AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

Master's Thesis in Computer Engineering

A Co-Simulation Based Approach for Developing Safety-Critical Systems

By Daniella Tola
201409551

Supervisor: Prof. Peter Gorm Larsen

Aarhus University Department of Engineering
2nd June 2020

Daniella Tola
201409551

Peter Gorm Larsen
Supervisor

Abstract

Safety-critical, cyber-physical systems may pose a threat to their environment, potentially causing harm to or even loss of human lives. A common approach to ensuring the safety of a cyber-physical system, is formulating a safety case, which consists of a set of claims, which is backed by arguments and collected evidence. For a complex system operating in a dynamic and unpredictable environment, producing this evidence by means of traditional methods is challenging. An alternative approach is to create models and use simulation to test the system safety under different conditions. Due to the heterogeneous nature of the system's mechanical, electrical and software components, it can be challenging to create a single monolithic model of the system. Co-simulation provides a solution to this by enabling simulation of the entire system using models of its parts.

This thesis proposes a safety case development process which incorporates co-simulation in the creation of evidence. The proposed process describes when it is suitable to use co-simulation in the development process, and how the co-simulation results can be used as safety evidence. The proposed process is demonstrated on a case study of an autonomous combine harvester.

Preface

This Master's thesis is the final project of the Master's degree in Computer Engineering at Aarhus University. It was conducted from January 27th to June 2nd 2020 and accounts for 30 ECTS points.

I would like to thank my academic supervisor, Peter Gorm Larsen, for being committed, always giving valuable feedback and pushing me to do my best. I would also like to thank Christian Møldrup Legaard for the interesting discussions and the feedback during the development of this thesis.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Overview

The level of automation in systems is increasing, causing the complexity of these systems to increase simultaneously. Many autonomous systems operate in environments with humans and other living creatures. If such systems fail they may pose a threat to their surroundings, thus they are called safety-critical systems [1].
Developing safety-critical systems is a complex and time-consuming process, where a large amount of effort is spent on designing, testing and demonstrating the system is sufficiently safe to deploy [1]. The verification of such systems is not an easy task, due to the complexity of the system and its operating environment. For example, verifying the safety of an aircraft during a thunderstorm is both difficult and unethical to test. It would require flying in these specific weather conditions and pilots willing to risk their lives to fly the aircraft [2]. The cost of developing safety-critical systems with current technologies is high, and therefore alternative technologies should be investigated [1].
As [1] and [3] conclude in their research, alternative methods for the development and verification of safety-critical systems must be explored. Modern technologies which potentially can be incorporated in the development process of a safety-critical system are model-based systems engineering, formal verification, simulation and co-simulation [3]. The ISO 26262 standard describes methods to ensure the functional safety of autonomous vehicles, where for highly automated systems it is recommended to perform simulations to verify the system safety [4]. Simulation is defined as the operation of the model of a system. Simulations allow configuring the parameters of the model, allowing to perform experiments [5]. Co-simulation can be defined as simulating a system, using various models developed using different tools. A co-simulation engine exchanges the data between the different models, and each tool which the model was developed in, must progress the model in time. Co-simulation allows using models from different Original Equipment Manufacturers (OEMs), which are developed in different tools.
This thesis project defines a process for the development of safety-critical systems, which incorporates modelling and co-simulation for the assurance of such systems. This process will be used in the development of the safety case of an autonomous combine harvester, which consists of several components manufactured different OEMs. Figure 1.1 gives an

overview of the main stages in the development process. By incorporating co-simulation in the assurance of safety-critical systems, it allows testing such systems in environments that would either be too difficult, expensive or unethical to test the real physical system in.

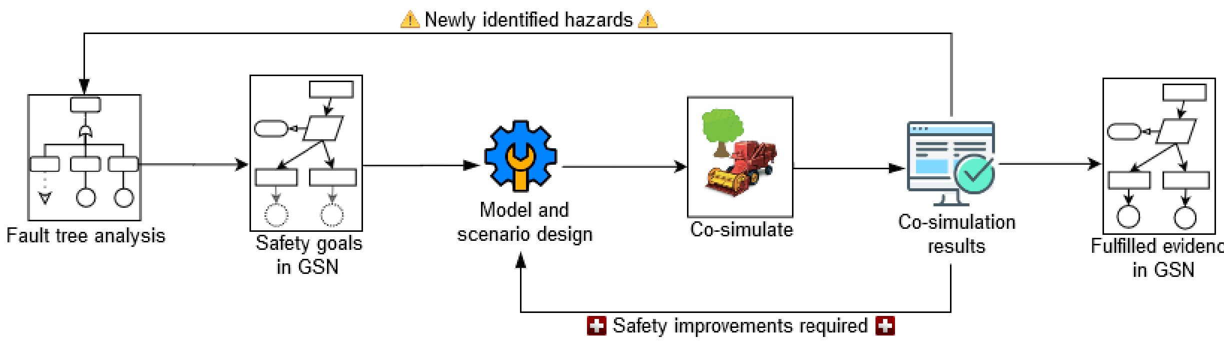

⚠ Newly identified hazards ⚠
Fault tree analysis
Safety goals in GSN
Model and scenario design
Co-simulate
Co-simulation results
Fulfilled evidence in GSN
Safety improvements required

Figure 1.1: An overview of the method of developing a safety case using co-simulation in the development process. The complete process is explained in chapter 5.

1.2 Motivation

To determine the safety properties of a system, a safety case will typically be created. The purpose of a safety case is to demonstrate that all the necessary precautions regarding the system safety have been taken. A safety case consists of safety goals, evidence that supports these goals, and arguments that describe the connection between the goals and the safety goals. The evidence in such safety cases can be obtained using different techniques, such as hazard logs of the operation of similar systems or mathematical proofs of specific characteristics of the system [6]. Subsection 1.2.1 describes the case study used in this thesis project. The case study will be presented for the reader to better understand the background of this thesis and what the challenges of developing such a system may be with regards to its safety aspects.
The motivation for this thesis is to propose a development process suitable for autonomous systems with focus on obtaining evidence, from co-simulations of a model of the system. This thesis incorporates co-simulation into the development process of a safety-critical system to demonstrate how and when it is reasonable to use it.
Using co-simulation for assurance of the system allows the system to be tested in specific environments, that in the real world may be difficult to realize and re-validate. It also allows testing the system under conditions that may not be feasible to test on the real system.
It is important to consider when it is advantageous to use co-simulation results as evidence compared to other types of evidence. The subsection 1.2.2 gives an overview of when it is suitable to use co-simulation and describes some of the resulting challenges.

1.2.1 Case Study

The case study is based on the development of an autonomous combine harvester. The purpose of the system is to harvest crops without the need for a driver to control the vehicle.

Figure 1.2 depicts a combine harvester. Such a combine harvester can be more than 10 meters long and more than 3 meters wide [7]. Combine harvesters are heavy vehicles that may potentially weigh more than 15 tons [8]. The system to be developed will be referred to using the acronym ACH throughout this thesis.

Figure 1.2: A 3D model of a combine harvester, specifically the Fendt IDEAL 9T. This image was taken using the *Fendt IDEAL AR* app.

The vehicle will be driving on crop fields without the requirement of a human monitoring this process. The system will only drive autonomously within the boundary of the crop field which is specified by the operator. Outside this boundary a driver is required to control the vehicle. This means that the software and hardware of the ACH is developed with focus on the system driving autonomously in crop fields, and for example not on roads with autonomous cars.
The system will mainly be operating in dry weather conditions, but the system must be able to drive in all types of weather conditions, i.e. rain, fog, etc. The system will be operating both during day- and night-time, meaning different types of sensors and lights must be mounted on the system.
The maximum speed of a combine harvester is approximately 40 km/h [9], where the optimum speed while harvesting is around 8 km/h. The speed of the harvester will vary depending on the type of crop being harvested and depending on the soil conditions [10].
This type of autonomous system may pose a threat to living beings on the crop field, or unripe crops that are not ready to harvest. The system therefore falls in the category of safety-critical systems. Figure 1.3 illustrates a autonomous combine harvester, with sensors used to observe its environment. As the figure shows, animals such as fawn may be hiding in between the crops, making it difficult to detect these animals, and drive safely on the crop field. Since the system must operate autonomously in a dynamic, unpredictable environment, it is necessary to mount different types of sensors, such as cameras, LiDARs and radars, as illustrated in figure 1.3, to ensure detecting obstacles at different ranges.
The focus of this thesis is the autonomous part of the vehicle, where the safety must be ensured. Some of the main challenges of developing such a safety-critical system are testing the system in:

- Controlled environments, for example testing the system in specific weather conditions. To be able to test this, it would require waiting for the correct weather conditions and

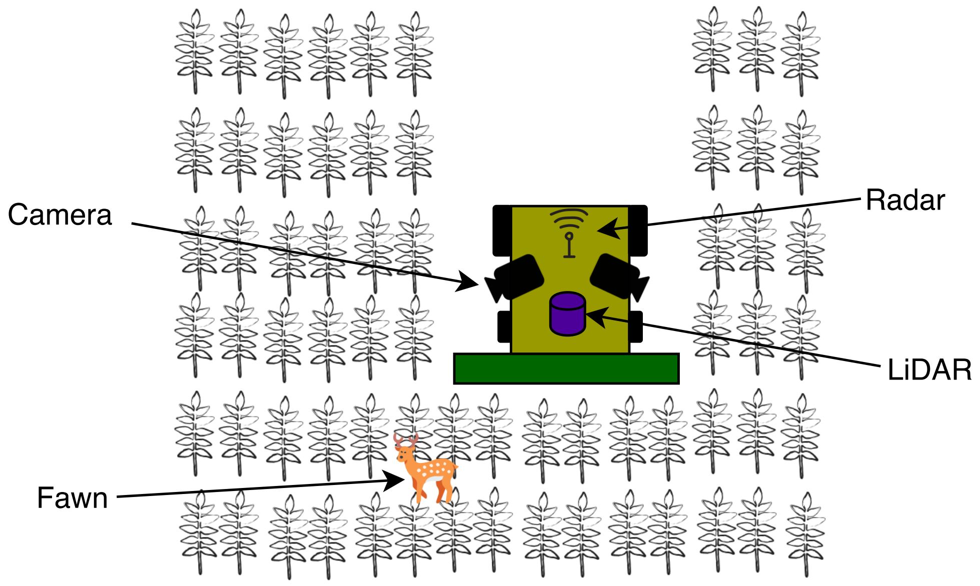


Figure 1.3: An illustration of the autonomous combine harvester to be developed. The combine harvester must drive autonomously in a crop field, using various sensors to observe its environment. The sensors are shown on the combine harvester.

waiting for the crops to be ripe or ready to harvest.

- Scenarios where a living being is on the field. For example testing the safety of the system when there may be living creatures camouflaged or hiding in bale. Performing this test is unethical, since the system is still under development, where there is a potential for the system to not operate safely in these conditions, potentially running over living creatures.

These challenges are some main reasons for attempting to incorporate co-simulation in the development of the safety-critical ACH. The main idea is to perform co-simulations of the ACH, and use these co-simulation results as evidence in the safety case.

1.2.2 Using Co-simulation

In general when working with a problem where there is an aspect of uncertainty in a random manner, it is suitable to model and simulate this [5]. Simulation is limited to models that are developed in the same tool. Co-simulation extends simulation, with the ability to simulate models developed in different tools. This makes more models available for reuse in co-simulation compared to simulation.
The descriptions of when it is suitable to use simulation for a problem, described in [5] can also be extended to co-simulations related to safety-critical systems:

- Impossible, unethical or expensive to test the behaviour of a system in the real world.

- Too difficult or impossible to create a mathematical proof that can be used as evidence to support safety goals.
- Expensive or difficult to validate the mathematical model of the system.

Co-simulation may be a time-consuming process considering the maturity of the currently available tools [11, 12].
There are a number of challenges when performing simulations, as described in [5], where some of these challenges can be extended to co-simulation:

- Using co-simulation in a case where using an analytical solution is appropriate and less time-consuming. For example if evidence is being constructed to prove a geometric characteristic, then using a geometry proof will be more suitable than performing a co-simulation.
- Ensuring the model of the system is valid. When modelling a system, necessary assumptions are made, and therefore it can be difficult to ensure the complete validity of the model.
- Determining how the environment affects the real system and how modelling this can be accomplished in a faithful manner.

Reviewing the advantages of co-simulation compared to the challenges, it is clear that these challenges can be limited, by performing a thorough analysis before using co-simulation, and by defining a thorough procedure for performing model validation.

1.3 Goal

The overall goal of this thesis project is to create a safety case development process that incorporates co-simulation results as evidence for supporting system safety claims.
This thesis describes the main steps that were taken to develop a safety case which incorporates co-simulation, and the benefits and limitations of this process. The main goals of this thesis are:

Goal 1: Define a procedure for incorporating co-simulation into the development process of a safety-critical system.

Goal 2: Demonstrate the feasibility of the process in the context of a case study.

Goal 3: Establish how safety-case analysis can be used to feed into the process of defining co-simulation scenarios.

Goal 4: Improve personal research related skills and improve documentation skills of defining a development process.

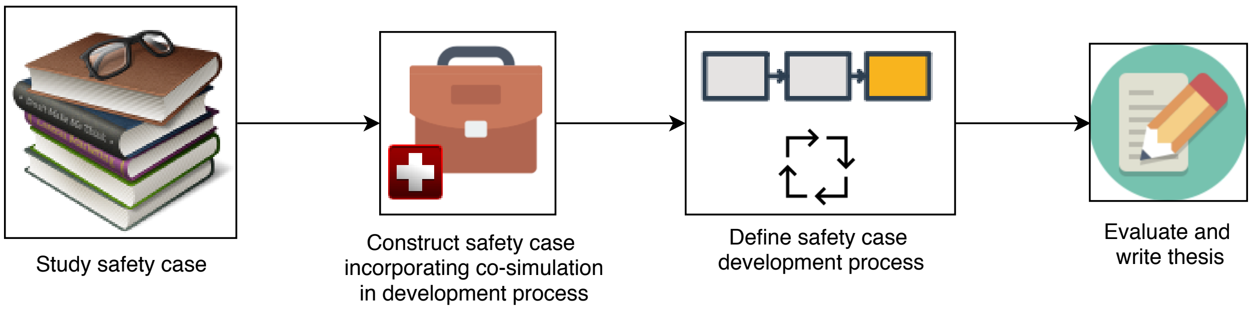

Study safety case
Construct safety case incorporating co-simulation in development process
Define safety case development process
Evaluate and write thesis

Figure 1.4: An illustration of the approach taken in this thesis, showing the main steps performed.

1.4 Approach

The approach of this thesis, to reach the goals described above, is illustrated in figure 1.4. This is naturally not a sequential process, which means the steps will mainly be performed in the given order, but can also be iterative. The description of each step of the approach are:

Study safety case: Research what a safety case is, why and when it is used. Research the development processes used to develop safety cases.

Construct safety case using co-simulation: Construct a safety case for the case study. The development of the safety case must incorporate co-simulation results as evidence.

Define development process: Based on the development process of the safety case constructed for the case study, define a safety case development process that incorporates co-simulation, and describe the benefits and challenges.

Evaluate and write thesis: Evaluate the pros and cons of incorporating co-simulation to the safety case development process performed on the case study, to determine in which situations co-simulation is suitable to use. Document the development process, the benefits and the challenges, and create a guide describing the steps to perform in the process.

1.5 Reading Guide

References: references are referred to using a number surrounded by square brackets, e.g. [1]. This refers to the first referenced artefact, which can be found in the Reference list of this thesis.

Glossary: The first occurrence of words described in the glossary, is denoted by the *italic* font style followed by a superscript of τ. An example of this is *cyber-physical system*$^{\tau}$.

Acronyms: Acronyms consisting of multiple words will be expressed with capital letters, after introducing the long version of the acronym first. For example the first time the acronym is introduced in a chapter, it will be written as follows: Cyber-Physical System (CPS). The second time the acronym is used in the same chapter, it will be written as CPS.

Quotations: When a quotation is used directly from literature, it is written in *italic* and surrounded by quotation marks. An example illustrating the style of a quotation is "*This is an example of a quotation*".

Emphasize: Emphasized words or names are written with quotation marks. An example illustrating an emphasized word is "emphasized".

Numbering of Figures, Listings and Tables: Figures, listings and tables are referred to using their name, followed by the chapter number and a period, followed by the figure number of that chapter. For example the first figure in chapter 1, would be referenced as figure 1.1.

Code Elements: Elements of code are written using a listing, with coloured text indicating the keywords of the used programming language. An example of a code element written in Python is illustrated in the listing below.

```python
    def my_example_function(a,b):
        c = a + b
        return c
```

Listing 1.1: An example of a code element written in Python.

1.6 Structure

This section describes the structure of the thesis. Figure 1.5 illustrates how the different chapters are connected to each other. For example chapter 2 describes the background theory, and therefore it must be read to understand the following chapters. Chapter 5 describes the process developed in this thesis, which is based on the work accomplished in chapters 3 and 4. Chapter 6 presents concluding remarks on this thesis project.

Chapter 2: Describes the theory that is necessary to understand the concepts used in this thesis.

Chapter 3: This chapter describes the main outcome of the safety analysis performed on the case study. It describes the safety goals, evidence and safety arguments that are required to demonstrate the safety of the system. Details of the safety analysis can be found in appendices A and B.

