*, 2002.

[10] J. Wang, "Analysis of safety-critical software elements in offshore safety studies," *Disaster Prevention and Management*, pp. 271 – 282, 2000.

[11] U. A. C. DIRECTORATE FOR SAFETY, "Software system safety,"

[12] T. Hardy, "Hazard analysis types and tools," 2010.

[13] M. Rausand, "Preliminary hazard analysis," 2005.

[14] R. Mohr, "Preliminary hazard analysis (lecture presentation)," *Sverdup Technology, Inc.*, vol. Fourth Edition, June 1993.

[15] K. Crow, "Failure modes and effects analysis (fmea)," 2002. Accessed on 25th January, 2011.

[16] J. B. Dugan, "Fault tree analysis of computer-based systems," *Reliability and Maintainability Symposium*, pp. 1 – 83.

[17] N. R. D. H. W.E. Vesely, F.F. Goldberg, "Fault tree handbook," *Division of Systems and Reliability Research Office of Nuclear Regulatory Research U.S. Nuclear Regulatory Commission Washington, DC 20555-0001*, 1981.

[18] N. Leveson, "Safeware: System safety and computers," 1995.

[19] N. G. Leveson, "Intent specifications: An approach to building human-centered specifications," vol. VOL.26, pp. 15 – 34, 2000.

[20] N. R. Mead, "Requirements ellicitation introduction," *Software engineering institute*, 2006.

[21] J. E. Rumbaugh, "Getting started: Using use cases to capture requirements," 1994.

[22] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse case," 2004.

[23] I. Alexander, "Misuse case : use cases with hostile intent," 2002.

[24] H. L. W. Klaus Marius and T. Maier, "Hazop analysis of uml-based software architecture descriptions of safety-critical systems," 2004.

[25] "Iec 61882: Hazard and operability studies (hazop studies) - application guide," 2001.

[26] M. Rausand, "Hazop - hazard and operability study," 2005.

[27] P. Q. R. I. PQRI, "Training guide: Hazard and operability analysis (hazop)." pdf.

[28] ACUSAFE, "The hazop (hazard and operability) method," 2009.

[29] *OMG. Unified Modeling Language: Superstructure, version 2.0*, 2004.

[30] N. M. Raimundas Matuleviciu and P. Heymans, "Alignment of misuse cases with security risk management,"

[31] J. J. Pauli and D. Xu, "Trade-off analysis of misuse casebased secure software architectures: A case study," *In Proceedings of the 3rd International Workshop on Modeling, Simulation, Verification and Validation of Enterprise Information Systems. MSVVEIS05*, pp. 89–95, 2005.

[32] M. Crowther, *Test Strategy Template*. Cyreath, 2008.

[33] B. J. Oates, *Researching Information systems and computing*. Sage publishers, 2006.

[34] I. Alexander, "Misuse cases: Use cases with hostile intent," *IEEE Software*, pp. 58 – 63, 203.

[35] C. M. Holloway and K. J. Hayhurs, "Proceedings of the 21st international system safety conference - 2003," in *Software System Safety and The NASA Aeronautics Blueprint*, 2003.

[36] U. A. U. A. F. Joint Services Computer Resources Management Group, U.S. Navy, *SOFTWARE SYSTEM SAFETY HANDBOOK, A Technical and Managerial Team Approach*. Joint Software System Safety Committee, Joint Services System Safety Panel, Electronic Industries Association, G-48 Committee.

[37] J.-m. B. A. U. M. V. Krzysztof Czarnecki et al, Ileana Ober, "11th international conference, models 2008 toulouse, france, september/october 2008 proceedings," in *Model Driven Engineering languages and systems*, 2008.

[38] J. Payne, "Software safety by the numbers," *EE Times design*, 2004.

[39] J. N. Ruck, "Applying misuse cases to improve the security of information systems," *Technical report*, vol. 1, pp. 43 – 54, 2009.

[40] D. o. M. e. University of UTAH, "Preliminary hazard analysis (pha) packet," in *Lecture Outline*.

[41] V. N. Vincoli, Jeffrey W and Reinhold, "Basic guide to system safety," p. 68, 1993.

Appendices

APPENDIX A

Experiment - Misuse Case and HAZOP

Thank you for participating in our experiment. The test will last approximatily 90 minutes. Feel free to ask any questions should there be any uncertainties.

A.1 Task

Your task is to use the Misuse Case and HAZOP (Hazard and Operational study) methods to identify the hazards for a steam boiler concept. An introduction to the methods with some examples are included below. Please read it through and familiarize yourself with the methods before moving on to the test itself.

A.1.1 Misuse Cases

Use cases generally describe behavior that the system owner wants the system to show. Use case diagrams are used for eliciting system requirements, preferably used for functional requirements, but they also offer some support for non-functional requirements such as safety and security threats to the system.

Misuse cases apply the concept of negative scenario for a situation that the system's owner does not want to occur in a use case context. Misuse cases are most often used in relation to security issues, but has also been applied to safety issues with good results. It enables the designers to focus on the safety issues from the initial stages of system design and maximize the capturing of threats that can occur to the system. They provide a good communication channel between the developers and all stakeholders in a project. The development of misuse cases will often lead to creation of new use cases to handle the threats posed by the misuser.

A.1.2 HAZOP

A HAZOP study is a group technique in a structured and systematic examination of a process or operation in order to identify and evaluate problems that may represent risks to personnel or equipment, or prevent efficient operation. The group starts by defining a set of study nodes, in our case all the "System Response" items from the misuse case analysis. For each of these study nodes, apply the guide words. For each guide word, identify likely *Consequences* and *Causes*, wether this item is considered a hazard or not, and possible solutions to mitigate the consequences.

A.2 Misuse Case Example

Consider a car, in which the basic function is to get the driver from point A to point B safely. Figure A.1 shows how you can loose control over the car by skidding because of bad weather. The misuse case is therefore "Bad weather", and the threat is "Make car skid". How can you help the driver maintain control over the car in bad weather conditions?