Chapter 4: This chapter describes how to define co-simulation scenarios from the outcome of the safety analysis. It then describes how the system, and the chosen scenarios are

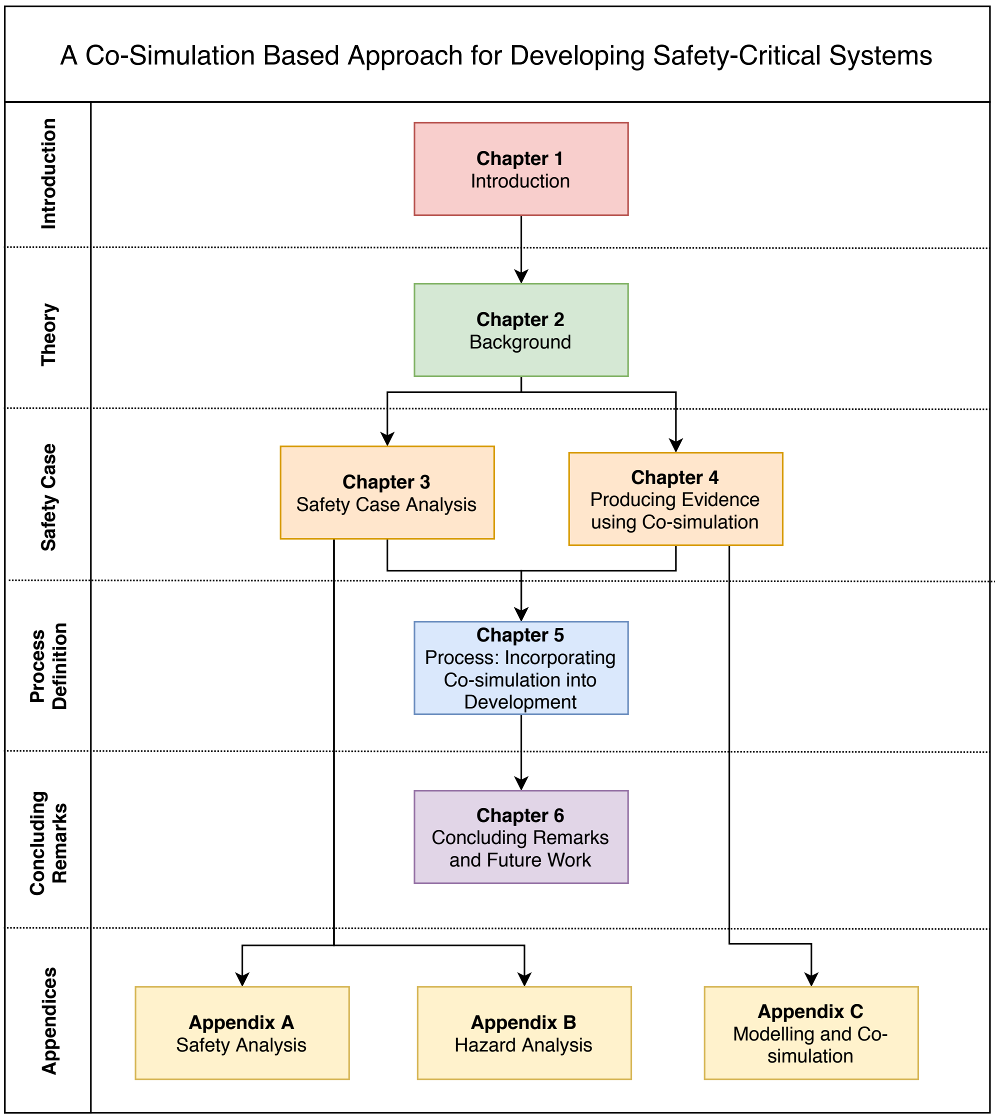

A Co-Simulation Based Approach for Developing Safety-Critical Systems
Introduction
Theory
Safety Case
Process Definition
Concluding Remarks
Appendices
Chapter 1 Introduction
Chapter 2 Background
Chapter 3 Safety Case Analysis
Chapter 4 Producing Evidence using Co-simulation
Chapter 5 Process: Incorporating Co-simulation into Development
Chapter 6 Concluding Remarks and Future Work
Appendix A Safety Analysis
Appendix B Hazard Analysis
Appendix C Modelling and Co-simulation

Figure 1.5: An illustration of the structure of the chapters in this thesis.

modelled. Co-simulations of the system in the specified scenarios are performed, and the results of these are used as evidence in the safety case. Details of the models and co-simulations can be found in appendix C.

Chapter 5: This chapter gives an overview of the process defined in this thesis. It documents a guide on how to apply this process when developing a safety case, and explains the pros and cons of using co-simulation in such a process.

Chapter 6: This chapter concludes the work of this thesis, and briefly describes potential future work areas.

Appendix A: This appendix contains the details of the safety analysis performed on the system, consisting among others of the system description and relevant safety standards used for defining safety requirements. This is part of the safety analysis performed in chapter 3.

Appendix B: This appendix contains the full hazard analysis performed on the system. This is part of the safety analysis performed in chapter 3.

Appendix C: This appendix contains details on the modelling and co-simulation tools used and the FMUs modelled in this thesis. This is part of the model design performed in chapter 4.

Chapter 2

Background

2.1 Introduction

This chapter describes the background information required to understand the work of this thesis project. Section 2.2 describes what a safety-critical system is, followed by section 2.3 which describes the main components of a safety case . Finally, section 2.4 gives a brief overview of the concept of modelling and performing co-simulations.

2.2 Safety-Critical Systems

A safety-critical system is a system which if fails, may have catastrophic consequences which may cost human lives, or result in damaging property or the environment. Examples of such systems are medical devices, avionic systems and self-driving cars. Each of these examples contains software that controls parts of the physical system.

One of the main challenges in safety-critical systems is assuring the safety of the system. There are multiple factors that may play a role in the system safety, among others software failures, inconsistent or incomplete system requirements, and system design errors [6].

Analysing the safety of software in safety-critical systems may be a demanding process, especially in cases where the software consists of thousands of lines of code [1]. It only takes one of these lines of code to contain an error for the system to possibly end in an unsafe state.

If the system requirements are incomplete or inconsistent, the software may contain these inconsistencies, which may lead to the system changing to an undefined state, where it may be unsafe.

An example of a safety-critical system which ended in an unsafe state due to system design errors, is the Lufthansa Flight 2904[1], which ended with running off the edge of the runway and catching fire. A safety feature in the plane did not allow it to brake in the first few seconds after it landed on the runway. The cause of this was due to the stormy weather conditions, that were not taken into account during the system design [6]. This shows that

[1]https://en.wikipedia.org/wiki/Lufthansa_Flight_2904

even in cases where no software failures occur, safety-critical systems can end in unsafe situations.
Many safety-critical systems are Cyber-Physical Systems (CPSs), which operate in dynamic environments. This adds a complexity to the development of such systems, since they must be able to operate safely under different potential environmental conditions that may occur. These examples illustrate the necessity of ensuring the safety of such systems before deploying them to the market. Many safety-critical systems are regulated by government bodies to ensure these systems are safe to deploy [6]. The next section describes a method used to demonstrate the safety of such systems.

2.3 Safety Case

This section describes what a safety case is and specifies some of the main tools used in developing a safety case. Subsection 2.3.1 gives an overview of a safety case and what it is used for. Then subsections 2.3.2 and 2.3.3 describe two of the main components of a safety case that are used in this thesis, hazards and risk assessment. Afterwards, subsection 2.3.4 describes a graphical notation for demonstrating a safety case. Finally, subsection 2.3.5 gives an overview of the development process of a safety case.

2.3.1 Overview

Safety cases are typically used to demonstrate to a government regulatory body that a developed system can operate safely in a specified environment [6].
A safety case is defined as [13]:

> "*A documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment.*"

The three main elements of a safety case are [13, 14]:

Claim: this may be a claim about the safety of the system or its subsystems. It may also be a safety requirement. An example of a system claim is "*The system can tolerate single component failures*".

Evidence: this is used to support the claims about the safety of the system. Evidence can for example be facts based on research or assumptions that are necessary. An example of using a fact as evidence is using a hazard log from a similar already deployed system, to prove that no hazards occurred, or that the hazards that may occur are registered and the system is designed to take care of these cases. An example of using an assumption as evidence is using the *mean time to failure*$^{\tau}$ of a component as part of how safe the system is.

Argument: the argument is used to connect the evidence with the safety claims, and describe how the evidence supports these safety claims.

Examples of the documents created throughout the development of a safety case are system descriptions, hazard analysis, system requirements, static analysis of the code, development processes used, the competences of the developers etc. [6]. The subsections below describe the main elements of the safety case that have been used in this thesis.

2.3.2 Hazards

This section describes the main characteristics of hazards in safety cases.
A hazard is defined as a system condition which can lead to the injury of human(s), damage to property or the environment. If a system is in a hazardous state, it does not imply that an accident will occur, but merely that if the right environmental conditions take place, then an accident may occur [6, 15].
Potential hazards in a safety-critical system are typically identified while designing the system. This is done by researching safety standards related to the system category, working with domain experts, and researching identified hazards in similar systems [6]. In the development process of safety-critical systems, improvements will typically be made, leading to a change in the system design. As the system design evolves, the hazards associated with the system will potentially change, meaning that a hazard identification must be performed on the evolved system design. Hazard identification is an iterative process as additional hazards may be identified during the development of the system [16].
The analysis of hazards is determining the potential root causes of events leading to the occurrence of the hazard. There are two main approaches for analysing hazards: top-down and bottom-up approaches. The top-down approach starts with a hazard and moves towards events that can lead to this hazard. The bottom-up approach starts with component failures and moves towards a hazard that may result from this failure.
A common top-down technique used is fault-tree analysis, where the events that may lead to the occurrence of the hazard are presented graphically using AND and OR gates [6].

2.3.3 Risk Assessment

Risk assessment is based on determining the severity and likelihood of the identified hazards. When determining the risk of a hazard it should be based on the operation of the whole system.
Risk assessment is used to determine the level of threat that each hazard poses, in order to establish a mitigation plan for each hazard [17]. There are three main categories in risk assessment [6]:

Intolerable: this describes events that pose a threat to human life and must not lead to an accident. The system must either be designed to mitigate the hazards from occurring or to change the system behavior when the hazards, in this risk category, occur to avoid any accidents.

As low as reasonably practical (ALARP): this describes events that are either less serious than the intolerable ones, or have a much lower probability of occurrence. The

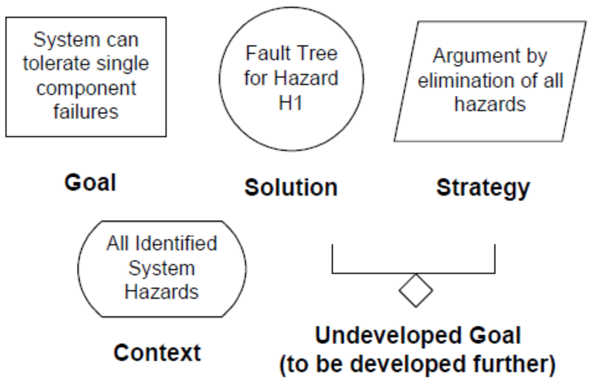

System can tolerate single component failures
Goal
Fault Tree for Hazard H1
Solution
Argument by elimination of all hazards
Strategy
All Identified System Hazards
Context
Undeveloped Goal (to be developed further)

Figure 2.1: The principal elements used in *GSN*. The figure is taken from [14].

mitigation of these types of hazards must be as low as reasonably practical, considering the risk and the costs associated with the mitigation of the hazard.

Acceptable: this describes events which result in minor damages. The system should be designed to reduce the hazards associated with these events, if possible to do so without a significant increase in the cost of developing the system.

2.3.4 Goal Structuring Notation

Many safety arguments are presented in plain text with long descriptions of how the evidence supports the safety claims. In many cases this is manageable for the regulator to read through and understand the connections between the evidence and claims. Unfortunately in other cases the text is written in a complicated way, either making it difficult to clearly understand the arguments or potentially leading to a misunderstanding of the information.
The structured technique Goal Structuring Notation (GSN) attempts to solve these potential misunderstandings. The purpose of using GSN, is that the graphical structure can explicitly show how the goal (claim) is split up into smaller goals, for which specific solutions and strategies are used to fulfill [14].
The three main elements of a safety case, evidence, claims and arguments are all presented in a GSN. The five principal elements used in GSN are shown in figure 2.1. The *goal* element corresponds to the safety claim, the *strategy* element corresponds to the argumentation which explains how a claim may be connected to a sub-claim or evidence. The *solution* element corresponds to the evidence that supports the safety claim. The *context* element describes the context of the goals, for example the context may describe the system scope or operational role [18, 14].

2.3.5 Development Process

In a number of systems, the safety cases are first developed after the physical system is designed and implemented. If issues are found during the development of the safety case, it may require redesign of the system [14]. This may be expensive depending on how far in

the development process the system is. If it is at the end of its life cycle, then it may be cumbersome to re-design the system in order to ensure its safety.
To avoid redesigning the system in the end of its development life cycle, it may in some cases be convenient to follow a phased safety case development process. A phased process divides the safety cases of the system into three main stages (or three main safety cases) which need to be developed simultaneously with the development of the system itself. If the development of a safety case is initiated at the beginning of the development process of the system, then there is a better chance for the design of the system to incorporate necessary safety mechanisms [19, 20].
The three main stages of a phased safety case development process are the Preliminary, Interim and Operational safety cases [19, 21, 20]. The Preliminary safety case is the first stage in the process of creating a phased safety case, followed by the Interim and the Operational safety cases. Figure 2.2 demonstrates the effort that used in each of these stages.

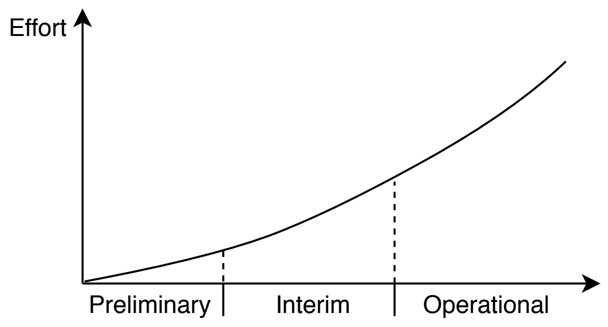


Figure 2.2: An illustration of the order the different safety phases are developed and the effort spent on developing each of the phased safety cases.

A phased safety case is suitable to employ when a novel system is being developed or a novel safety argument is being used. The uncertainty in a novel system occurs when new technologies are used, while the uncertainty in novel safety arguments occurs when they deviate from current standards or if novel forms of evidence are used [20]. The main advantages of using a phased safety case process are [19, 21]:

- Gradually developing the safety case, with possibility to regularly get feedback from the regulator.
- Avoid redesign of the system after implementation in order to comply to safety requirements and standards.

2.4 Modelling and Co-simulation

This section describes the concepts of modelling and co-simulation. Subsection 2.4.1 briefly describes what modelling means, followed by subsection 2.4.2 which describes the concept of co-simulation. Finally, subsection 2.4.3 describes a standard used for interfacing models that can be run in a co-simulation engine.

2.4.1 Modelling

Modelling is the process of creating a model that represents a system. A model is defined by J. Van Amerongen as [22]:

> *"A simplified description of a system, just complex enough to describe or study the phenomena that are relevant for our problem context."*

As the definition describes, a model should be complex enough to represent the most interesting characteristics of a system, but at the same time simple enough to understand and experiment with. The complexity of a system model should be increased iteratively with time, in order to achieve a closer representation of the true system [5, 23].
It is important that the models that are created are valid representations of the system. One model validation technique is simulating the model with defined inputs and testing the true system with the same input values, then comparing the output of the system and the model [5].
Models are used in many contexts, it may be to predict the stock market, or to analyse a system's behaviour under specific conditions. This thesis will focus on modelling a CPS to perform simulation experiments.

2.4.2 Co-simulation

Simulations are valuable with the fact that it is possible to set up the model parameters and simulate different experiments, which may be expensive or impractical to perform on the true system. Simulation is typically used before the complete system has been developed, to utilize the advantages of simulation when designing the system [5].
Many systems, such as CPSs, consist of a physical process and a computational process. The physical processes will typically be modelled as continuous-time models, while the computational processes will typically be modelled as discrete-time models [24]. These models will usually be created using different tools, and in order to simulate these models together in one simulation, it is necessary to perform co-simulations [12]. Figure 2.3 illustrates how a co-simulation can combine models created in different tools and simulate them.
Co-simulation is the process of executing models realized in different tools simultaneously, where the data exchange between these models is handled by a co-simulation engine [24]. The main task of the co-simulation engine is to progress time in the co-simulation and handle the information exchange between the models. Each tool is responsible for progressing its own models in time.
For the co-simulation engine to be able to read and write the values of the signals in each model, these models must implement an interface such as the Functional Mock-Up Interface (FMI) [25].

2.4.3 Functional Mock-up Interface

The FMI is a standard used for interfacing models that are used in co-simulations. A model that implements the FMI is called a Functional Mock-Up Unit (FMU). An FMU can be

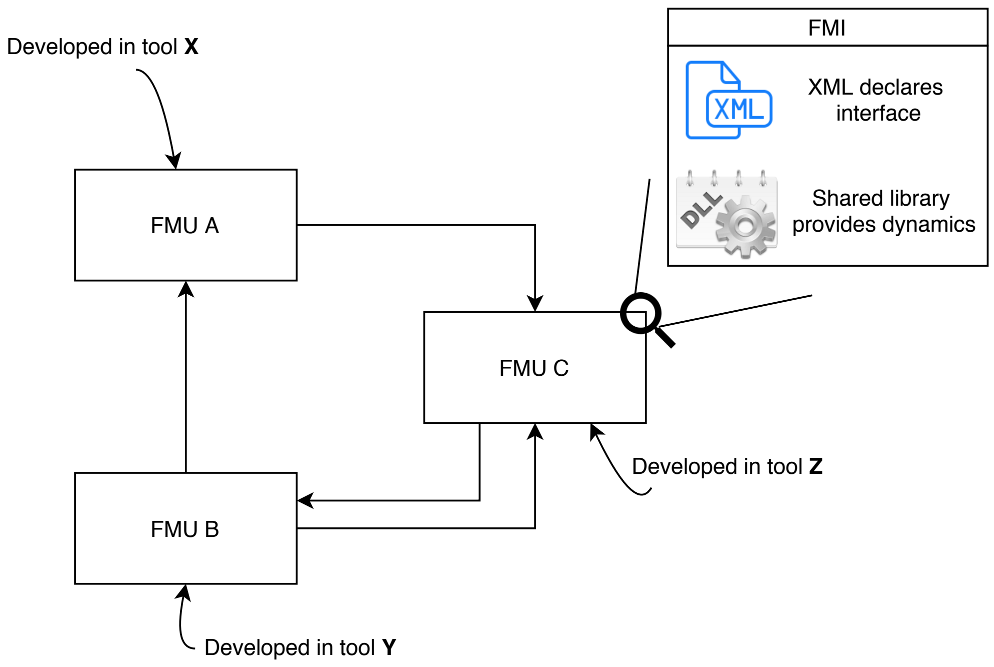


Figure 2.3: An illustration of how co-simulation can combine FMUs developed in three different tools. The figure also illustrates which files an FMU contains. Specifically, an XML file that declares the interface of the FMU, which primarily consists of inputs, outputs, parameters, and which tool the FMU is developed in. Also a shared library containing the functionality of the FMU.

generated by many tools that support this feature. The generated FMU can then be used together in a co-simulation of a complete system or subsystem.
The FMI standard for co-simulation defines a number of functions for creation, initialization and termination of the FMUs. The generated FMU consists of a zip-file including all the necessary information in order to run the FMU in a co-simulation. An example of one of the files that are contained in an FMU, is an XML-file with definition of co-simulation parameters and information about the model [25]. Figure 2.3 illustrates how the FMI standard can be used to encapsulate models, allowing to use these models as black-boxes, with only the interface of these models being visible. This aspect of encapsulating the code of a model, makes it possible for Original Equipment Manufacturers (OEMs) to share models of their equipment, and still protect their intellectual property.

Chapter 3

Safety Case Analysis

3.1 Introduction

Safety-critical systems may damage the environment around them due to component failure(s), erroneous use of the system and various other causes. It is of high importance that a safety-critical system is properly designed and validated to avoid dangerous scenarios.
In order to assure the safety of a system, a safety case will usually be constructed when developing a safety-critical system. A typical safety case consists of multiple documents, attempting to assure the reader that the system is safe to deploy [6].
This chapter presents the relevant parts of the initial safety case analysis created for the Autonomous Combine Harvester (ACH). Section 3.2 gives an overview of a safety case, and describes the steps that will be carried out in this thesis. Afterwards, section 3.3 describes the identified hazards, followed by section 3.4 which describes the risks of each hazard. Finally, section 3.5 describes the focus of the safety case in this thesis project, followed by a graphical representation of the safety case using GSN in section 3.6.

3.2 Safety Case Overview

This section will give an overview of the steps taken for constructing a safety case of the ACH. Subsection 3.2.1 describes the motivation for choosing a phased-safety case development process, and explains why a preliminary safety case will be developed. Subsection 3.2.2 describes the main stages of a preliminary safety case and the stages directly associated with the Goal Structuring Notation (GSN).

3.2.1 Preliminary Safety Case

The ACH is a complex system consisting of hardware, software and mechanical parts, which all need to cooperate for the system to be able to provide its services. Before starting the development of a safety case, it is necessary to consider which safety case development process is appropriate to use.

There exist systems that are in a familiar category of the case study, for example autonomous cars, and combine harvesters that are partially automated, with for example automatic path planning. There is a difference between these systems and a fully autonomous combine harvester. There currently do, to our knowledge, not exist any similar systems with specific and thorough safety standards that can directly be followed. Therefore, the most convenient method to use is the phased safety case development process. The first phase consists of developing the preliminary safety case. The preliminary safety case is developed before a detailed system design, meaning that the safety of the system is defined at a high level [21].

3.2.2 Safety Case Stages

An overview of the main stages of the preliminary safety case are shown in figure 3.1. The main stages have been adopted from combining the traditional steps in a safety case [21] and the steps defined in a traditional GSN [14].
As the figure shows, the GSN can be directly used as part of the safety case. The first phase focuses on gathering information about the system and its hazards. The second phase is associated with defining safety objectives and producing evidence that shows these objectives are satisfied.

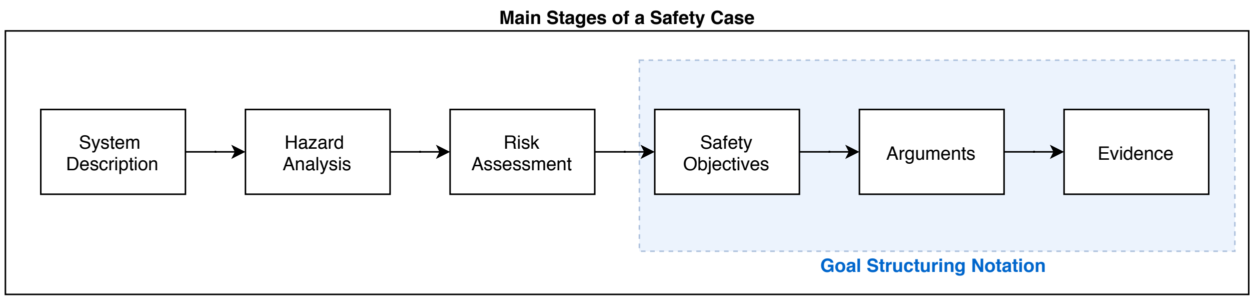


Figure 3.1: The main stages of a safety case, which are relevant for this thesis project.

The following sections describe the most relevant information acquired during the early phases of the safety case. Appendix A contains more details of the findings of this phase. The system was described above in section 1.2.1, and will therefore not be described in this chapter.

3.3 Hazard Analysis

The definitions of a hazard and its analysis were introduced in section 2.3.2. There currently does, to our knowledge, not exist a detailed hazard analysis for autonomous combine harvesters that is publicly available. Therefore, other approaches for defining these hazards have been applied.
In order to define a thorough preliminary hazard analysis, domain expertise is required [17]. To obtain this, discussions with an engineer from the company AGCO were organized. Safety standards related to autonomous road vehicles and combine harvesters were studied, to read

more about the outcome of these standards, see appendix A.3. The outcome of the research lead to defining the following identified hazards:

H1: Collision with object.

H2: Damage to unharvested crops.

H3: Fire ignition.

H4: Damage to harvesting machinery.

H5: Contamination of harvested crops.

In this analysis a top-down approach is used, where a fault tree analysis of each hazard is performed, to determine the events and root causes. A full analysis of each of the hazards listed above can be found in appendix B.

3.4 Risk Assessment

The theory of risk assessment was described above in section 3.4. The risk of each hazard is defined by determining the probability of the occurrence of the hazard, combined with the severity of the accident that will occur due to the hazard [6]. The risk analysis of each identified hazard is shown in table 3.1. The acceptability column in the table denotes how serious the effect of the hazard may be. In this case there are three different levels: intolerable, ALARP and acceptable. These definitions can be used to determine which risks must be prioritized to spend time on reducing.
The risks proposed in this table are the risks of the hazards before defining any risk reduction mechanisms. The descriptions for choosing the specific values for each of the columns in the table can be found in appendix A.4.

Identified Hazard	**Hazard probability**	**Accident severity**	**Estimated risk**	**Acceptability**
H1: Collision with object	Medium	High	High	Intolerable
H2: Damage to unharvested crops	Low	Medium	Medium	ALARP
H3: Fire ignition	Low	High	High	Intolerable
H4: Damage to harvesting machinery	Low	Medium	Medium	ALARP
H5: Contamination of harvested crops	Low	Medium	Medium	ALARP

Table 3.1: Risk assessment of the identified hazards.

It is important to note that the specific risk of a hazard is arbitrary and may vary depending on where in the world, the system will be developed or used. For example, the probability and severity of the combine harvester catching fire is lower in a country like Denmark, compared to a country like Australia. In that case it makes sense to set the estimated risk to high, as the system must operate safely irregardless of which country it is operating in.

3.5 Safety Focus

This section describes the scope of the safety case in this thesis and relates the safety case to co-simulation. Subsection 3.5.1 describes the scope of the safety cases, describing which hazard will be the main focus. Afterwards, subsection 3.5.2 presents an analysis of the chosen hazard. Finally, subsection 3.5.3 describes the challenges of using traditional evidence creation techniques, and motivates the use of co-simulation in the safety case.

3.5.1 Scope

Only a few of the system hazards have been included in this Master thesis, since the focus is mainly on the process. If a full hazard analysis was conducted, there would be a larger amount of hazards, that all would need to be considered. By performing a risk assessment of each identified hazard, it is possible to determine which hazards impose a high risk, and therefore must be further analysed.
The risk assessment in section 3.4 above, shows that two of the hazards have a high risk, and the others have a medium risk. The focus should be on the hazards with a high risk, which are H1 and H3. H1 can be mitigated using software, redundant sensors, and creating algorithms that increase the probability of the system detecting an obstacle and braking before collision. H3 can be mitigated using mechanics isolating the parts of the system that may become too hot, decreasing the probability of a fire occurring. The most relevant of these two hazards to work further on with co-simulation is H1, since it can be mitigated by software algorithms. Therefore, H1 will be the focus of the safety case for this thesis, and will be used to demonstrate the complete process of developing a safety case using co-simulation and Goal Structuring Notation (GSN).

3.5.2 Analysis of Hazard

The following paragraphs will contain relevant parts of the analysis of *H1: Collision with object*.
H1 describes the case where the vehicle collides with an object. Figure 3.2 shows the first part of the fault tree analysis of the hazard. The descriptions of each system state and root cause related to H1 can be found in appendix B.2. The fault tree shows that there are five system states that may lead to the vehicle colliding with an object.
Moving further down the fault tree analysis of this hazard, the "Obstacle not detected by sensor" event occurs due to various other events, such as "Decreased LiDAR performance" or "Camera view obscured". The fault-tree analysis of these two events are illustrated in figures 3.3 and 3.4. As the diagrams illustrate, the events of dense fog or rain may decrease the performance of the sensors, affecting the safety of the system [26, 27, 28, 29].

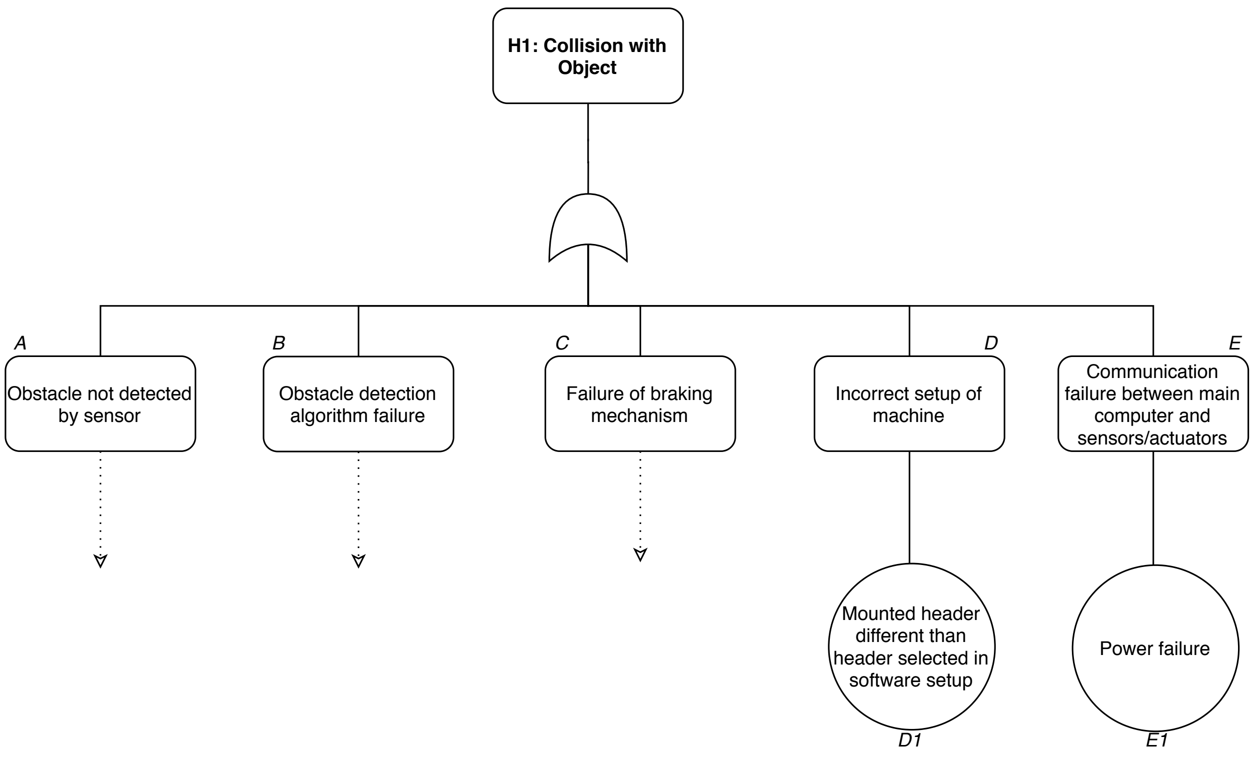

H1: Collision with Object
A
Obstacle not detected by sensor
B
Obstacle detection algorithm failure
C
Failure of braking mechanism
D
Incorrect setup of machine
E
Communication failure between main computer and sensors/actuators
Mounted header different than header selected in software setup
D1
Power failure
E1

Figure 3.2: First part of the fault tree of *H1 - Collision with Object*.

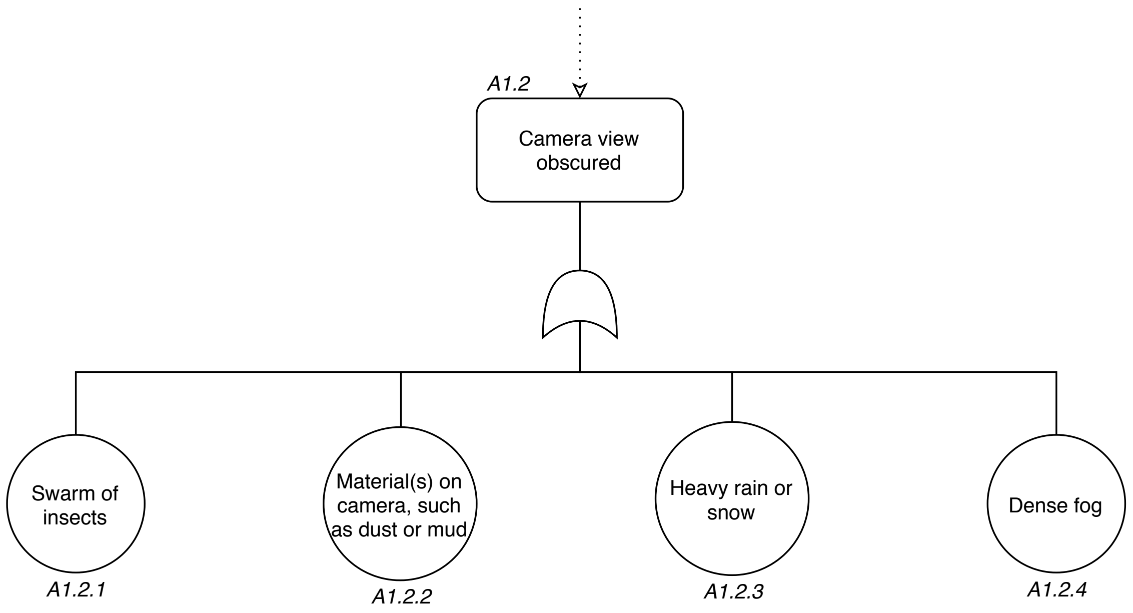


Figure 3.3: This fault tree shows the root causes for *A1.2: Camera view obscured*, and is a continuation of the fault tree in figure B.2.

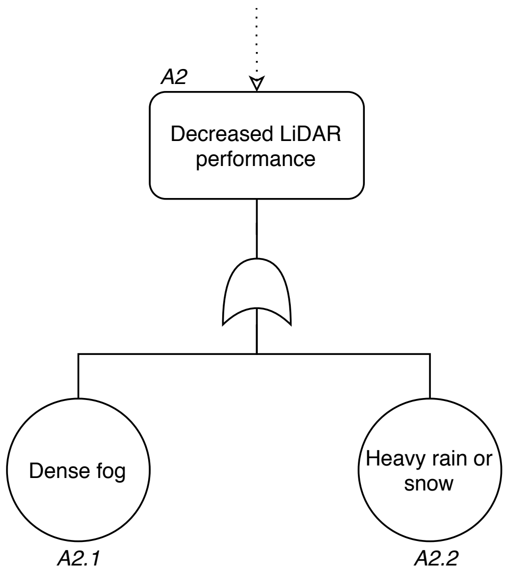


Figure 3.4: This fault tree shows the root causes for *A2: Decreased LiDAR performance*, and is a continuation of the fault tree in figure B.2.

The mitigation of this hazard can be carried out by adding a safety controller which receives object data from sensors, and controls the speed of the system to ensure no collisions occur. Figure 3.5 illustrates how such a subsystem can be created, and how the different components affect each other. Table 3.2 describes each of these components.

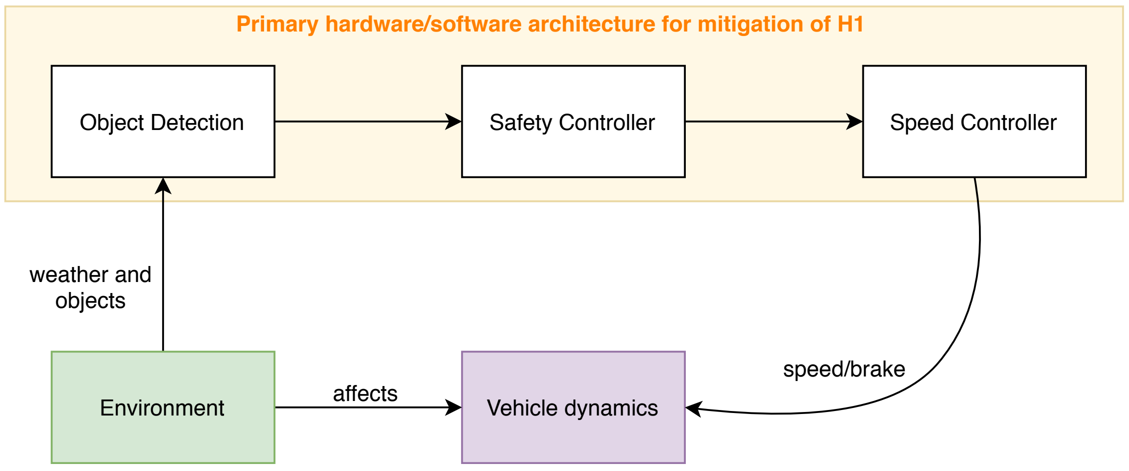


Figure 3.5: An overview of how the hazard, H1, may be mitigated using hardware and software. The components in the yellow box denote the software and hardware related mitigating the hazard. The Environment and Vehicle Dynamics components represent the physical nature of the operating environment and system dynamics.

Figure 3.5 illustrates the complex interactions of a subsystem that can be used to mitigate H1. Notice the interactions of the software/hardware components of the system, with the physical

Subsystem	Description
Object Detection	This subsystem consists of a combination of sensors that detect objects in the vehicle's environment. The weather conditions and other factors can affect the detection range and accuracy of the subsystem.
Safety Controller	This subsystem monitors data from the object detection subsystem and the current braking distance, where it uses the information to ensure the safety of the system by controlling the speed of the vehicle. This can also be seen as a safety controller, that will control the system as soon as it detects a potential unsafe state.
Speed Controller	This subsystem controls the speed and deceleration of the vehicle. It uses the speed of the vehicle to calculate the braking distance, which is communicated to the Supervisory Control Logic subsystem. Once this subsystem receives a brake signal, it will affect the vehicle dynamics and brake the system.
Vehicle Dynamics	This element describes the vehicle dynamics of the system, which consists of the hardware and mechanical components. The vehicle dynamics are affected by the environment it is operating in. For example the amount of moisture in the soil under the vehicle may affect the speed of the vehicle, therefore affecting its dynamics.
Environment	This element characterizes the physical environment the system is operating in. For example, it describes the weather, the obstacles in the vehicle's path, and the soil that the vehicle is driving on.

Table 3.2: Descriptions of the elements illustrated in figure 3.5.

environment. These are complex interactions, which need to be analysed together in order to produce valid evidence. This is where co-simulation can be used.

3.5.3 Motivation for using Co-simulation

This section describes the challenges of producing evidence using traditional methods, in order to demonstrate the mitigation of hazard, H1. The motivation for using co-simulation for producing evidence is also described.
As section 3.5.2 above describes, the mitigation of H1 can be carried out by creating a subsystem with a safety controller, that receives inputs from sensors, and controls the vehicle. Constructing evidence for such a system using traditional techniques is difficult due to a number of aspects, which are described below. Using co-simulation for producing evidence will also be motivated for each of the challenges:

Complex subsystem interactions: The complex interactions of the subsystem are shown in figure 3.5, where it is clear that these interactions are between physical and software components, which may be a demanding process to analyse. It may be difficult or even impossible to mathematically prove this subsystem is safe, due to the interactions with the environment. These subsystems can each be modelled, and the interactions between them can be co-simulated.

Unethical or expensive tests: Constructing evidence for many systems may be carried out by performing tests on the physical system. Some of these tests may be unethical, expensive, or may damage the environment. Examples of such tests are ensuring the system avoids collision when a living being is on its path. Testing such a scenario can advantageously be performed using co-simulation.