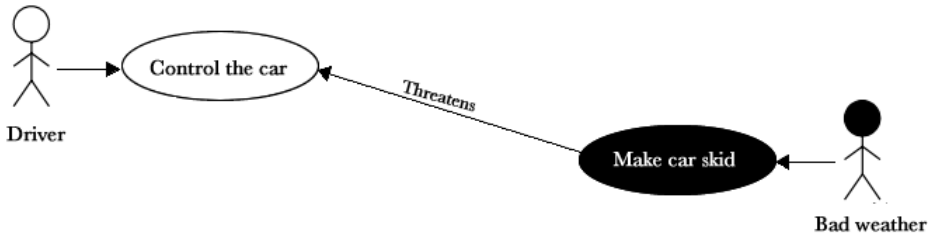


Figure A.1: Loosing control.

Figure A.2 shows how two new functions are introduced to help the driver; traction control and an anti-lock braking system. We say these functions *mitigate* the threat.

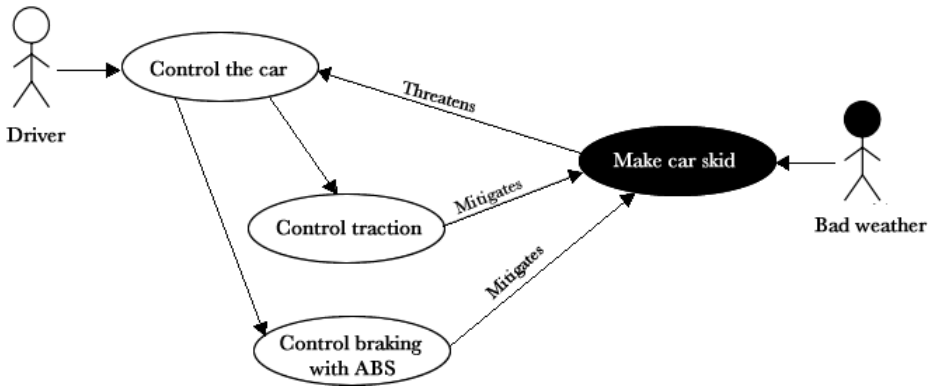


Figure A.2: Threat mitigated.

Threats can be introduced to the new mitigation use cases as well. For example, a threat could be "ABS breaks too softly". A new use case could be introduced to adress this issue, and so fourth. You could go on endlessly, but this would cause the system to be overly expensive and complicated. It is up to the analysists to evaluate when they have made the system safe enough, and that further expanding the system safety is either too costly or nonessential.

A.2.1 Textual Misuse Case

Following the misuse case diagrams above, the textual misuse cases expands the the system overview to include lower level behaviours.

Table A.1: Textual misuse case: Car Control

Car Control			
Use case	**System Response**	**Threats**	**Migitation**
Control car	Prevent car from crashing	Skidding	Traction control ABS
Control traction	Prevent excessive throttle being applied on bad condition surface	System prevents any throttle applied to the wheels Software error Car component failure	Power the brake lights Backup functionalities Exception handling Component redundancy
ABS braking	Prevent wheels from locking while braking	Speed sensor failure Hydraulic failure Software error	Notify driver and disable system Backup functionalities Component redundancy

Note: The threats column should only include situations where there could be a risk of danger involved. If you have a chainsaw, "Prevents chainsaw from starting", although an undesired effect, is not dangerous.

A.3 Example of HAZOP

Moving on to HAZOP, you apply all the guide words to the system responses. There are two sets of guide words, one for software and one for mechanical hazards. The software version is used whereever software is controlling the process. Preventing the wheels from locking is controlled by software, as it monitors the wheels speed and increases and decreases the brake pressure accordingly. Manual braking, however, has no involvement of software. Braking is purely a result from the driver pushing the brake pedal.

Following are two examples of how a HAZOP table on *Prevent wheels from locking while braking* (software) and *Manual braking* (mechanical) could look like. Note that not all guide words are applicable for everything. If a guide word does not make sense for that particular item, skip it.

Table A.2: HAZOP: Prevent wheels from locking while braking

Prevent wheels from locking while braking				
Guideword	**Consequences**	**Cause**	**Hazard**	**Possible solutions**
TOO LATE	ABS reduces pressure on the brakes too late, wheels lock	Sensor failure Hydrolic failure	Yes	Add redundancy
NEVER	ABS fails entirely, wheels lock when braking	Software error Sensor failure Hydrolic failure	Yes	Sanity check Add redundancy
UNEXPECTED	ABS releases brakes suddenly and when not needed	Software error	Yes	Sanity check
SPORADIC	ABS releases and applies pressure on the brakes sporadicly and unintentionally	Software error	Yes	Self-test Sanity check
TOO OFTEN	ABS activates at too high wheel speed	Sensor failure	Yes	Self-test Sanity check
INCOMPLETE	Speed value message incomprehensible	Software failure	Yes	Self-test Sanity check
INCORRECT	Wheel speed registered in ABS not same as real wheel speed	Sensor failure Software failure	Yes	Sanity check Add redundancy
UNCHANGING	Wheel speed value does not change	Sensor failure Software failure	Yes	Self-test Alert driver for repair

Table A.3: HAZOP: Manual braking

Manual braking				
Guideword	**Consequences**	**Cause**	**Hazard**	**Possible solutions**
NO OR NOT	Brakes do not respond when pushing pedal	Hydraulic failure	Yes	Alert driver
MORE	Brakes respond stronger than intended when pushing pedal	Hydraulic failure	Yes	Alert driver
LESS	Brakes respond less than intended when pushing pedal	Hydraulic failure	Yes	Alert driver
AS WELL AS	-	-	-	-
PART OF	-	-	-	-
REVERSE	-	-	-	-
OTHER THAN	-	-	-	-
EARLY	-	-	-	-
LATE	Delay from push of pedal till brakes initiated	Hydraulic failure	Yes	Alert driver
BEFORE	-	-	-	-
AFTER	-	-	-	-

A.3.1 Example of Hazards Checklist

These are examples of hazards that might appear on several systems during their operations, they will guide you in finding the possible failure of the steam boiler. You do not need to evaluate every item in your analysis, use the ones you find relevant.