3.6 Formalizing the Goals of a Safety Case

The last stages of the safety case focus on defining safety objectives, constructing evidence combined with an argument showing the system is safe. The following subsections will illustrate an example of GSN, then describe how the specific GSN example was built up using the information obtained in the previous phases of the safety case. The principles of GSN were introduced in section 2.3.4.
An overview of the initial GSN constructed for H1 is shown in figure 3.6.

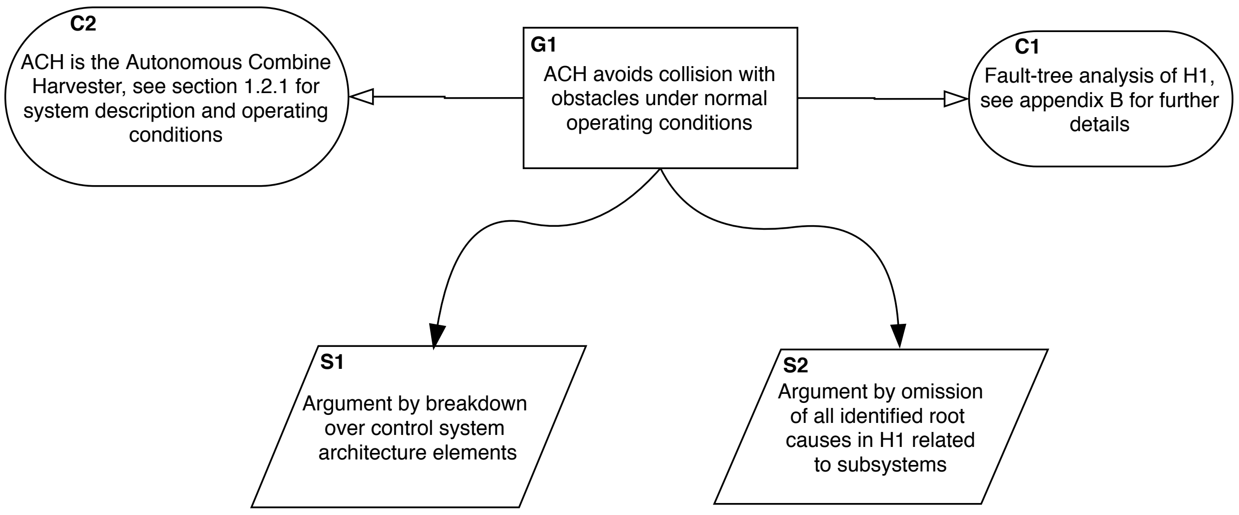

C2
ACH is the Autonomous Combine Harvester, see section 1.2.1 for system description and operating conditions
G1
ACH avoids collision with obstacles under normal operating conditions
C1
Fault-tree analysis of H1, see appendix B for further details
S1
Argument by breakdown over control system architecture elements
S2
Argument by omission of all identified root causes in H1 related to subsystems

Figure 3.6: The initial GSN diagram with the top-level claim "The Autonomous Combine Harvester avoids collision with obstacles under normal operating conditions".

The GSN containing S1, can be seen in figure 3.7, and was adapted from the safety case pattern *Control System Architecture Breakdown Argument* [30]. It demonstrates the safety goals of each of the components of the subsystem for mitigating H1, illustrated in figure 3.5.

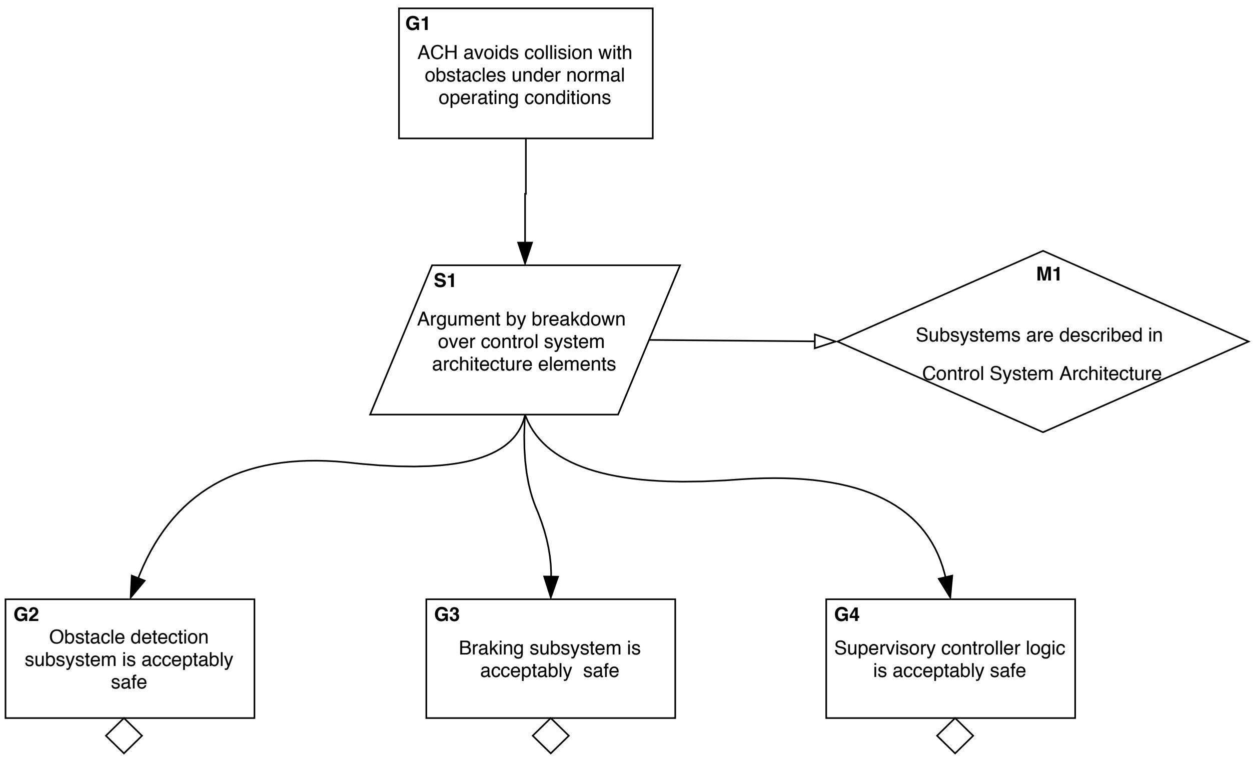

G1
ACH avoids collision with obstacles under normal operating conditions
S1
Argument by breakdown over control system architecture elements
M1
Subsystems are described in Control System Architecture
G2
Obstacle detection subsystem is acceptably safe
G3
Braking subsystem is acceptably safe
G4
Supervisory controller logic is acceptably safe

Figure 3.7: The GSN diagram with the top-level claim "The Autonomous Combine Harvester avoids collision with obstacles under normal operating conditions", with details of S1 continuing the diagram in figure 3.6. This diagram contains the most relevant parts of the GSN, the full diagram can be seen in figure A.2.

The GSN containing S2 is shown in figure 3.8. The GSN is at a high abstraction level, which means there are some details that are not included, for example only three of the root causes are included in the diagram, and only one hazardous event, G5, is included. The reason for this is that the focus of the thesis will only be on these included sub-goals. The evidence in the diagram is dashed since it is not available yet, and is what will be developed in this thesis.
Table 3.3 describes the most relevant elements of the GSNs presented in figures 3.6, 3.7 and 3.8.

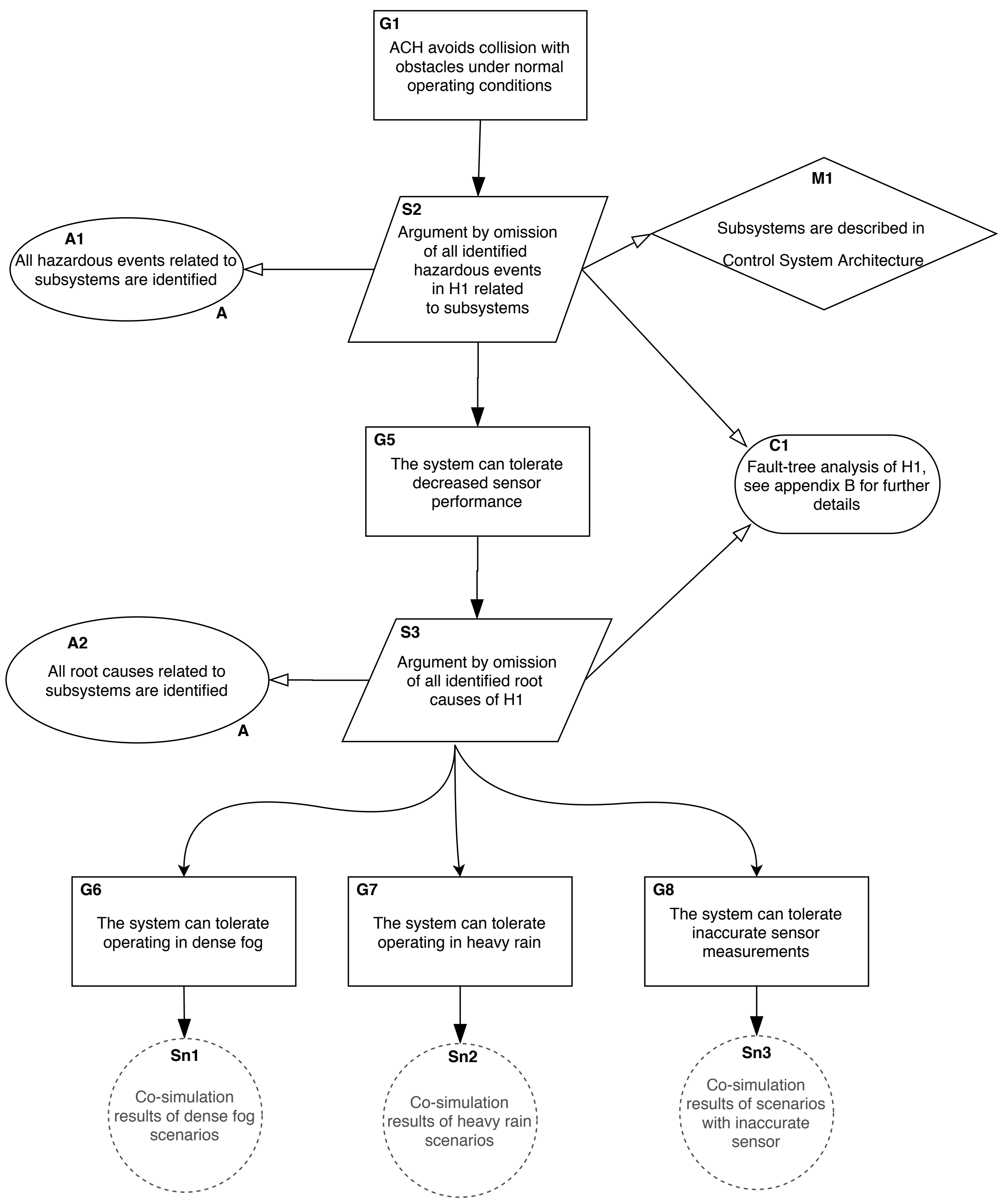

G1
ACH avoids collision with obstacles under normal operating conditions
S2
Argument by omission of all identified hazardous events in H1 related to subsystems
M1
Subsystems are described in Control System Architecture
A1
All hazardous events related to subsystems are identified
A
G5
The system can tolerate decreased sensor performance
C1
Fault-tree analysis of H1, see appendix B for further details
S3
Argument by omission of all identified root causes of H1
A2
All root causes related to subsystems are identified
A
G6
The system can tolerate operating in dense fog
G7
The system can tolerate operating in heavy rain
G8
The system can tolerate inaccurate sensor measurements
Sn1
Co-simulation results of dense fog scenarios
Sn2
Co-simulation results of heavy rain scenarios
Sn3
Co-simulation results of scenarios with inaccurate sensor

Figure 3.8: The GSN diagram with the top-level claim "The Autonomous Combine Harvester avoids collision with obstacles under normal operating conditions", with details of S2 continuing the diagram in figure 3.6. This diagram contains the most relevant parts of the GSN, the full diagram can be seen in figure A.3.

ID	Description
G1	The top-level safety goal of the GSN was derived from the hazard analysis of H1. The context C1 describes the hazard analysis of H1. The context C2 describes the normal operating environments of the system.
S1	One of the strategies adopted to fulfill the claim. The argument will be constructed by breaking down the safety goal into safety goals of subsystems. This will be done using the model of the individual elements shown in M1.
S2	The other strategy used is to omit the hazardous events related to H1, identified in the hazard analysis in appendix B. Another strategy to fulfill the claim is to mitigate the root causes that lead to the hazard association with collision with obstacles.
M1	A model of the control architecture system, showing the relationship between the subsystems supervisory control logic, object detection, and braking subsystem. The model can be seen in figure 3.5.

Table 3.3: Descriptions of the most relevant elements in the Goal Structuring Notation in figures 3.6, 3.7 and 3.8. The table containing the full descriptions can be found in appendix A.6.

3.6.1 Safety Goals

The safety goals of the GSN can be directly derived from the fault-tree analysis of a hazard, as shown in figure 3.9. The hazard can be directly translated to the top-level safety goal of the GSN, and the events and root causes of the hazard can be translated to sub-goals of the GSN. The GSN shown in figure 3.9 is just an abstract overview, used to describe how the fault-tree analysis can be related to the safety goals of a GSN. The detailed GSN can be seen in figure 3.7.
It is important to notice the relation between the hazard and the events that can lead to such a hazard. In this case the relation is denoted by an OR gate, meaning that the situation in which the events occur separately, and the situation in which the events occur simultaneously may both lead to the hazard. In other situations, it may be required to reformulate the top-level goal or use GSN *assumptions*.

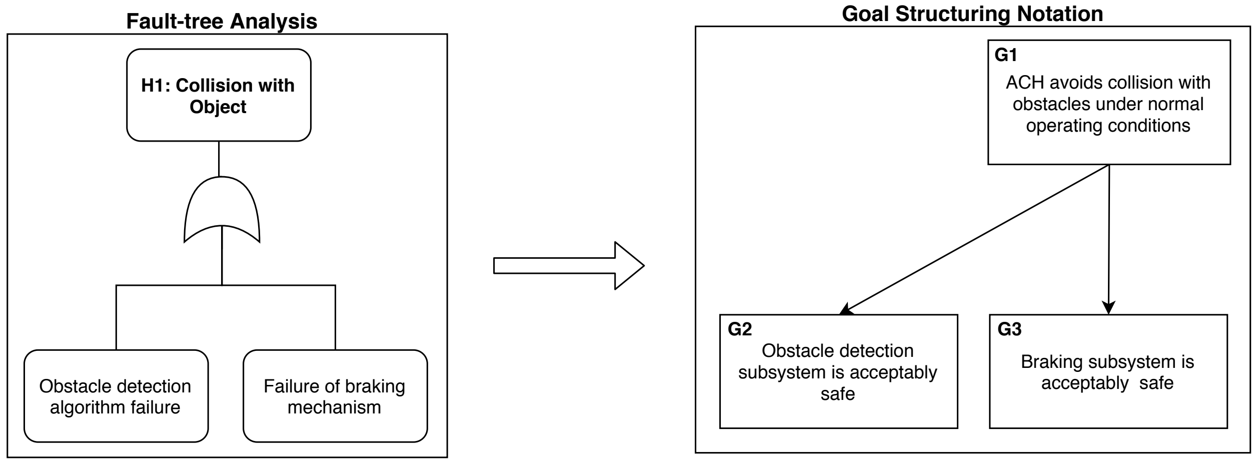

Fault-tree Analysis
H1: Collision with Object
Obstacle detection algorithm failure
Failure of braking mechanism
Goal Structuring Notation
G1
ACH avoids collision with obstacles under normal operating conditions
G2
Obstacle detection subsystem is acceptably safe
G3
Braking subsystem is acceptably safe

Figure 3.9: An example showing how the fault-tree analysis of a hazard can directly be translated to a GSN. This example shows how an initial GSN for H1 can be constructed from the fault-tree analysis.

It is important to note that the goals in a GSN are not always constructed directly from the fault-tree analysis of a hazard, but can also be constructed from other contexts, such as safety requirements, see appendix A.5 for the ACH safety requirements.

3.6.2 Context

The context in a GSN can be information retrieved from different types of sources, such as safety standards, system description, hazard analysis or an information artefact in the form of a model [18]. An example of a context in the initial GSN for the system, is shown in figure 3.6. *C1* is the context which refers to the fault-tree analysis that was performed for H1. The top-level goal *G1* arises from this context.
In another example the context referred to is a model, as illustrated in figure 3.7. The element M1 is the model of the subsystems presenting an overview of the architecture in figure 3.5.

3.6.3 Evidence

In order to complete the safety case, each of the sub-goals in figures 3.7 and 3.8 need to be further developed until evidence can be constructed to support the claims. Evidence for a safety case can be produced in various ways, e.g. system tests or mathematical analysis. In a preliminary safety case, the system design is still in the initial phases, and therefore no physical system exists. In this case it is difficult to perform system tests, and therefore other methods need to be adopted, such as co-simulation.
The safety goals and evidence that will be focused on in this thesis are G6, G7, G8 and Sn1, Sn2 and Sn3 illustrated in figure 3.8. Relevant co-simulation scenarios must be constructed and co-simulated in order to obtain the evidence supporting the safety goals.

Chapter 4

Producing Evidence using Co-simulation

4.1 Introduction

Designing a safety-critical system is a complex process [31]. One of the difficult aspects of this is to understand how each of the individual parts of a system affect each other [32]. This is especially true when designing functional safety in a system, where it is crucial to understand the collaboration of the different modules, to ensure a *robust design*$^{\tau}$ [33]. A suitable method to achieve this understanding is performing a co-simulation of the system. This requires testing the model of the system in chosen relevant scenarios.
Relevant scenarios to co-simulate will be defined, followed by co-simulation of the functional safety of the Autonomous Combine Harvester (ACH).
Section 4.2 describes how the scenarios related to the evidence can be defined and modelled. Afterwards, section 4.3 describes the architecture of the system model, followed by section 4.4 which describes the main characteristics of the designed models in this thesis project, and section 4.5 which describes the functional safety realization. Finally, section 4.6 describes the results obtained from the co-simulations, followed by section 4.7 which explains how the co-simulation results are used as evidence in the safety case.