Common Causes
___Utility Outages
___ Moisture/Humidity
___ Temperature Extremes
___ Seismic Disturbance/Impact
___ Vibration
___ Flooding
___ Dust/Dirt
___ Faulty Calibration
___ Fire
___ Single-Operator Coupling
___ Location
___ Radiation
___ Wear-Out
___ Maintenance Error
___ Vermin/Varmints/Mud Daubers

Software Errors
__ Overflow {Too much work for the CPU}
__Wrong values
__ Wrong timing of message send/receive
__ Loss of data
__ Infinite loops
__ Errors in handling or interpreting data
__ Calculation errors

Control Systems
___ Power Outage
___ Interferences (EMI/ESI)
___ Moisture
___ Sneak Circuit
___ Sneak Software
___ Lightning Strike
___ Grounding Failure
___ Inadvertent Activation

Leaks/Spills (Material Conditions)
___ Liquids/Cryogens
___ Gases/Vapors
___ Dusts - Irritating
___ Radiation Sources
___ Flammable
___ Toxic
___ Reactive

Explosives (Initiators)
___ Heat
___ Friction
___ Impact/Shock
___ Vibration
___ Electrostatic Discharge
___ Chemical Contamination
___ Lightning
___ Welding (Stray Current/Sparks)

Figure A.3: Hazards Checklists [14].

A.4 Instructions

To test the methods, we have included a concept from a steam boiler. There are numerous things that can go wrong if this system does not behave as intended. With the help of the analysis methods, you are to identify these possible hazards. You are to do the following:

1. Familiarize yourself with the system. Try to understand the purposes of the functions and the flow of information.
2. Perform a misuse case analysis on the systems use cases.
3. Translate the misuse case diagrams into txtual misuse cases.
4. Perform a HAZOP analysis based on the results from the textual misuse cases.

A.5 System Description

The system includes two control units and several mechanical parts. Water is fed to a tank through a non-return valve by a pump, a heating element heats the water inside the tank and the steam generated is delivered to the industrial process through a valve. A safety release valve is included in case of too high steam pressure. Two control units, using several sensor, controls the process in order to have the right amount of water, heat and steam pressure in the tank.

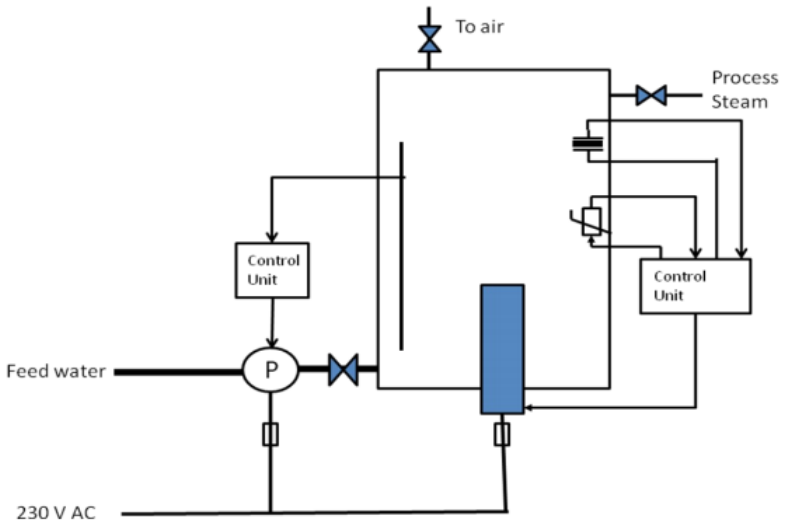


Figure A.4: Steam boiler concept.

A.5.1 System Detailed

This section describes the concept system in detail. See figure A.4 for an illustration of how the components are connected.

Components

This section lists the components and their purpose in the system.

- Tank: A water tank designed to withstand high pressure.
- Water pipe: Leads the water from storage to the tank.
- Water pump: Regulates the water flow to the tank.
- Non-return valve: Prevent reverse water flow.
- Heating element: Provides heat to transform water into steam.
- Steam valve: Releases steam from tank to the steam pipe.
- Steam pipe: Delivers steam to the industrial process.
- Emergency release steam valve: In case of too high steam pressure, this valve opens.
- Water level sensor: Measures the water level and sends the value to the control unit.
- Water level control unit: Controls the water pump.
- Temperature sensor: Measures the temperature in the tank.
- Steam pressure sensor: Measures the steam pressure in the tank.
- Steam pressure control unit: Controls the heating element.
- Wiring: Wires connecting the components to each other and to electric power.
- Electricity: Power is provided from a 230 V AC outlet.

Functional Requirements

The steam boiler has several functional requirements that ensure that the system performs all the operations required to deliver steam to the industrial process and all the components function as expected. They are listed as follows:

- The steam boiler shall deliver steam at a predefined, constant pressure to an industrial process.
- Steam is produced by heating water using an electric heating element.
- The steam pressure is controlled by regulating the temperature setting on the heating element thermostat.
- The water level in the tank is controlled by a feeding pump which pumps water into the tank via a non-return valve.
- The safety of the steam boiler is taken care of by a safety valve that opens to air. The release pressure for the safety valve is fixed, based on the strength of the boiler.

A.6 Analysis

This section provides the use cases based on the functional requirements of the boiler. You can use this as a basis when you start your analysis.

Table A.4: Textual misuse case

Use case	**System Response**	**Threats**	**Migitation**

Table A.5: HAZOP

Guideword	**Consequences**	**Cause**	**Hazard**	**Possible solutions**
TOO LATE				
NEVER				
UNEXPECTED				
SPORADIC				
TOO OFTEN				
INCOMPLETE				
INCORRECT				
UNCHANGING				

Table A.6: HAZOP

Guideword	**Consequences**	**Cause**	**Hazard**	**Possible solutions**
NO OR NOT				
MORE				
LESS				
AS WELL AS				
PART OF				
REVERSE				
OTHER THAN				
EARLY				
LATE				
BEFORE				
AFTER				

APPENDIX B

Experiment - Preliminary Hazard Analysis

Thank you for participating in our experiment. The test will last approximatily 90 minutes. Feel free to ask any questions should there be any uncertainties.