4.2 Coupling Safety Case to Scenarios

This section describes how the necessary scenarios can be created, in order to be able to produce co-simulation results that can be used as evidence. Figure 4.1 illustrates how the models to be created are related to the evidence. The figure illustrates how the scenarios define the model parameters, and how the co-simulation of the model with specific parameters is used as evidence in the safety case. In order to be able to produce the required evidence, scenarios must be defined, followed by modelling the system and environment. It is important to note that the system interaction with the environment must be part of these scenarios, since the safety goals related to the evidence are associated with the environment of the system.
Before modelling and co-simulating the system, a few preparation steps need to be taken:

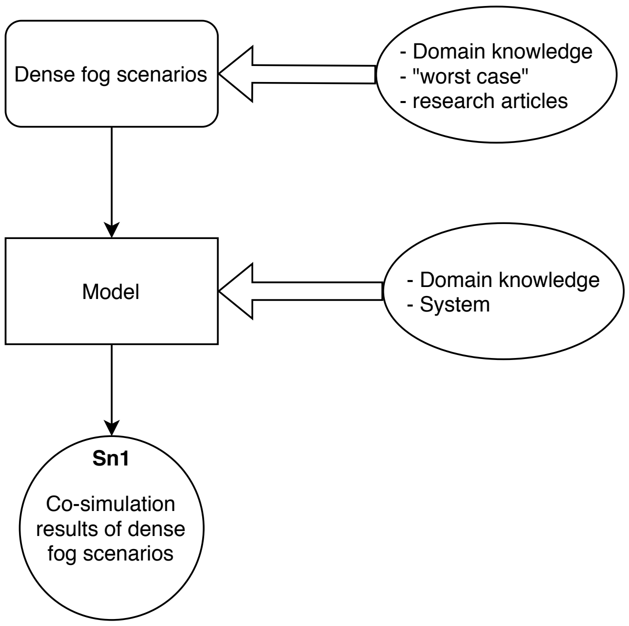


Figure 4.1: An illustration of how the scenarios related to specific evidence are created, and how these scenarios define the co-simulation parameters of the models used. The models are created using domain knowledge about the system, or by reusing other models. The co-simulation results of the model with specific parameters, can then be used to prove that the system is safe, and therefore used as evidence in the safety case. The scenarios are defined using domain knowledge or researched articles. The scenarios are usually defined with values of worst case scenarios.

(a) Contextualize the scenario settings, i.e. create the connection between evidence from a GSN to an environmental setting.

(b) Define how the scenarios can be modelled.

Each of the steps listed above will be described in this section, and performed for the ACH.

4.2.1 Contextualize Scenario Settings

To be able to produce evidence for the safety case, it is necessary to relate the evidence to scenario settings. These settings are then modelled, and co-simulated to prove the safety of the system. This section focuses on defining environmental settings that relate to the evidence in the GSN. Such settings are usually defined by creating a scenario where the system may pose a threat to its environment.
Scenarios of the system operating in dense fog, heavy rain and with inaccurate sensors must be defined. The system must operate safely in defined scenarios related to the evidence, as the scenarios illustrated in figure 4.2, where animals are included in the environment, to be able to illustrate the system safety in such cases.
As figure 4.2 demonstrates, the system may drive into obstacles if they are occluded by water droplets from fog or rain. The vehicle must perform a safety stop to avoid collision with any

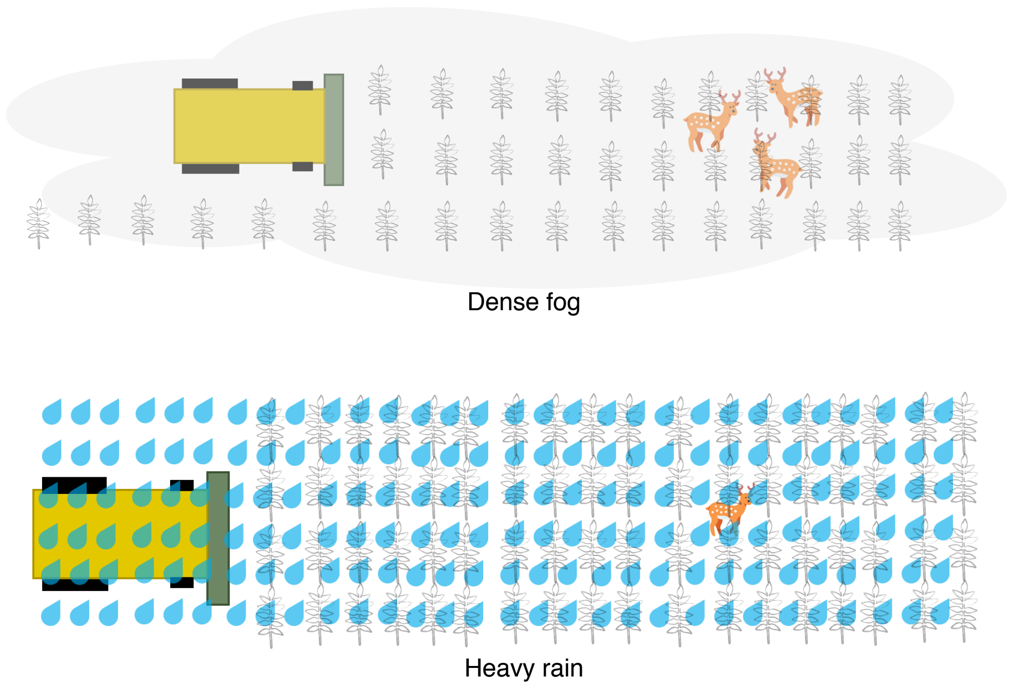


Figure 4.2: An illustration of the scenarios that can be constructed from the evidence in the GSN. The top figure illustrates a scenario where the system is operating in dense fog. The bottom figure illustrates a scenario where the system is operating in heavy rain.

object, irregardless of the environmental conditions. To test the system in such scenarios, the vehicle must be tested with different speeds, in order to ensure the vehicle has enough time to perform a safety stop before a collision occurs. Combine harvesters will typically drive around 2 m/s on a crop field [10], and therefore this speed must be tested. The chosen speeds are therefore, 1, 2 and 3 m/s. When starting the system, the initial distance to any obstacles must be greater than 1 meter, to allow the system to actually start driving, and gain a constant speed. Of course in the further development of this system, there should be a safety monitor to ensure the system will not start if there are obstacles within a defined safety zone. The initially defined scenarios that must be constructed and co-simulated are described in table 4.1.
The next step is to determine how these scenarios can be modelled.

Scenario	Vehicle speed	Initial distance to obstacle
Dense fog	{1,2,3} [m/s]	>1 meter
Heavy rain	{1,2,3} [m/s]	>1 meter
Inaccurate sensor	{1,2,3} [m/s]	>1 meter

Table 4.1: Initially defined scenarios that must be run in a co-simulation.

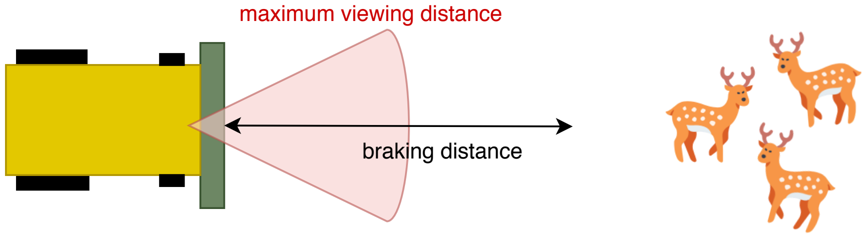


Figure 4.3: A visualization of an example where the braking distance of the vehicle, due to the speed, is greater than the maximum viewing distance of the sensor. This scenario shows that the vehicle will not necessarily have enough time to brake in order to avoid collision with obstacles, leading to a possible unsafe situation.

4.2.2 Defining Scenario and Model Parameters

The effect of each scenario must be understood in order to be able to properly model the system. For example, the effect of dense fog on the system was determined by studying relevant articles, where experiments were performed on sensors, such as LiDARs and cameras. The results of the experiments performed in these articles were used to define the scenario parameters.
It is necessary to scope the specific parameter values to use in the co-simulation scenarios, since it is not possible to test the system with every possible parameter value. Therefore, the most relevant values must be tested. These values can be derived from articles or domain knowledge from experts, which may already be identified in the hazard analysis of the system. The two weather conditions of dense fog and heavy rain were analysed together, where the articles [26, 27, 28, 29] were studied. The relevant outcome from studying these articles is that the maximum range of these sensors decreases significantly when operating in dense fog or heavy rain. Looking at the safety aspects of this, figure 4.3 illustrates a case where the decreased maximum range may lead to the system colliding with the obstacles on its path. This shows the relevance of modelling and co-simulating these scenarios, which are related to evidence *Sn1* and *Sn2* in figure 3.8. The specific values of the maximum viewing distance were determined using the experiment results of the articles [26, 27, 28, 29], and are presented in table 4.4 and figure 4.4.
Inaccurate sensor measurements may also affect the performance of the system. A LiDAR may have inaccurate measurements of close obstacles, where in some cases it may not measure them, leading to a minimum range of the LiDAR [34]. An example of the minimum distance of a LiDAR is 0.5 meters [35]. This information is related to evidence *Sn3* in figure 3.8, where the LiDAR may give inaccurate measurements or not measure obstacles at all. These specific values can be used to model the related scenarios.
Figure 4.4 illustrates the process of studying articles that help in defining the specific parameters for modelling the scenarios. It is most reasonable to co-simulate worst case scenarios from the data obtained from studying the articles. In some cases it is a good idea to co-simulate scenarios that are slightly worse than the worst case scenario, in order to account for any neglected factors that may affect the system. As [26] describes, the maximum viewing distance in dense fog can go down to approximately 5 meters, in experiments performed

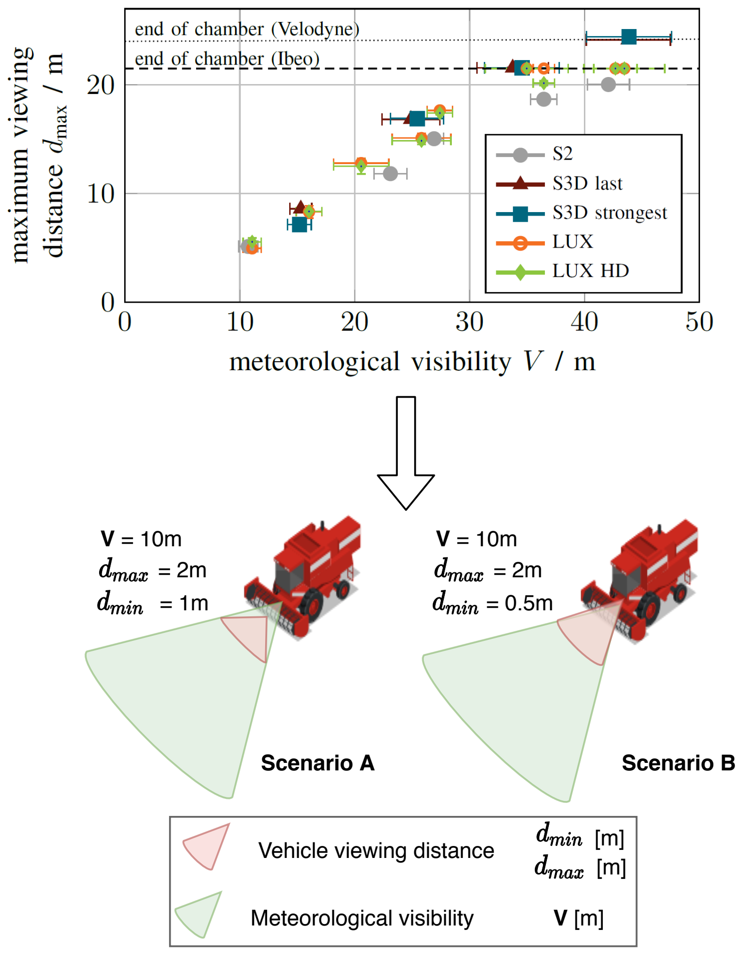


Figure 4.4: An example of how the root cause of dense fog affects the LiDAR, can be used to describe co-simulation scenarios. The graph on the top is taken from [26]. The graph shows measurements from experiments with objects of a reflectivity of 90%. The distance of the meteorological fog is varied in the experiments, and the maximum viewing distance of different types of LiDARs is registered. The types of LiDARs used in the experiment are described in the legend of the graph. The results of the experiments can be used to define the viewing distance of the sensors mounted on the ACH.

using reflective targets with a 90% reflectivity [26]. Not all objects on a crop field will have such a high percentage of reflectivity, and therefore it is crucial to choose scenarios with a lower maximum viewing distance than 5 meters. Therefore, when modelling dense fog or heavy rain, the maximum viewing distance was set to 2 meters, which is much less than the 5 meters described in the articles, with objects of 90% reflectivity. The same matter accounts to modelling inaccurate sensors, where the minimum range was defined as 0.5 meters, it was modelled to be both 0.5 and 1 meters, to test worst case scenarios.
Before designing a model, the purpose and abstraction level should be determined. In this case the purpose is to model the relevant scenarios in order to use the co-simulation results of the model as evidence. The safety case that is being developed is a preliminary safety case,

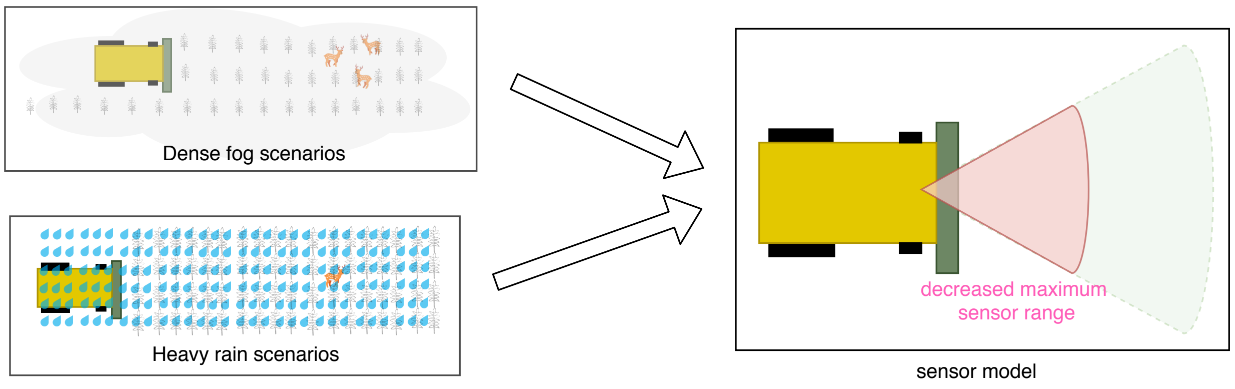


Figure 4.5: An illustration of how the scenarios related to dense fog and heavy rain may be modelled simultaneously, since they affect the system in similar ways. The figure demonstrates that the scenarios where dense fog or heavy rain occur affect the maximum viewing range of the sensor.

meaning the initial design of the system is not yet developed. Therefore, the abstraction level of these models should be at a high level.
As described above, the effect of heavy rain and dense fog on the system, is a decrease in the maximum range. These two scenarios can be modelled simultaneously by decreasing the range of the sensors in the vehicle. The range of this subsystem can be adjusted according to which scenario is being modelled. Figure 4.5 illustrates that the scenarios associated with dense fog and heavy rain can be modelled in one model with the same parameters, since they may affect the system in similar ways, as described above.

4.3 Designing Model Architecture

After the relevant scenarios are determined, in section 4.2, the next step is to decide which components must be modelled at the chosen abstraction level, in order to be able to co-simulate the scenarios. Section 4.3.1 describes how to determine which components must be modelled, and section 4.3.2 describes the designed architecture of the system model.

4.3.1 Determining required Models

The chosen scenarios can be modelled at different levels of abstraction, depending on the purpose of the co-simulation. For example, it is possible to only model the supervisory controller, object detection subsystem and braking mechanism, as shown in figure 3.5, where the focus is on the logic of the supervisory controller.
The disadvantage of modelling the system this way, is that in principle it does not prove that the system will be able to brake in time. It will essentially just be a co-simulation testing the supervisory controller logic. In order to construct satisfactory evidence, it is necessary to model the following components (derived from the safety analysis performed in chapter 3):

- The vehicle dynamics
- The vehicle controller (including the braking mechanism)
- The object detection subsystem
- The supervisory controller

In order to make the co-simulation of scenarios easier, the following components may be modelled:

- Component(s) to model the environment of the scenarios
- A component to visualize co-simulation

4.3.2 Architecture

An overview of the co-simulation setup is shown in figure 4.6. Tables 4.2 and 4.3 describe the connections between the FMUs and the tools which the FMUs were developed in respectively. The tools in which the FMUs are developed are described in appendix C.1. It is important to note that the vehicle and controller FMUs were reused from an example project from *INTO-CPS*. The other FMUs were implemented in this thesis.

Signal	**Description**
r_xy	The x and y position of the vehicle.
ctrl_r_sig	The control signals from the Controller to the Vehicle. These signals are forwarded by the Supervisory Controller to the Vehicle.
obs_data	The x and y positions of the center of the obstacles existing in the environment. These are the true obstacles. The signal also contains the number of obstacles.
sen_data	The x and y positions of the obstacles currently detected by the sensor. The signal also contains the number of obstacles.
r_mes	Measurements of the rotational and angular velocity of the vehicle.
theta	The rotational orientation of the vehicle.

Table 4.2: Descriptions of the signals denoted in figure 4.6.

FMU	**Tool**
Controller	Overture
Vehicle	20-sim
Environment	PyFMU
Sensor	PyFMU
Supervisory Controller	PyFMU
Monitor	PyFMU

Table 4.3: The programs that each FMU was developed in and exported from.

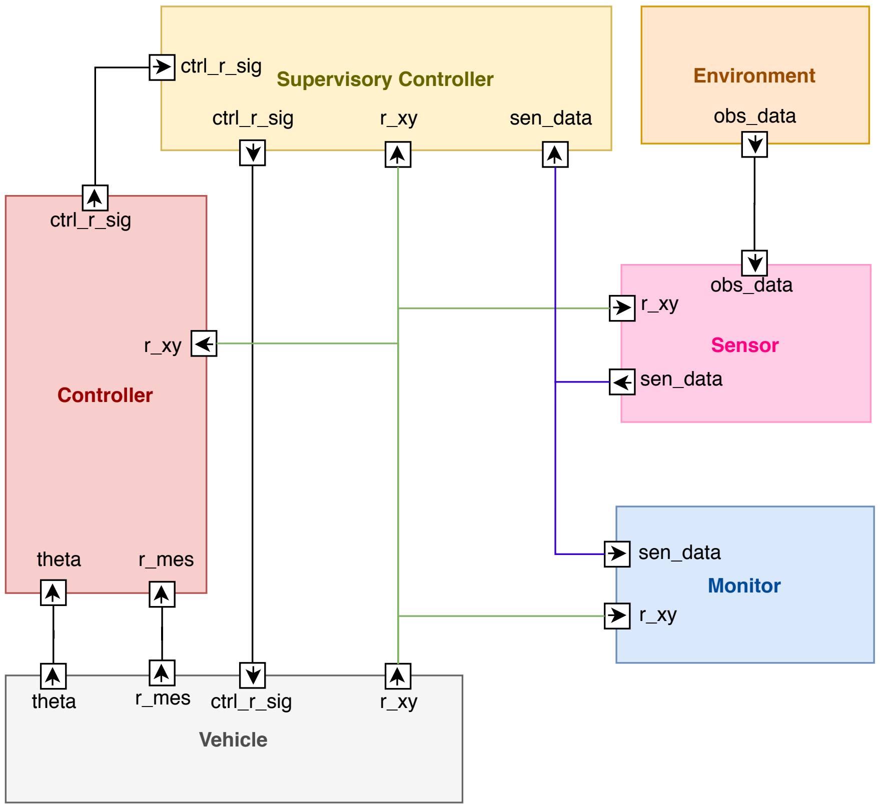


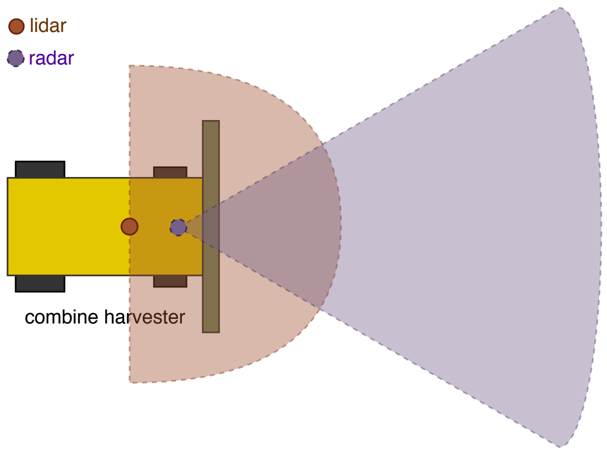
Figure 4.6: An overview of the FMUs, and connections between these FMUs, used to model the system with hazards. The connections denoted in the figure summarize multiple connections between the FMUs. The connections between the FMUs are described in table 4.2.

4.4 Main Realization Characteristics

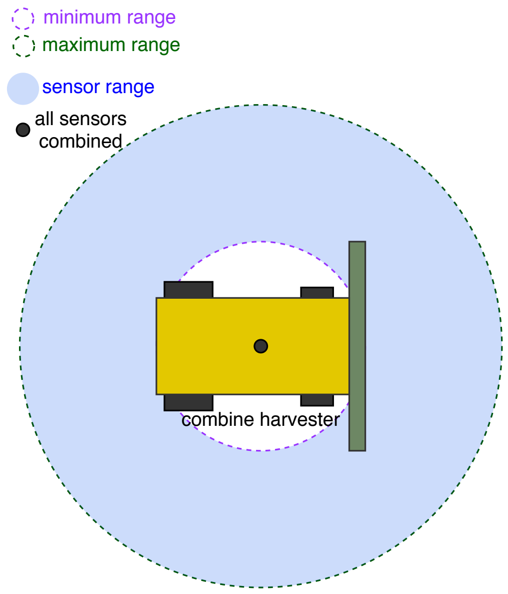
The main characteristics of the FMUs implemented in this project are described in this section. Subsection 4.4.1 describes how the sensors are generalized into one sensor model. Afterwards, subsection 4.4.2 describes how the range of the sensors can be set as a parameter. Finally, subsection 4.4.3 describes how the environment can be adjusted to co-simulate different environments.

4.4.1 Generalizing the Sensors

A model is typically an abstraction of a physical system, where the relevant aspects of the physical system are modelled, and other irrelevant aspects are abstracted away. The ACH has at least the three sensors: camera, lidar and radar. When modelling the sensors, the exact type of sensor and how it perceives data and processes it, is abstracted away leading to a generalized model of such sensors. Figure 4.7 shows how the true sensors on such a system are generalized in the sensor model.



(a) A mounted lidar and radar on a combine harvester, showing the range of each of the sensors. This is just an example, showing two mounted sensors, but in reality there will be several types of sensors mounted in different places on such a system.



(b) An example illustrating how the sensors are modelled, by generalizing a combination of all of the sensors on a combine harvester. The range of the generalized sensors together is modelled as a circular area around the vehicle, with a minimum and maximum range.

Figure 4.7: Generalization of sensors on the Autonomous Combine Harvester.

4.4.2 Sensor Visibility Range

The model of the sensors contains two parameters, one for setting the minimum range and one for setting the maximum range of the sensor. These parameters could be adjusted in the co-simulation setup in order to easily simulate various scenarios. Figure 4.8 shows an example of a live plot, showing the range of the sensor visibility. By adjusting the sensor range it is possible to observe the effect on the overall perception module of the system.

4.4.3 Adjusting the Environment

Performing co-simulations is associated with running a number of different scenarios. Each scenario represents a situation that may occur in reality when the ACH is operating in a crop field.
To ensure a smooth and easy process of adjusting scenarios, an *Environment* FMU was created, which allows uploading an image with obstacles on it. The *Environment* FMU processes the image and outputs the locations of the obstacles on the image. An example of one of the scenarios uploaded to the environment is shown in figure 4.9.

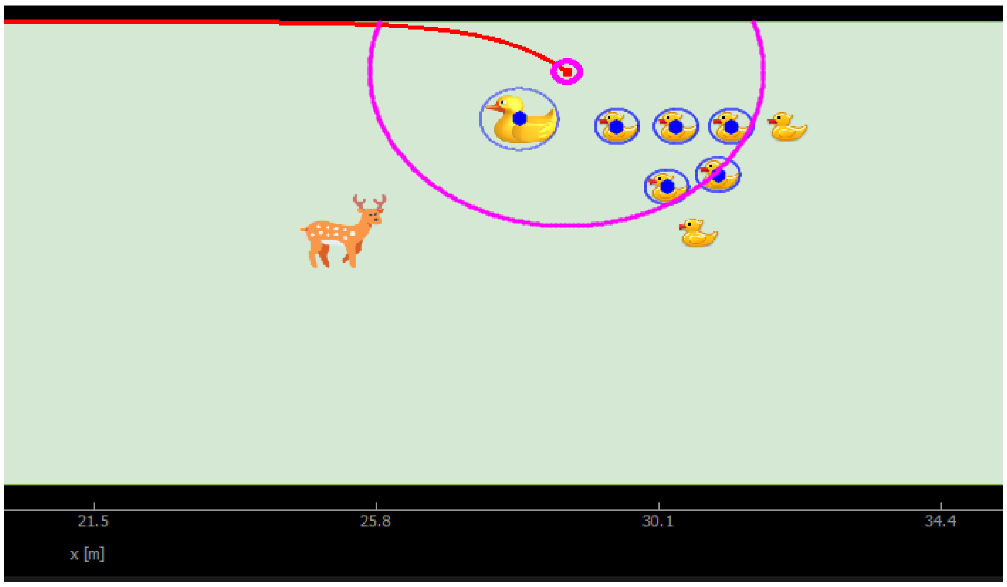


Figure 4.8: An example of a co-simulation run of the system with visualization of the minimum and maximum range of the sensor, and the currently detected objects. The red lines displays the route the vehicle has driven so far. The two pink circles represent the sensor minimum and maximum range. The blue circles around the ducks represent the size of the objects that the sensor has detected. The minimum and maximum ranges are set to 0.2m and 3m respectively in this co-simulation.

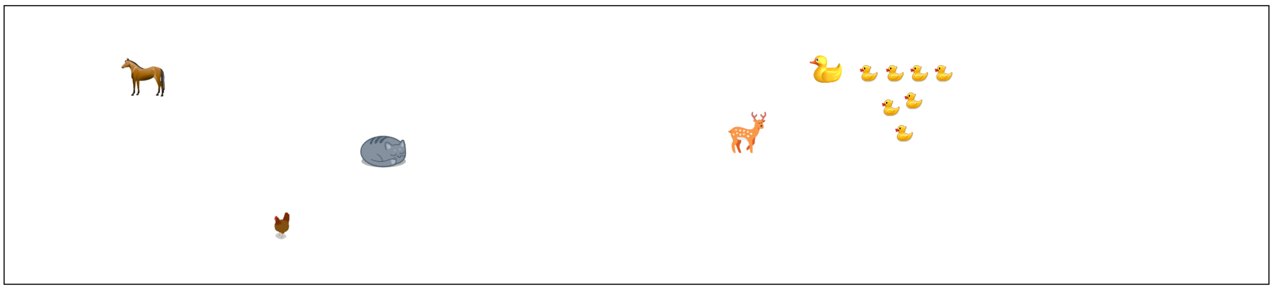


Figure 4.9: An example of a map used for co-simulating a scenario. Each animal located on the map is an obstacle.

4.5 Functional Safety

This section describes how the functional safety is designed and realized in the models.

4.5.1 Design

This step incorporates researching and implementing different designs of functional safety in the model in order to mitigate the hazardous situations that may occur due to bad weather. There are different designs that can deal with the mitigation of such hazards. Some of the designs may be to improve the object detection algorithm, other designs may be to slow down the vehicle speed to decrease the braking distance. Examples of possible designs are:

a) Fused data from the camera and the LiDAR can be used to increase the maximum viewing distance of the object detection algorithm [36].

b) Using predicted braking distance to control the vehicle's maximum speed [37, 38].

c) Using predefined constant braking distance to control the vehicle's maximum speed [38].

Design *b* is chosen to be realized in this thesis, since it can be modelled in the right abstraction level, and the sensor range and vehicle speed can be used to determine the braking distance, and adjusting the vehicle speed. The design will be realised as illustrated in figure 3.5.

4.5.2 Realization

This section describes the purpose of the speed control mechanism, the modelling assumptions and how the mechanism is realized.

4.5.2.1 Purpose

The purpose of this speed control mechanism is to mitigate the hazardous situation, where the vehicle's braking distance is greater than the sensor's maximum viewing distance, as illustrated above in figure 4.3. The mechanism is designed to register the maximum range of the sensor, which is used as the target braking distance, where the vehicle speed is then adjusted to satisfy the target braking distance.

4.5.2.2 Modelling Assumptions

To model the speed control mechanism, some assumptions were made, in order to adjust the realization to a high abstraction level. The main assumptions are listed below:

- The braking distance is the same when the vehicle is turning as when it is driving straight.
- The friction between the wheels and the ground is constant, meaning that the model does not take into account if the soil is wet or dry.

4.5.2.3 Realization in the Supervisory Controller

A speed control mechanism was realized in the Supervisory Controller, which performs the following main steps:

1. Calculate braking distance from current vehicle speed.
2. Compare braking distance with maximum sensor range.
3. If braking distance is larger than sensor range, calculate and set new vehicle speed.

4. Calculate distance between vehicle and obstacles detected by sensor. If the obstacle is within the calculated braking distance, perform a safety stop, by setting the vehicle speed to 0 m/s.

The estimator of the braking distance is calculated using the following simplified formula [39]:

$$braking\_distance = \frac{speed^2}{2 * \mu * g} \tag{4.1}$$

Where the distance is measured in meters, *speed* represents the speed in $\frac{m}{s}$, μ represents the coefficient of friction, and g represents the gravity of earth in $\frac{m}{s^2}$. A braking safety margin was added to the braking distance estimator, since it is an approximation of the real vehicle speed. In order to create an accurate prediction the estimator must include other factors of the vehicle dynamics [37], which in this case are abstracted away.
In step 1 the braking distance is calculated using equation 4.1. Step 2 compares if the calculated braking distance is larger than the sensor range. If the braking distance is smaller or equal to the sensor range, then no changes are made. In step 3 the braking distance is larger than the sensor range, therefore safety actions are required. A new vehicle speed is calculated, which will ensure the braking distance of the vehicle decreases to be the same or less than the sensor range. The new vehicle speed is calculated using equation 4.1. Step 4 covers the scenarios where an obstacle is within the braking distance of the system, where the supervisory controller performs a safety stop, to avoid colliding with the obstacle.
The algorithm below illustrates how the speed control mechanism is realized. Line 8 shows the equation used for calculating the new vehicle speed, by using the distance of the maximum sensor range minus the braking safety margin.
The value of the safety margin added to the estimator was determined by performing multiple tests with different values of the parameter, where the best value was found to be 0.5 meters. This value allowed the vehicle to brake in time without braking too far away from the obstacles. These tests to find the optimal value of the parameter may have also been done using the Design Space Exploration feature in the INTO-CPS tool-chain, instead of manually running multiple co-simulations.

```python
1      if sensor_max_range < speed_based_dist:
2          g = 9.8 # gravity of Earth in [m/s^2]
3          a = 2 * mu * g
4          d = sensor_max_range - self.braking_safety_margin
5          if d <= 0:
6              v_cal = self.min_speed
7          else:
8              v_cal = np.sqrt(d * a)
9          cal_speed = v_cal
10     return cal_speed # calculated speed
```

Listing 4.1: Realization of the speed control mechanism using Python.

The test results of this model are described in appendix C.2.6.1, where the results of testing different safety margin values are described.

4.6 Co-simulation Scenario Results

This section presents the co-simulated scenarios, the results and suggested safety improvements. Subsection 4.6.1 gives an overview of the results, where subsection 4.6.2 performs an analysis of the results. Finally, subsection 4.6.3 briefly describes different techniques that can be used to improve the safety of the system.

4.6.1 Co-simulation Results

The final co-simulation step is to run each of the defined scenarios, in order to determine if the current safety mechanisms satisfy the system safety goals. The scenarios were defined by researching the articles related to the evidence required, as described in section 4.2. The specific scenarios that were co-simulated are presented in table 4.4. The table outlines which environment, sensor range, and initial vehicle speed that was set, and the last column expresses the results of co-simulating each of these scenarios. The maps of the environment can be found in appendix C.4.
The scenarios where the sensor range is set to [0.5 ; 2] demonstrates bad weather, i.e. heavy rain or dense fog, which decreases the sensor range. The scenarios where the sensor range is set to [1 ; 2] demonstrates inaccurate measurements leading to a worse minimum measurement range. The co-simulation was performed with a step-size of 0.05s.

Scenario	**Environment**	**Sensor range** $[m]$	**Initial vehicle speed** $[\frac{m}{s}]$	**Safety Stop**
1	map1	[0.5 ; 2]	1	Success
2	map1	[0.5 ; 2]	2	Success
3	map1	[0.5 ; 2]	3	Success
4	map1	[1 ; 2]	1	Failure
5	map1	[1 ; 2]	2	Failure
6	map1	[1 ; 2]	3	Success
7	map2	[0.5 ; 2]	1	Success
8	map2	[0.5 ; 2]	2	Success
9	map2	[0.5 ; 2]	3	Success
10	map2	[1 ; 2]	1	Failure
11	map2	[1 ; 2]	2	Failure
12	map2	[1 ; 2]	3	Success

Table 4.4: Table describing the co-simulation scenarios that were run and the results.

4.6.2 Analysing Results

Table 4.4 illustrates that the system is specifically not safe in scenarios *4, 5, 10* and *11*. There can be multiple issues related to why the vehicle does not perform a safety stop properly in the failed scenarios.

Analysing the data from the co-simulation of these specific scenarios, made it possible to determine the cause of failure. Figure 4.10 illustrates a situation where the system may not perform a safety stop as it should.
The problem is that the supervisory controller will only perform a safety stop when the sensor perceives an obstacle within the braking distance of the vehicle. This means in situations where the braking distance is less than the distance to the obstacle, as shown in figure 4.10a, the supervisory controller will not perform a safety stop. What will happen is when the vehicle approaches the obstacle, the distance to the obstacle is decreased, and will be lower than the minimum sensor range, meaning the sensor will not perceive the obstacle, as shown in figure 4.10b. The obstacle will actually be within the braking distance, but since it is not perceived by the sensor, the supervisory controller does not register any obstacles within the braking distance, and therefore does not perform a safety stop.
In the scenarios where the vehicle's speed is set to $3\frac{m}{s}$, the braking distance is calculated to be within the sensor range of $[1;2]$ meters. This means the obstacle is within the braking distance and within the sensor range, and therefore the supervisory controller performs a safety stop and avoids collision with the obstacle.

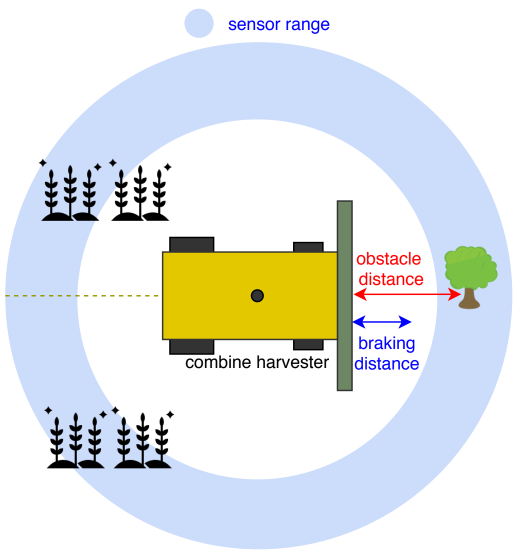
4.6.3 Improving Safety Mechanisms

It is clear that the safety of the system must be improved, in order to avoid collisions while operating in conditions such as the ones in scenarios *4, 5, 10* and *11*. To improve the safety of the system, it is possible to add the following changes to the model, and run co-simulations, for example using design space exploration in order to find the most suitable design:

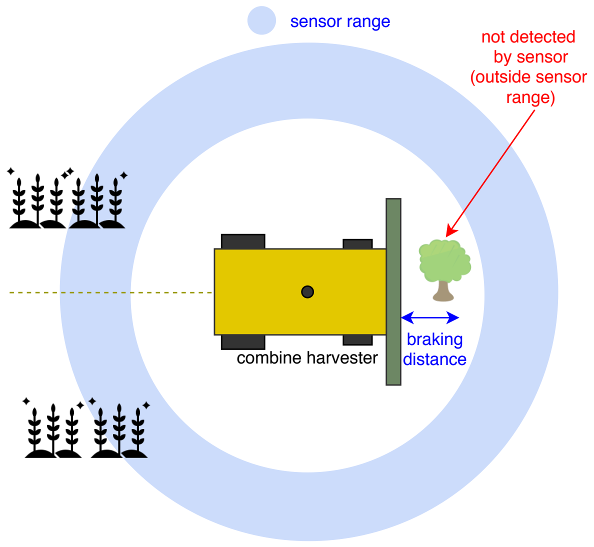
Adjusting safety bound to incorporate minimum sensor range: if the minimum range of the sensor can be determined, then it is possible to set the safety stop distance to be the maximum value of either the minimum sensor range, or the calculated braking distance. This will allow the supervisory controller to perform a safety stop as soon as an obstacle is within the defined bounds. This can only be used when the minimum range of the sensor is known.

Track obstacles: it may be necessary to create a tracking algorithm, to track detected obstacles, and estimate their positions. These estimations may be used to perform a safety stop, to avoid collision with tracked obstacles. In this case it may be necessary to track the obstacles for at least an amount of time, to ensure the system does not perform safety stops, for falsely detected obstacles. A possible tracking algorithm may be implemented using the Kalman Filter [40].

Fault-tolerance: it may be necessary to add some sort of fault tolerance for such situations, where it is possible to use hardware redundancy, such as standby sparing [41]. This is only relevant in the case where the scenario occurs due to sensor failure.



(a) An illustration of a situation where the distance from the vehicle to the obstacle is greater than the calculated braking distance in the supervisory controller. The supervisory controller will not perform a safety stop, since the obstacle is not within the braking distance of the vehicle, thus it is not posing a threat in this case.



(b) An illustration of a situation where an obstacle is within the braking distance of the vehicle, but due to a decreased sensor performance, the obstacle is not detected by the system. The blue circle shows the area in which the detection subsystem can detect objects. This shows that objects that are within the white circular area around the vehicle will not be detected. The supervisory controller does not perform a safety stop, since the object is not detected by the system.