B.1 Task

Your task is to use the Preliminary Hazard Anaylsis (PHA) method to identify the hazards for a steam boiler used in industrial processes. An introduction to the method with some examples is included below. Please read it and familiarize yourself with the method before moving on to the actual test.

B.1.1 Preliminary Hazard Anaylsis (PHA)

PHA is a design tool that helps in identifying and dealing with hazards in the early stages of design. It helps in recognizing and correcting hazards while the design is still in its earliest stages. The objective of PHA is to present the hazards and risks associated with the steam boiler by evaluating the likelihood and consequences of the major hazards and risks to the environment and people associated with the equipment. This will help system analysts to formulate proper measures to deal with these hazards. Hazard analysis should be subsequently performed to assess the ability of the design to minimize the harmful effects of the expected hazards. Failure Modes and Effects Analyses (FMEA), Failure Modes, Effects and Criticality Analyses (FMECA), and Fault Tree Analyses (FTA) are also commonly used to assess and minimize the hazards of a design [40].

B.1.2 Benefits of PHA

- It helps to ensure that the product is safe for use having all the hazards and risks identified.
- Modifications are less expensive and easier to implement in the earlier stages of design.
- It reduces the time spent during the design phase since there is a limited number of surprises during the development phase. In some cases, taking the time to perform a PHA may actually speed up the design process.

B.1.3 PHA Steps

- Identifying known hazards - is performed using a preliminary hazard matrix form which devides hazards into generic groups and associates potential failures with the generic hazards group.The system analysts fills in the potential hazards to the system together with a hazard checklist that has a list of specific hazards from sources such as equipment description, accident report data, past operational history of similar tasks and review of other historic records [41].

- Determining the causes and effects of the hazards - the causes for hazards are numerous and PHA attempts to identify all possible causes when the details of the design is defined in detail. The effects on personnel, equipment, facilities and operations are determined which enables PHA to estimate the overall effects of hazards or failures. The effects of the hazards may be categorized as follows:

  - Catastrophic - Causes multiple injuries, fatalities, or loss of a facility.
  - Critical - May cause severe injury, severe occupational illness, or major property damage.
  - Marginal - May cause minor injury, minor occupational illness resulting in lost workdays, or minor property damage.
  - Negligible - Probably would not affect the safety or health of personnel, but is still in violation of a safety or health standard.

- Determining the probability that the accident will be caused by the hazards - hazards are also classified according to probability of their occurrence within the system on the following cases.

  - Probable - Likely to occur immediately or within a short period of time.
  - Reasonably Probable - Probably will occur in time.
  - Remote - Possible to occur in time.
  - Extremely Remote - Unlikely to occur

- Establish initial design and procedural requirements to eliminate or control these hazardous conditions and potential failures.

B.1.4 Example of PHA

The following is an example of PHA analysis for a motor vehicle. In this case, one identifies the hazard that is posed to the motor vehicle followed by its cause. This will enable you to find the main effect of the hazard if it were to occur thereafter, the preventive action(s) that should be taken is listed in the last column.

Table B.1: Preliminary Hazard Analysis

Hazard	**Causes**	**Main effect**	**Preventive action**
Car skids	Bad weather	Car looses control	Use winter tires, traction control
Car overheats	Low level of coolant	Engine catches fire	Fill coolant
Car stalls	Low battery, low fuel in tank	Engine stops and doesn't start again	Replace battery
Tire burst	Worn out tires	Car looses control	Replace tires
Difficulty in steering	Disalligned wheels	Car steers to one side	Align wheels and steering column

B.1.5 Example of Hazards Checklist

These are examples of hazards that might appear on several systems during their operations, they will guide you in finding the possible failure of the steam boiler.

Common Causes
___Utility Outages
___ Moisture/Humidity
___ Temperature Extremes
___ Seismic Disturbance/Impact
___ Vibration
___ Flooding
___ Dust/Dirt
___ Faulty Calibration
___ Fire
___ Single-Operator Coupling
___ Location
___ Radiation
___ Wear-Out
___ Maintenance Error
___ Vermin/Varmints/Mud Daubers

Software Errors
__ Overflow {Too much work for the CPU}
__Wrong values
__ Wrong timing of message send/receive
__ Loss of data
__ Infinite loops
__ Errors in handling or interpreting data
__ Calculation errors

Control Systems
___ Power Outage
___ Interferences (EMI/ESI)
___ Moisture
___ Sneak Circuit
___ Sneak Software
___ Lightning Strike
___ Grounding Failure
___ Inadvertent Activation

Leaks/Spills (Material Conditions)
___ Liquids/Cryogens
___ Gases/Vapors
___ Dusts - Irritating
___ Radiation Sources
___ Flammable
___ Toxic
___ Reactive

Explosives (Initiators)
___ Heat
___ Friction
___ Impact/Shock
___ Vibration
___ Electrostatic Discharge
___ Chemical Contamination
___ Lightning
___ Welding (Stray Current/Sparks)

Figure B.1: Hazards Checklists [14].

B.2 System Description

The system includes two control units and several mechanical parts. Water is fed to a tank through a non-return valve by a pump, a heating element heats the water inside the tank and the steam generated is delivered to the industrial process through a valve. A safety release valve is included in case of too high steam pressure. Two control units, using several sensor, controls the process in order to have the right amount of water, heat and steam pressure in the tank.

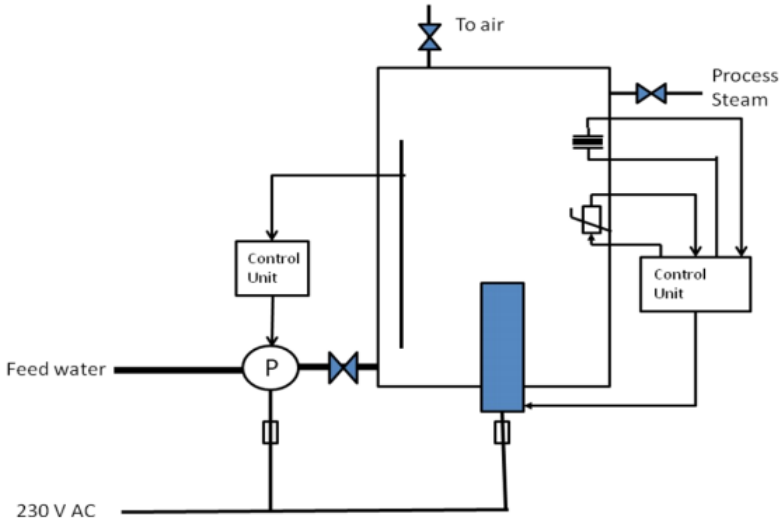


Figure B.2: Steam boiler concept.

System Detailed

This section describes the concept system in detail. See figure B.2 for an illustration of how the components are connected.

Functional Requirements

The steam boiler has several functional requirements that ensure that the system performs all the operations required to deliver steam to the industrial process and all the components function as expected. They are listed as follows:

- The steam boiler shall deliver steam at a predefined, constant pressure to an industrial process.
- Steam is produced by heating water using an electric heating element.
- The steam pressure is controlled by regulating the temperature setting on the heating element thermostat.
- The water level in the tank is controlled by a feeding pump which pumps water into the tank via a non-return valve.

- The safety of the steam boiler is taken care of by a safety valve that opens to air. The release pressure for the safety valve is fixed, based on the strength of the boiler.

Components

This section lists the components and their purpose in the system.

- Tank: A water tank designed to withstand high pressure.
- Water pipe: Leads the water from storage to the tank.
- Water pump: Regulates the water flow to the tank.
- Non-return valve: Prevent reverse water flow.
- Heating element: Provides heat to transform water into steam.
- Steam valve: Releases steam from tank to the steam pipe.
- Steam pipe: Delivers steam to the industrial process.
- Emergency release steam valve: In case of too high steam pressure, this valve opens.
- Water level sensor: Measures the water level and sends the value to the control unit.
- Water level control unit: Controls the water pump.
- Temperature sensor: Measures the temperature in the tank.
- Steam pressure sensor: Measures the steam pressure in the tank.
- Steam pressure control unit: Controls the heating element.
- Wiring: Wires connecting the components to each other and to electric power.
- Electricity: Power is provided from a 230 V AC outlet.

B.3 Analysis

B.3.1 Instructions

1. Identify hazards that can ocurr in the system.
2. Fill them into the table below together with causes, main effect and, if you can think of any, preventive actions.

Table B.2: Preliminary Hazard Analysis

Hazard	**Causes**	**Main effect**	**Preventive action**

APPENDIX C

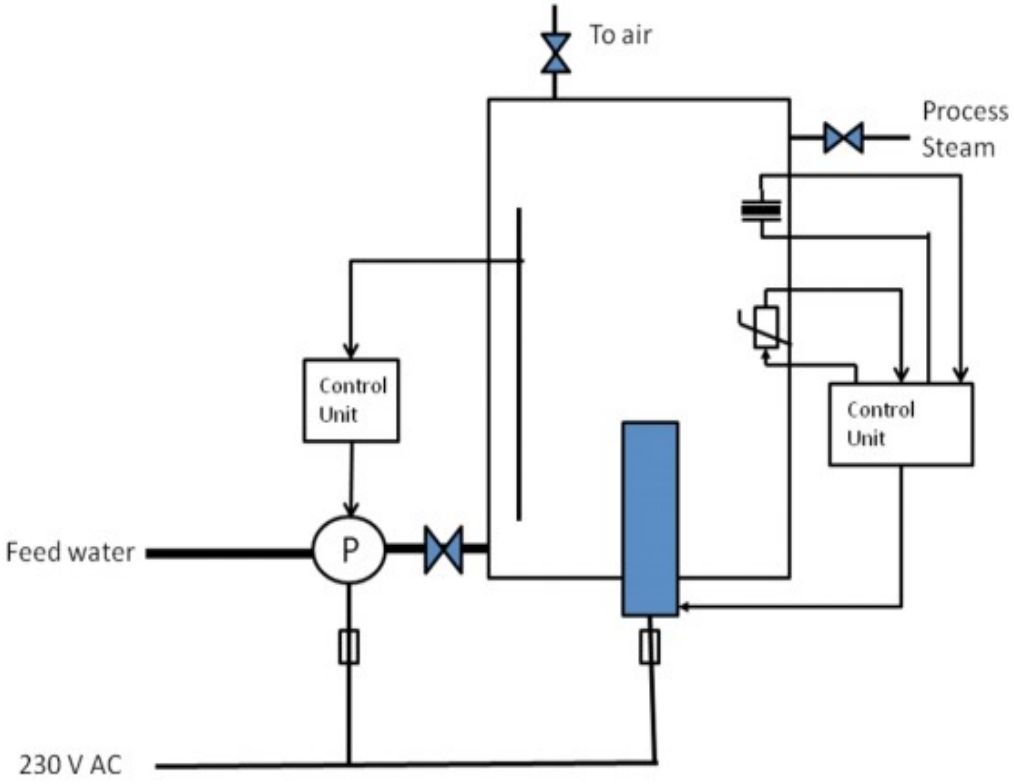
Experiment Presentation

This is the small presentation we had prior to the student experiment. The purpose was to introduce hazard analysis, and prepare the students for the experiment. We discussed the boiler system concept so that they would understand better how the system worked and the components purpose. This was also described in the handout, but a short introduction of it all was important for them to get an idea of what they were about to do.

Safety Analysis

- The process of identifying faults in systems that can harm personell or the environment.
- It is important to identify these faults as early as possible.

Steam Boiler



Outline

- The groups shall consist of 5 people.
- Read through the papers to get an understanding of the task.
- Use the forms provided, deliver a group result.

APPENDIX D

Results for Individual Components

This section includes all the results for the individual components. Each line represents a group (group A first, group B second etc.) and how many hazards they identified for this component.