Figure 4.10: Visualization of the conditions where the system failed the safety tests in the co-simulation scenarios. a) shows the vehicle driving towards an obstacle. b) shows the vehicle approaching the obstacle, where it has continued following its path from a).

4.7 Using Co-simulation Results as Evidence

In order to use the co-simulation results as evidence, it is important to note that the model must represent the true system sufficiently for these tests to be legitimate. To determine this, it is necessary to perform tests of defined scenarios on the model and on the true system and compare these results. The performance of the model must be similar to the performance of the system, in order to ascertain that the model sufficiently represents the true system.
The scenarios that will be defined for testing the model and true system, should be scenarios that can easily be constructed, are not unethical, and where the test setup is simple.
The successful co-simulation results from section 4.6.1 are used as evidence in the safety case with the assumption that the model sufficiently represents the true system. The use of the results as evidence in a GSN, is illustrated in figure 4.11. The co-simulation results show that the current system design operates safely in environments with dense fog and rain, but not when inaccurate sensor measurements occur. This means safety improvements to the

system design must be made.
The GSN in figure 4.11 will be added as one of the documents in the complete safety case of the system.

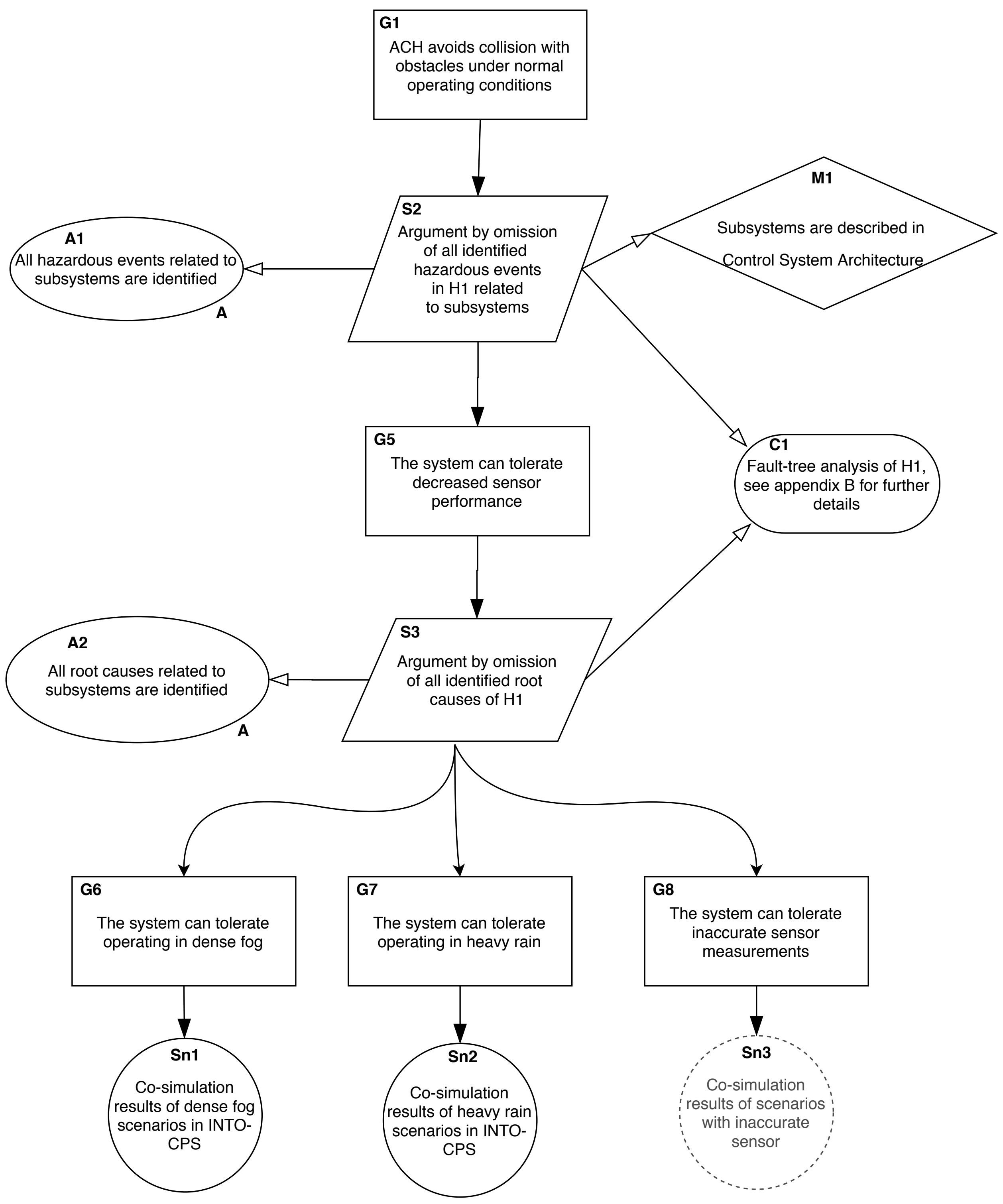

G1
ACH avoids collision with obstacles under normal operating conditions
S2
Argument by omission of all identified hazardous events in H1 related to subsystems
A1
All hazardous events related to subsystems are identified
A
M1
Subsystems are described in Control System Architecture
G5
The system can tolerate decreased sensor performance
C1
Fault-tree analysis of H1, see appendix B for further details
S3
Argument by omission of all identified root causes of H1
A2
All root causes related to subsystems are identified
A
G6
The system can tolerate operating in dense fog
G7
The system can tolerate operating in heavy rain
G8
The system can tolerate inaccurate sensor measurements
Sn1
Co-simulation results of dense fog scenarios in INTO-CPS
Sn2
Co-simulation results of heavy rain scenarios in INTO-CPS
Sn3
Co-simulation results of scenarios with inaccurate sensor

Figure 4.11: The GSN diagram continuing the GSN in figure 3.6. The evidence of the system being able to operate safely in decreased fog and heavy rain is constructed using results from co-simulations of relevant scenarios.

Chapter 5

Process: Incorporating Co-simulation into Development

5.1 Introduction

Safety cases are developed to provide a convincing argument, combined with evidence, that a system is safe under normal operating conditions [6]. Developing a complete safety case is not always a straightforward process [19]. There are multiple safety standards that need to be considered [42]. System operating scenarios, hazards and safety goals must be defined during the process [43]. All of these elements are then used when defining the arguments connecting the evidence with the safety goals. Constructing this evidence is a crucial task, which can be performed using various techniques, where simulation is a possibility [13], though it is not made clear how this may be used in the development process.
A simulation can be performed on models of a system which are realized using the same tool. This can be used in many applications, but when modelling a complex Cyber-Physical System (CPS), it is common that the different parts of the system are modelled in different tools. This is especially in cases where external models are used [12].
This chapter describes the proposed safety case development process, explicitly describing how co-simulation can be incorporated.
Section 5.2 provides a brief overview of the proposed process. Then section 5.3 describes how the safety analysis is performed, followed by section 5.4 which demonstrates how the safety analysis can be used to design co-simulation scenarios. Afterwards section 5.5 describes how such autonomous systems can be modelled, then section 5.6 describes how co-simulation results can be used as evidence in the safety case. Finally, Section 5.7 discusses the challenges and benefits of using co-simulation, followed by section 5.8 which describes which type(s) of evidence is reasonable to obtain using co-simulation.

5.2 Process Overview

The three main elements of a safety case are presenting a claim, constructing evidence, and forming an argument that describes why and how the evidence supports the claim about the

system [14].
The process developed in this thesis project is based on constructing a preliminary safety case for an autonomous combine harvester. The vision of the process is to:

- Create a systematic and clear development process of a safety case.
- Combine safety case development with a technology that allows constructing evidence using various techniques.

The stages of the process are illustrated in figure 5.1. The first stage consists of identifying hazards, and performing a fault-tree analysis, which can be used when identifying safety goals in a Goal Structuring Notation (GSN). The second stage is to discover which safety goals need to be supported by evidence, and determine which of these safety goals are reasonable to satisfy using co-simulation. This leads to the need for designing co-simulation scenarios that are directly related to the hazards of the chosen safety goals. Once the scenarios are defined, a model of the system consisting of the system design and functional safety must be constructed. Co-simulations of the modelled system in the defined scenarios will then be performed. The results of the co-simulation can then, if adequate, be used as evidence in the safety case. If the results are insufficient, then safety improvements can be made to the model, or additional hazards may be identified, as shown in the figure, leading to an iterative development process.
It is important to note that the contributed stages in this development process are stages *3*, *4* and *5*, where in other safety case development processes other techniques will be used. Before continuing to stage *3* it must be determined if co-simulation based evidence will be constructed or if the evidence will be obtained using other techniques, such as mathematical analysis.

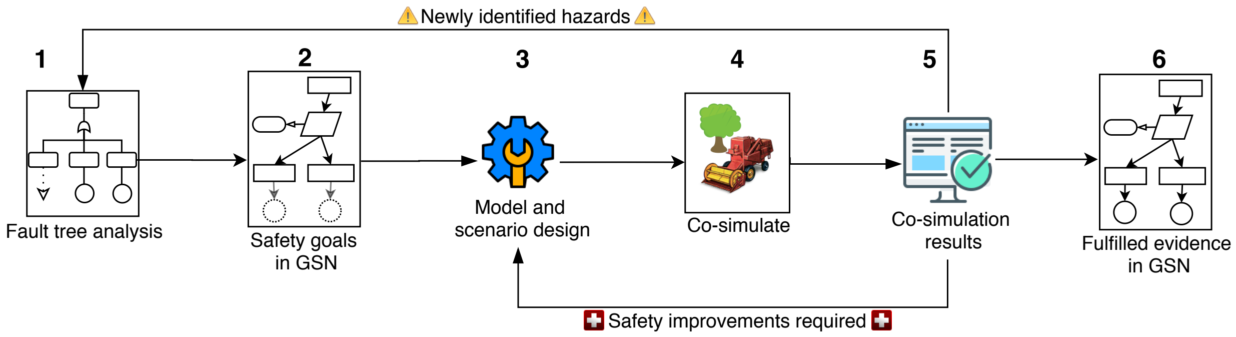


Figure 5.1: An overview of the method of developing a safety case using co-simulation in the development process.

5.3 Safety Analysis

This step consists of constructing the hazard analysis in stage *1* and defining the safety goals in stage *2*, as illustrated in figure 5.1.

The purpose of the safety analysis is to define the safety goals (claims) of the system. The safety goals are one of the main elements in a safety case, which are used as the basis for implementing the necessary safety measures in the system. The steps performed in the safety analysis, are analysing the system with respect to its environment, hazard analysis, and assessing the risks of the hazards. The outcome from the hazard analysis and studying relevant safety standards can then be used to define safety goals and safety requirements.
The safety goals are used in a GSN, which gives an overview of which safety goals can be directly supported by evidence. The next steps are to construct the necessary evidence and form the arguments.
This step is performed as the initial step in the development process, but will also be incorporated in the later stages of the process, when additional hazards are identified. These additional hazards must be analysed, and possibly new safety goals need to be defined.
The complete description of the safety analysis performed on the case study was presented above in chapter 3.

5.4 Coupling Safety Analysis with Co-simulation Scenarios

This step consists of choosing relevant safety goals from the GSN(s) constructed in stage *2* and defining co-simulation scenarios in stage *3*, as illustrated in figure 5.1. It is in this step determined if co-simulation is appropriate to use for constructing evidence in the safety case. The purpose of this step is to determine which elements of evidence in the GSN(s) are suitable to co-simulate, and how to define relevant co-simulation scenarios.
The most important steps when going from a safety analysis to creating co-simulation scenarios are described below:

Choose Appropriate Safety Goals and Evidence: before moving on with using co-simulation in the safety case, it is necessary to determine if it is suitable or not. One of the factors that plays a role in this, is the conditions for testing this part of the system in real life, which depends on the environmental conditions, if the tests performed are ethical, and what the cost of the test setup is. One of the other factors is the complexity of the interactions between the subsystems, especially if it is a CPS. In this step it is crucial to compare the pros and cons of using co-simulation with other evidence construction techniques. It is important to consider the time that will be spent on the development of the model(s) and running co-simulations compared to the time spent on constructing evidence using other techniques.

Contextualize scenario settings: the evidence that needs to be produced must be related to the environmental settings of the system. These settings describe the connection between the evidence in a GSN and the modelling scenarios. This step consists of defining a setting where the system can be tested to prove its safety for each of the evidence components. Essentially, this step defines how the system and environment should be set up in order to perform tests that will fulfill the evidence of the safety case. These tests can also be used on the physical system, where for example specific environmental conditions may be needed, the speed of the vehicle may be defined, and so on.

Define scenario and model parameters: to be able to model the contextualized scenarios, the effect of each event related to the evidence must be determined. This can be done by researching articles related to the context of the evidence, by discussions with domain experts, or using previous data from similar systems. Specific parameter values for co-simulation scenarios must be defined in this process. The purpose of the model and the abstraction level is defined. The purpose of the model is to run co-simulations and use the results as evidence in the safety case. The abstraction level of the model is determined depending on how far in the process, the system development is. For example, in this thesis the preliminary safety case was being created, and therefore the abstraction level was chosen to be high.

The complete description of associating safety analysis with co-simulation scenarios, including performing this on the case study, was presented above in chapters 3 and 4.

5.5 Modelling and Co-simulation

This step consists of modelling the system and designing the scenarios in stage *3* and co-simulating the system in stage *4*, as illustrated in figure 5.1.
The typical steps that will be performed in the modelling and co-simulation stages are described below:

Determine the required models: the models of the system and environment need to be determined, in order to construct a system architecture. These models can be determined by looking at the system description and safety goals defined in stages *1* and *2*, and the scenarios defined in stage *3*. Models of the faults required to construct the scenarios are also determined in this process.

Design system architecture: now that the required models are defined, the system architecture is designed in order to provide an overview of the interconnections between the models.

Model and validate: each model of the subsystems will be created and validated by performing standalone tests of the exported FMUs. After validating each standalone model, a co-simulation combining each of the created FMUs must be performed, in order to validate the system as a whole, before introducing faults. If FMUs of such subsystems already exist, then it is possible to reuse these models.

Design safety mechanisms: to ensure the system is safe, it must contain safety mechanisms that allow it to operate safely in the defined scenarios. Therefore, in this step different methods to mitigate the faults are researched, and at least one safety mechanism is modelled. In many cases it is possible to model various safety mechanisms, and perform a design-space exploration, in order to determine which design functions best according to the safety and system service.

Run co-simulations: The last step is to run co-simulations of each of the defined scenarios found in section 5.4.

The complete description of the modelling and co-simulation of the system in the case study is described above in chapter 4.

5.6 Proving Safety Case Through Co-simulation

This step consists of retrieving the co-simulation results in stage *5*, and using these results as evidence in the GSN in stage *6*, as illustrated in figure 5.1.
The main activities performed are:

Running co-simulation scenarios: when running co-simulations of each of the defined scenarios, it may take time to change each FMU parameter, wait for the simulation to run, and change a parameter again. In the INTO-CPS tool chain it is possible to define the different values of the parameters of each FMU, and allow the tool to run co-simulations with combinations of these parameters. In INTO-CPS this can be done in the Design Space Exploration suite [44].

Extracting results: a convenient method to retrieve the co-simulation results in a structured manner, is to create an FMU that saves the relevant data from the co-simulation to a file. Then a script can be run to determine if the results confirm if the system is safe or not. The idea is to create an automated process, to avoid mistakes, and generate test results that are easy to study.

Analysing scenario results: each scenario is analysed in order to determine which conditions lead to a system failure, making it possible to go back to stage *3* and redesign or improve the modelled safety mechanisms. It may also be the case that additional hazards are identified, making it necessary to go back to stage *1* and perform a hazard analysis, moving down through the process again.

Validating successful results: In order to be able to use co-simulation results as evidence, it is necessary to validate the truth of the results. The results can be validated by performing defined scenario tests on the physical system and model, and compare the results [5]. The scenarios to be defined, must be convenient to test on the physical system and to co-simulate. The scenarios should contain chosen normal and failure scenarios of the system, which are possible to test on the physical system without the need to perform unethical or extremely costly tests. For example an unethical test setup would be to define a scenario where a human must be standing in front of an autonomous vehicle, in order to test if the vehicle will perform a safety stop.

Using results as evidence: once it is assured that the co-simulation results are legitimate, they can be used as evidence in the GSN. Once all the evidence required is constructed, the development of the safety case can be concluded, with the GSN and the rest of the documents associated to the safety case, i.e. system description, hazard analysis, safety goals, standards, etc. If the physical system is not constructed yet, then it is possible to use co-simulation as evidence with the assumption that the model sufficiently represents the true system. In this case the model will first be validated when the physical system is developed.

5.7 Benefits and Challenges

The main benefits of using co-simulation in the safety case development process are seen as the following points:

- It is possible to get early design feedback on safety mechanisms.
- It is possible to explore different designs of the safety mechanisms in the system, by for example using the design-space exploration suite in the INTO-CPS tool chain. The design-space exploration suite also allows automating co-simulation tests.
- It is possible to control the test environment: no unethical tests, independent of environmental conditions such as weather, decrease cost of test setup.
- It is possible to establish rapid modelling by reusing existing models and utilizing frameworks such as PyFMU[1] for FMU development.
- It is possible to generate FMUs from the code running on the true system, and use these in co-simulations.
- It is possible to reuse the co-simulation scenarios to test the system in the later development phases, where the software that will be running on the true system can be packaged as an FMU and used in the co-simulations.

The main challenges and limitations of using co-simulation are seen as the following points:

- The current co-simulation tools available are not mature enough yet [11, 12], meaning that the time that will be used to construct evidence using co-simulation will be larger than necessary.
- It requires domain expertise if models of system are to be created from scratch.
- The model needs to be up to date with the physical system, meaning if any major changes are performed on the physical system, then the corresponding changes need to be performed on the model as well.

[1]`https://github.com/INTO-CPS-Association/pyfmu`

- It is important to understand that the co-simulation results may not always be reliable or valid. An example of invalid results was found during the development of this thesis project. While co-simulating the ACH in the different scenarios, as described above in section 4.6.1, it was clear from the analysis of the results, that the system should have failed in two specific scenarios, while it only failed in one. The cause of this unreliable result, was both due to the order in which the co-simulation engine ran and exchanged signals between the FMUs, but also the size of the time step of the co-simulation. The scenarios were all run with a step size of 1 second, but after determining the results were unreliable, the co-simulations were performed again, but with a step size of 0.5 seconds. The results of running the co-simulation with a lower step size were deemed reliable and correct. This example illustrates how it may be a challenging task to determine the validity of all the co-simulation results.