ID 1 - Tank	
PHA	Misuse Case + HAZOP
1	0
1	0
1	1
1	0
1	1
	0

Table D.1: Component 1

ID 2 - Water pipe	
PHA	Misuse Case + HAZOP
0	1
1	0
1	1
0	1
0	1
	1

Table D.2: Component 2

ID 3 - Water pump	
PHA	Misuse Case + HAZOP
0	1
0	1
1	0
1	1
1	0
	1

Table D.3: Component 3

ID 4 - Non-return valve	
PHA	Misuse Case + HAZOP
1	0
0	2
1	0
0	2
1	1
	1

Table D.4: Component 4

ID 5 - Heating element	
PHA	Misuse Case + HAZOP
1	1
0	1
0	2
0	0
1	0
	0

Table D.5: Component 5

ID 6 - Steam valve	
PHA	Misuse Case + HAZOP
0	0
1	1
1	0
0	2
1	1
	2

Table D.6: Component 6

ID 7 - Steam pipe	
PHA	Misuse Case + HAZOP
0	1
1	0
0	1
0	0
1	0
	0

Table D.7: Component 7

ID 8 - Emergencay release steam valve	
PHA	Misuse Case + HAZOP
0	0
1	0
1	1
1	1
1	1
	0

Table D.8: Component 8

ID 9 - Water level sensor	
PHA	Misuse Case + HAZOP
1	3
1	2
1	3
1	2
1	1
	2

Table D.9: Component 9

ID 10 - Water level control unit	
PHA	Misuse Case + HAZOP
0	3
1	2
0	2
1	2
1	3
	2

Table D.10: Component 10

ID 11 - Temperature sensor	
PHA	Misuse Case + HAZOP
0	2
0	2
1	2
0	2
0	1
	2

Table D.11: Component 11

ID 12 - Steam pressure sensor	
PHA	Misuse Case + HAZOP
1	2
0	1
1	1
1	2
1	2
	1

Table D.12: Component 12

ID 13 - Steam pressure control unit	
PHA	Misuse Case + HAZOP
1	2
0	1
0	3
1	2
0	3
	1

Table D.13: Component 13

ID 14 - Wiring	
PHA	Misuse Case + HAZOP
1	0
0	0
1	1
0	0
0	0
	0

Table D.14: Component 14

ID 15 - Electricity	
PHA	Misuse Case + HAZOP
1	0
0	0
1	0
1	0
0	0
	1

Table D.15: Component 15

APPENDIX E

Steam Boiler Pilot Application System

This section includes the steam boiler system analysis conducted by Tor Stålhane and Tormod Wien. It was this concept that was used to test the procedure with the students. They, however, were not given the whole analysis, only the concept diagram and a description of the system.

ABB boiler pilot application

Tor Stålhane, NTNU
Tormod Wien, ABB

Project concept

- A boiler that can deliver steam at a predefined pressure to an industrial process.
- Heat is supplied through an electric heating element.
- The steam is supplied through a valve to the industrial process.
- The tank has a safety valve that opens to the air. Water is fed to the tank through a pump via a non-return valve.
- Control units:
    - Water level: The pump is coupled to a control system together with a water level indicator to keep the water level between predefined max and min levels.
    - Steam pressure: The heating element is coupled to a control system together with pressure indicator and a temperature indicator.

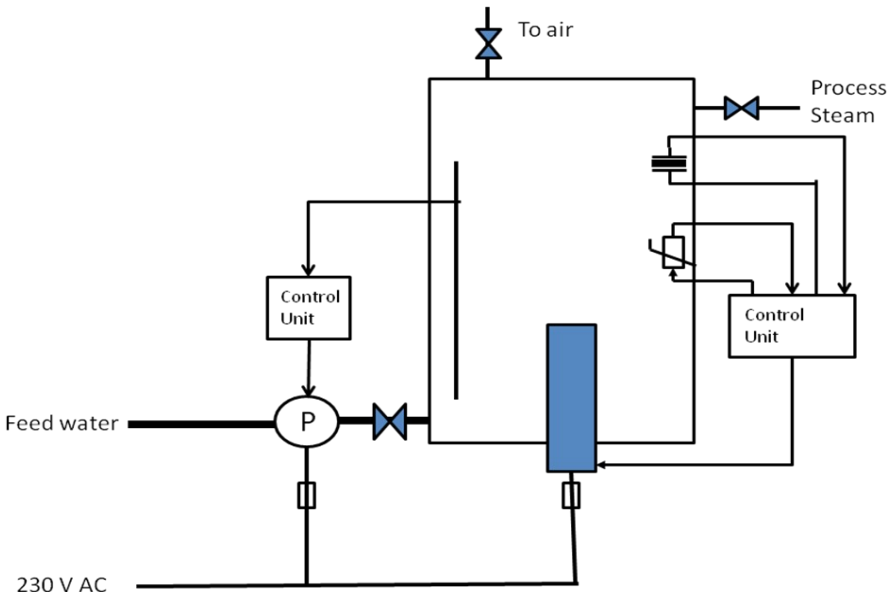


Figure 12: Concept diagram

The two control units for water level and steam pressure are shown below:

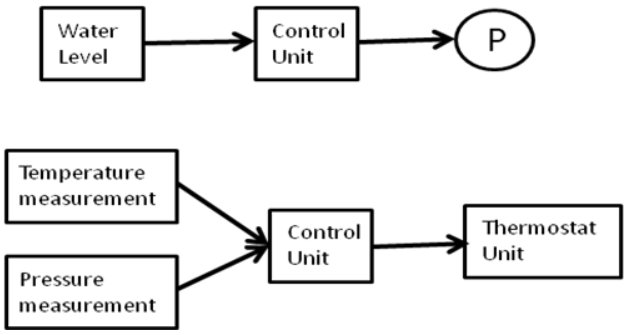


Figure 34: Control loops for water level (top) and steam pressure

Functional requirements

1. The steam boiler shall deliver steam at a predefined, constant pressure to an industrial process.
2. Steam is produced by heating water using an electric heating element.
3. The steam pressure is controlled by regulating the temperature setting on the heating element thermostat.
4. The water level in the tank is controlled by a feeding pump which pups water into the tank via a non-return valve.
5. The safety of the steam boiler is taken care of by a safety valve that opens to air. The release pressure for the safety valve is fixed, based on the boiler's strength.
6. The system shall be SIL2 certifiable.

Preliminary HazOp – PHA

Based on the diagram in chapter 1, we ran a PHA. The result is shown in the table below.

Hazard	Cause	Main effect	Preventive action
Too high pressure in the tank	Not able to turn off the heating (sensor, control, actuator, connections)	Boiler explodes	Safety valve
			Turn off the heat
	Feeding pump failure (too strong)	Boiler rupture	Turn off power to the feed pump
Too high water level	Water level regulation failure (sensor, control, actuator, connections)	Water to the process	Pump emergency stop
Too high pressure in the feed pipe	Non-return valve failure	Release boiling water to the water supply	Two non-return valves in series
			Emergency valve for releasing pressure

The tank is too hot	Too little water and too much heat (sensor, control, actuator, connections)	Tank gets hot/fire	Turn off the heat
			Add water?
Unintentional leaks	Corrosion	People get scalded	Inspection, collector tray or quality assurance
	Bad welding/fittings	People get scalded	Inspection, collector tray or quality assurance
Electric shock	Short circuit	People get hurt/killed	Fuses
Flooding	Breakage in pipes	Damage to equipment and/or environment	Flow meter, collector tray

Safety requirements

Based on the diagrams in chapter 1 and the PHA in chapter 3, we have defined the fooling safety requirements:

- Too high water level: If the water level in the boiler tank exceeds the max critical limit, the feeding pump shall be shut down. This hazard can be caused in two ways:
  - Water level sensor failure. In this case it is enough to stop the pump
  - Pump failure – pump cannot be stopped. In this case we need to cut the pump's power.
- Too high pressure in the feed pipe:
  - The feeding pipe must be designed to take the same pressure as the boiler tank.
  - To keep the pressure in the feeding pipe below the critical pressure, one of the following safety mechanisms could be implemented.
    - Two non-return valves in series
    - Emergency valve in the feeding pipe to release the pressure
- The tank is too hot: The tank shall be equipped with a control system that receives temperature measurements from an external temperature sensor and can send turn-off signals to the heating element. The turn-off temperature is set to TBD centigrade. Since the reason for this hazard can be a s.a. on for the heating element we need to cut the heating element's power.
- Unintentional leaks: The tank and all fittings shall be inspected at regular intervals no longer than TBD. The boiler tank shall be placed within a collector tray with a holding capacity of minimum TBD liters.
- Flooding: The feeding pipe shall be equipped with a flow meter. The flow meter will stop water from entering the tank at TBD liters.

The two hazards "Too high pressure in the tank " and "Electric shock " are taken care of by the use of a safety valve and fuses for the heating element and the feeding pump. The resulting system – all safety requirements fulfilled – is shown in the diagram below. The type

of indicator is indicated by a letter – P for pressure, T for temperature and L for water level. There are two temperature indicators – one for the water temperature and one for the temperature on the outside of the tank.

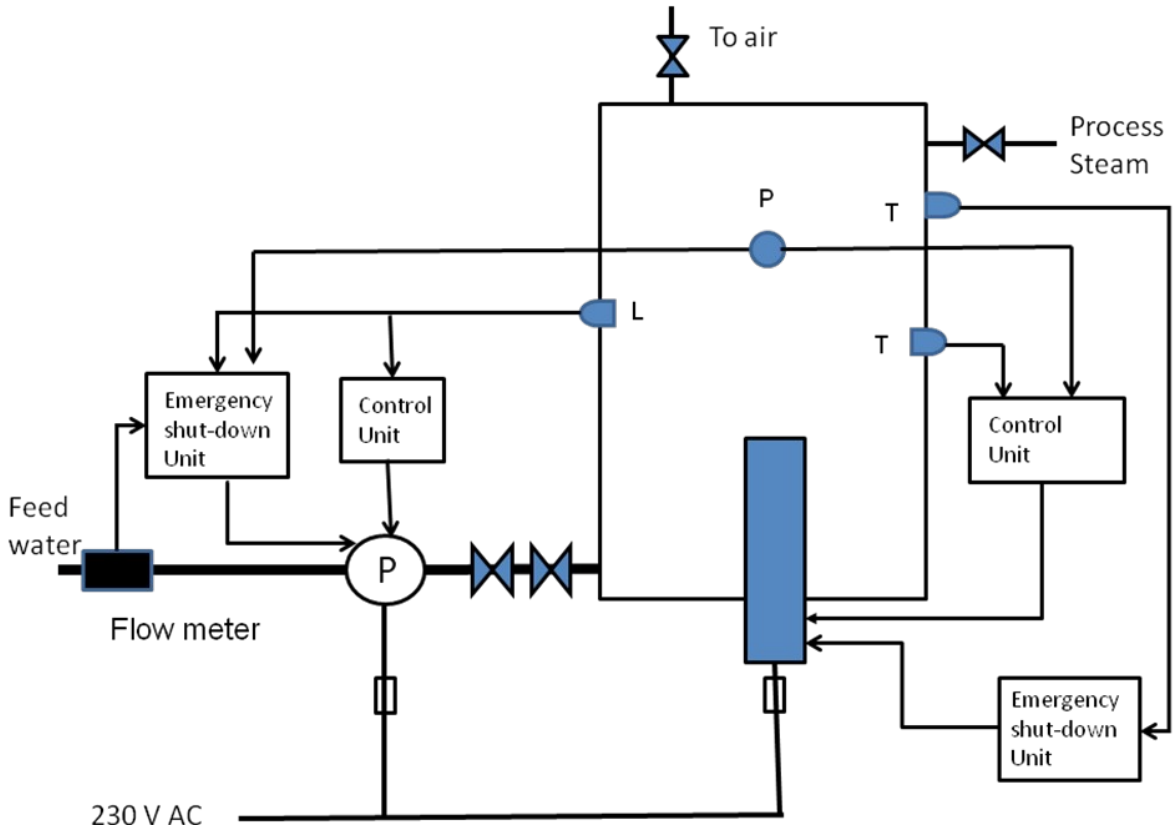


Figure 3: Concept diagram with safety control units included

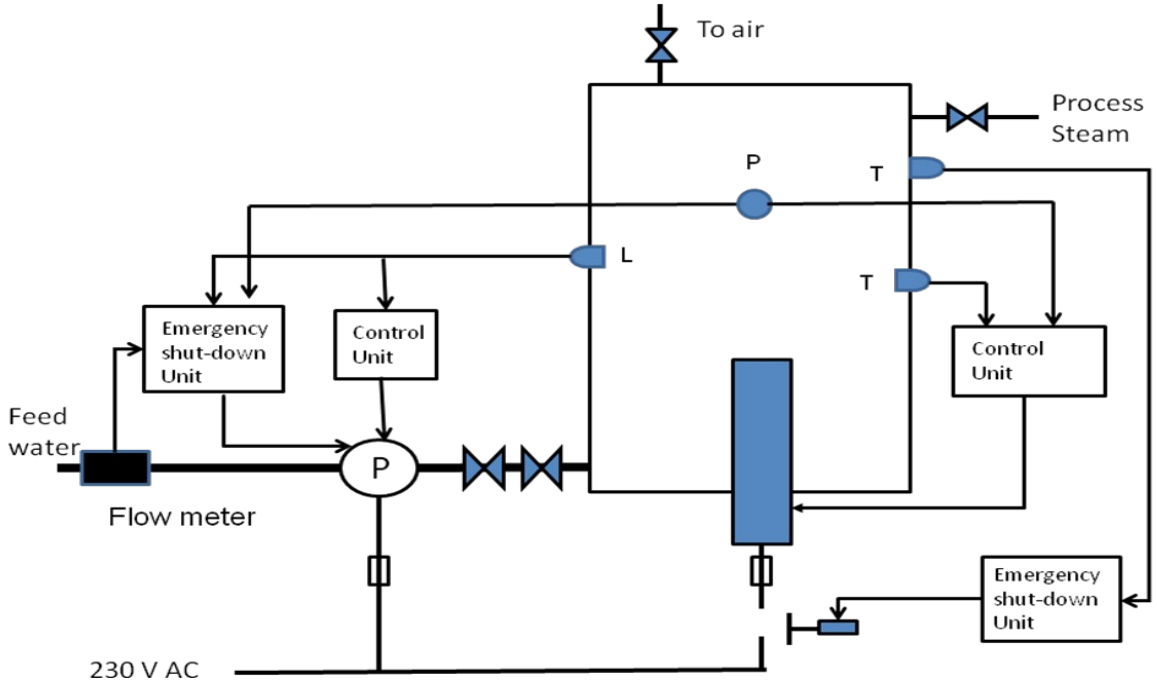


Figure 3: Concept diagram with power control for heating eleement included

The two emergency shutdown units for the feeding pump and the heating unit are shown below.

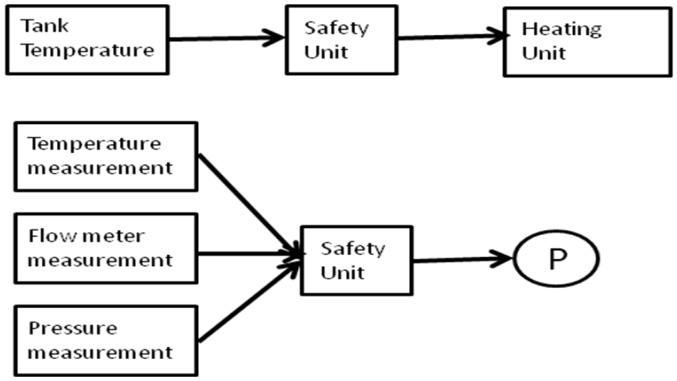


Figure 44: Safety control loops for steam pressure (top) and feeding pump

Possible, planned extensions

At the meeting at ABB on August 25, we agreed to later – next version – include the following extensions to the boiler:

- Feed-water tank
- Blow-down valve with flash tank

Boilerplate requirements

The code BP<no.> refers to the boilerplates used. Based on the available boilerplates and decomposition rules, the safety and functional requirements formulated using boilerplates are as follows:

6.1 Functional requirements

1. BP2:
   The <steam boiler> **shall be able to** <deliver> [<steam> **to** <an industrial process>]
2. BP2
   The <steam boiler> **shall be able to** <produce> [<steam> **using(**<electrical> <heating element>**)**]
3. BP2
   The <steam boiler> **shall be able to** <control> [<steam pressure> **using(**<thermostat> of <electrical> <heating element>**)**]
4. BP2
   The <steam boiler> **shall be able to** <control> [<water level> **using(**<feeding pump>**)**]
   The <feeding pump> **shall be able to** <deliver> [<water> **using(**<non-return valve>)]
5. BP63, BP64
   If [<steam pressure> **greater than** <critical pressure level>] **then the** <steam boiler> **shall** [<open> <safety valve>]

6.2 Safety requirements

1. BP63, BP64
   If [<water level> **greater than** <max water level>] **then the** <safety system> **shall** [<stop> <feeding pump>]
2. BP63, BP64
   If [<steam pressure> **greater than** <max pressure level>] **then the** <safety system> **shall** [<stop> <feeding pump>]
3. BP63, BP64
   If [<feeding pipe pressure> **greater than** <max pressure level>] **then the** <safety system> **shall [**[<stop> feeding pump>] **and** [<open > <pipe release valve>]]
4. BP63, BP64
   If [<external temperature> **greater than** <max external temperature>] **then the** <safety system> **shall** [<cut power~~turn off~~> <heating element>]
5. BP32, BP33
   The <user> **shall be able to** [<inspect> **the** <steam boiler>] **at a minimum rate of** <TBD> **times per** <year>
6. BP63, BP54
   If [<water flow> **greater than** <max water flow>] **then the** <safety system> **shall** [<stop> <feeding pump>]