5.8 Determine Evidence Construction Technique

This section describes how to determine which technique(s) to use when constructing evidence for specific safety goals of a safety case. Each safety goal needs to be supported by evidence, thus such goals must be analysed in order to determine which evidence construction technique is most suitable to use.
Figure 5.2 illustrates how it is possible to choose between various evidence construction techniques when developing a safety case. It is also possible to combine a number of these evidence construction techniques.
The time spent on creating the model, defining the scenarios, analysing the results, and performing tests of the physical system and model must be taken into account. This must be compared to the time spent on for example performing a mathematical analysis and proof. It is important to consider the pros and cons of creating a co-simulation compared to, for example, performing a mathematical analysis.
The most relevant considerations to make when determining if it is justifiable to use co-simulation are:

- The amount of time spent on developing model(s), defining co-simulation scenarios and running tests, is smaller than the time spent on constructing other types of evidence. For example if it is possible to prove the safety goals using geometry proof, then this is of course a more suitable technique, since it takes less time than creating and co-simulating a model of a system. When comparing the time spent on the different evidence techniques, it is also important to look at the advantages that come with co-simulation. An example of an advantage, is that the same models can be reused for constructing evidence related to different safety goals, or reused in system development.

- If similar systems exist, their hazard logs may be used as part of the evidence. If they do not exist, then it may be appropriate to run co-simulations of the system in various scenarios.

- If it may be relevant to model the interactions between the software and the physical system.

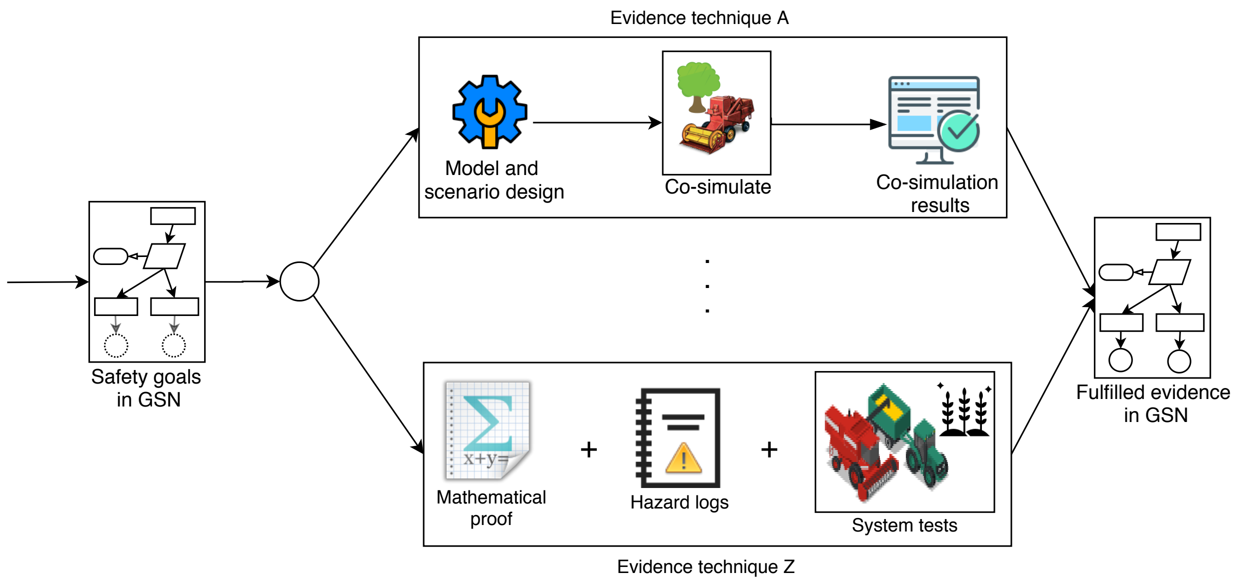


Figure 5.2: An illustration of the step required to use co-simulation as evidence compared to other possible techniques for constructing evidence in a safety case. The figure demonstrates that after defining the safety goals, the technique for constructing the different evidence must be determined.

- If there are complicated interactions between the subsystems, then it may be difficult to perform a mathematical analysis of this.
- Compare the benefits and challenges of using co-simulation, as described in section 5.7.

Chapter 6

Concluding Remarks and Future Work

6.1 Introduction

This chapter gives a summary of the main benefits and challenges determined during the work of the thesis, and briefly describes what was achieved. Section 6.2 discusses the main outcome of the thesis, followed by section 6.3 which evaluates the goals of this thesis. Afterwards, section 6.4 describes potential future work related to the work carried out in this thesis. Finally, section 6.5 summarizes the main outcome of this thesis.

6.2 Discussion

This section describes the main benefits and challenges of co-simulation in a safety case development process, and also relates these to the case study of this thesis. Subsection 6.2.1 describes the benefits and challenges of reusing models, followed by subsection 6.2.2 which describes advances in modelling tools. Afterwards, subsection 6.2.3 describes how the model fidelity is related to the different process phases. Finally, subsection 6.2.4 describes additional outcomes from the development of the safety case for the case study of this thesis.

6.2.1 Reusing Models

This subsection describes the benefits and challenges of reusing models in the proposed development process.
Models are created and used for a purpose, and a reused model must reflect a similar purpose. High abstraction level models that represent a general functionality can be easier to reuse. This is due to the fact that such models are general for a number of systems. For example a model of an object detection system which represents fused sensor data, can be reused in multiple autonomous vehicles, since its purpose is to detect obstacles, where the modelling of the system is performed at a high abstraction level. If the modelling must be performed at a lower abstraction level where each specific sensor must be modelled standalone, then a different model must be used for the various systems if they use different sensor types. In this case it may be possible to use models from the OEMs of the sensors.

The benefits of reusing models is the time saved on creating the model, but also if the model has been validated for a similar system, then there is a higher possibility that the model is valid. It is important when reusing a model that the functionality is understood in order to avoid any incorrect assumptions or erroneous use of the model.
Challenges of reusing models may be if the model has not been properly validated, or if the model performs operations that are not documented or may be difficult to understand. A challenge is the fact that FMUs can be represented as black boxes containing a model, with only specific inputs and outputs available. Such models may be difficult to use, due to possible assumptions made in the model that are not explicit, the operating area may not be defined, the precision and the limitations of the model may not be described either. These challenges hold back the reuse of models, meaning some of the advantages of co-simulation are not leveraged. Hopefully in the future such practical limitations will be mitigated by creating the following:

Model database: a model database should be created, containing different types of models with different abstraction levels. Each of these models should implement a standard such as FMI, allowing usage of these models in different co-simulation tools. Component providers should develop models of their components and add them to the database, to increase reuse possibilities of low abstraction level models. This database could with advantage be inspired by package managers such as *pip*[1], which provides a simple interface for installing new packages.

Documentation convention: a convention for documenting models must be created, in order to reuse FMUs in various co-simulations and projects. The standard must define how to document the assumptions made, the operating environment, the precision and limitations of the model. The documentation of these models may advantageously incorporate modelling frames, which document information about the model context and behaviour [45]. Documenting some of these model characteristics may be difficult. For example, the operating environment can be documented by testing a model in different ranges. A model of a vehicle could be tested with different speeds, where descriptions of the model behaviour at these speeds can be provided.

A possibility would be to combine the documentation convention with the model database, where each model in the database has a document describing the model assumptions and so on. Figure 6.1 illustrates the proposed ideas of how to mitigate the practical challenges of reusing models.
This thesis has showed how it is possible to reuse black box models, and further develop the functionality of a system by combining these reused models with models created during the thesis project. This showed the simplicity of the process of combining models developed in different tools and in different languages can be.
The models that were reused in this thesis project, the Controller and Vehicle FMUs, were not fully documented or validated. Complications occurred when the speed setpoint was increased and the vehicle performed a turn in attempt to follow the given route. The deviation between the vehicle and the route increased, and in some cases, the vehicle would

[1]`https://pypi.org/project/pip/`

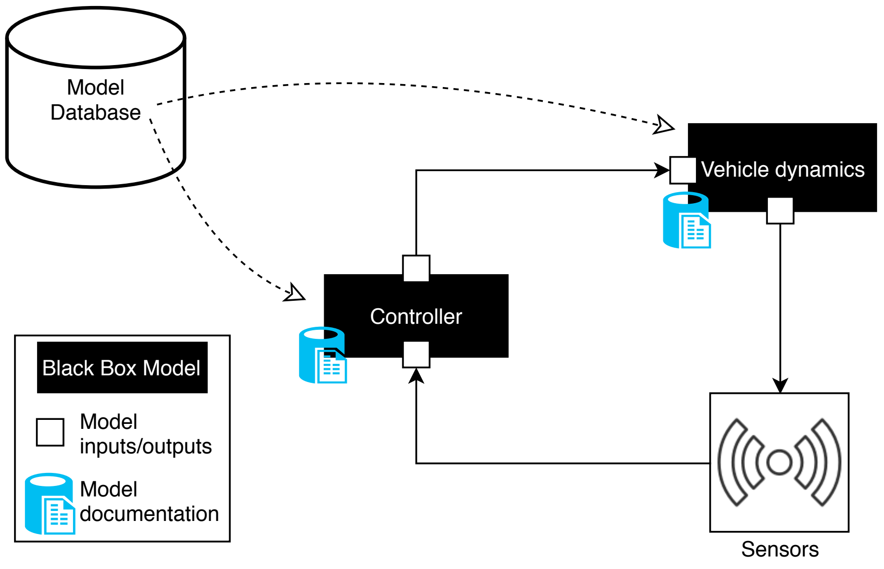


Figure 6.1: An illustration of how models from a database can be used. The models presented are black boxes that have inputs and outputs. Each of the models from the database are documented following a convention. The sensor module in this case is self-developed, while the controller and vehicle dynamics are reused from the database.

discontinue following the given route as illustrated in figure 6.2. The cause of this is possibly because of the simplicity of the model. However, this does not work when decreasing the speed of the vehicle when it performs a turn. This results in the vehicle taking large turns which increases the deviation from the given route. In this case the vehicle dynamics were mainly used to allow the vehicle to drive around an environment and determine if the vehicle performs a safety stop correctly. The best results will of course be obtained by using a model that follows the route as intended, but in this case the results can still be used to demonstrate the safety of the system.

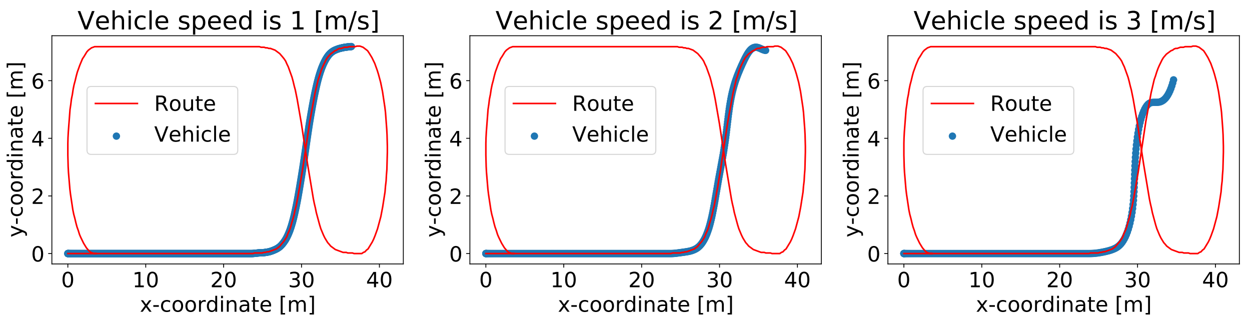


Figure 6.2: These three plots illustrate the given route (red, thin line) and the route that the vehicle has driven (blue, thick line) at three different speeds, 1, 2 and 3 m/s. As the third plot, with a vehicle speed of 3 m/s displays, the model becomes unstable at this speed, and therefore does not follow the route as intended.

6.2.2 Advancing Modelling Tools

To incorporate co-simulation in a safety case development process, it is necessary that tools supporting this are available and straightforward to use. In this thesis project the INTO-CPS tool chain was used for running the co-simulations. The reused FMUs were developed in the tools Overture and 20-sim, while the FMUs developed during this thesis were produced using the tool PyFMU[2].
Research in advancing modelling and co-simulation tools is an ongoing process [11], emphasizing some of these tools are not mature enough yet for industrial use. To ease the adaptation of co-simulation in the development process of Cyber-Physical Systems (CPSs), such tools need to be accessible. For example PyFMU allows generating FMUs from a python script, which is a language that is used by various developers around the world [46]. By allowing developers to construct FMUs in Python, it gives the possibility of a smoother adaptation of co-simulation in the development of safety-critical systems.
One of the benefits of the current co-simulation tools, such as the INTO-CPS tool chain, is the fact that it is possible to perform design-space explorations. For safety-critical systems this tool suite is a great advance that allows experimenting with different designs of safety mechanisms. The results of these experiments can be used to determine which design provides the best compromise between functional safety, cost and system availability.
The design-space exploration in this thesis was performed using the co-simulation tool FMPy[3], where parameters of the FMUs were changed manually and co-simulations were run for each parameter change that reflected a new scenario. The INTO-CPS design-space exploration tool was not used during the development of this thesis, since FMUs created using PyFMU must run in Python 3.7, while the design-space exploration in INTO-CPS must run with Python 2.7.

6.2.3 Model Fidelity

When modelling a system, it is good practice to gradually increase the complexity of the model [5], to better ensure the model operates correctly at each level. This development process of increasing the fidelity of the model can be used during the progress of the safety case. The safety case scenarios defined during the preliminary safety case can be reused or further extended in the interim safety case and the same applies to the operational safety case. As figure 6.3 illustrates, the same co-simulation scenarios can be used with the models of different fidelity levels. The reason for this is each model implements the FMI standard, which allows the co-simulation engine to exchange information using the inputs and outputs defined in the model interface. When models are further developed and become more detailed, the same interfaces of inputs and outputs can be developed to run the same co-simulation scenarios on the more detailed models [23].
The model used in this thesis project is primitive, since it is the first model of the system created for use in the development of the safety case. During the progress of the development of the system and its safety, increasing levels of details will be added to the model. This

[2]`https://github.com/INTO-CPS-Association/pyfmu`
[3]`https://github.com/CATIA-Systems/FMPy`

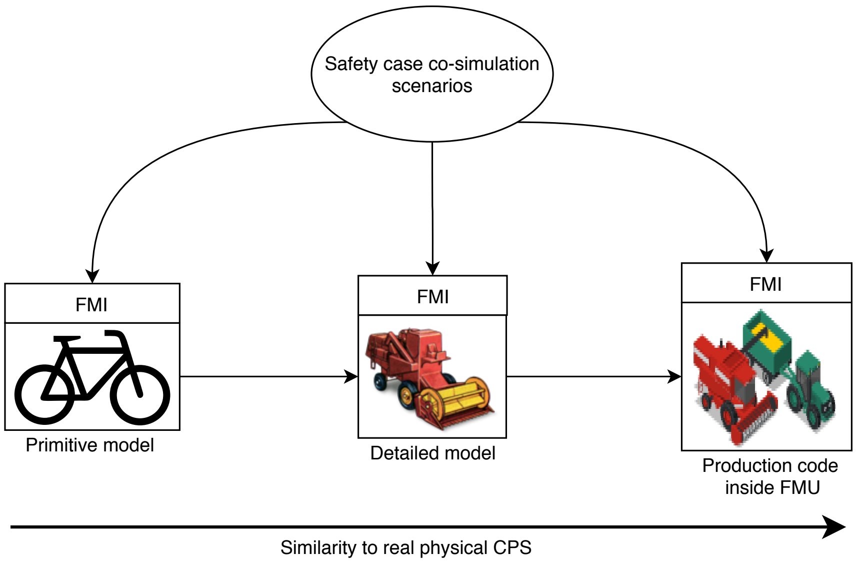

Safety case co-simulation scenarios
FMI
FMI
FMI
Primitive model
Detailed model
Production code inside FMU
Similarity to real physical CPS

Figure 6.3: An illustration of how the model(s) used can be extended with more details throughout the development of the system. At the beginning of the development process, the system design is simple and incomplete, with time an increasing amount of details are added to the design, this also applies to the model of the system. The same safety case co-simulation scenarios can be used to test the models constructed throughout the process.

more detailed model can then be used with some of the same co-simulation scenarios defined in this thesis project. The vision is that in the future, when the operational safety case must be constructed, the production code can be encapsulated in an FMU, and models (if available) of the hardware components, such as sensors, actuators etc. can be used in the model of the system.

6.2.4 Additional Outcomes

Apart from researching the use of co-simulation in the development of safety-critical systems, additional outcomes were produced during this thesis project. These are described below, where each of the functionalities can be used not only in the case study of this thesis, but in other CPSs:

Changing environment: Autonomous systems operate in dynamic environments, which must be included in co-simulations. The environment of the CPS was realised as a standalone FMU, where an image could be loaded, and by using image processing tools, the obstacles in the image could be detected and the positions of the obstacles could be outputted from the FMU. By setting up the environment using an image, this makes the process of co-simulating the system using different environmental conditions easier. Currently, the environment only registers static obstacles, but this can be further developed and incorporate FMUs of dynamic obstacles.

Reusable sensor module: Autonomous systems rely only on sensors to detect changes in their environment. A high abstraction level sensor module was developed, which models the combination of various sensors. The minimum and maximum range of the module are set as parameters that can be modified before a co-simulation run. This is used to model the sensory system in different environments, for example to allow modelling the system in dense fog, where the visibility of the sensors may be decreased. The current sensor module can detect objects in a circular radius, but can be modified to detect obstacles in a different shape.

False positive detection may occur in such sensory systems, due to e.g. insects near the sensors, water droplets reflecting the rays from the LiDAR and other reasons. Generating false positive objects in the sensor module allows modelling such circumstances, where a mean time to failure rate is given for controlling the amount of generated false positives.

Live plotting: Running a co-simulation takes time, and in many cases one or two parameters may be set incorrectly, the system may fail early in the simulation while the developer must wait until the co-simulation has finished running before being able to view the results. Spending time on these human mistakes makes the process of running co-simulations even longer. By using an FMU which plots the autonomous system simultaneously with the co-simulation run, it allows to earlier discover errors in the co-simulation. The live plotting FMU developed in this thesis, also plots the range of the sensors while the vehicle is driving, which gives better insight of why in some cases the system may behave incorrectly.

6.3 Evaluation of Goals

This section evaluates the initial goals presented in section 1.3 above. The goals are described again, followed by an evaluation of them.

6.3.1 Revisiting the Goals

In chapter 1 the following goals were defined for this thesis project:

Goal 1: Define a procedure for incorporating co-simulation into the development process of a safety-critical system.

Goal 2: Demonstrate the feasibility of the process in the context of a case study.

Goal 3: Establish how safety-case analysis can be used to feed into the process of defining co-simulation scenarios.

Goal 4: Improve personal research related skills and improve documentation skills of defining a development process.

6.3.2 Evaluation

Goal 1: This goal was *partially accomplished.* A procedure for how to incorporate co-simulation into the development process of a safety-critical system was defined. This procedure was established from the work carried out in this thesis project, by constructing a preliminary safety case and researching where and how co-simulation can be incorporated into this process. The procedure defines the different stages in the proposed process: analyse system safety aspects and goals, determine if evidence is appropriate to construct using co-simulation, define co-simulation scenarios, model and co-simulate, use co-simulation results as evidence in a safety case. Chapter 5 describes the proposed process of developing a safety-critical system with co-simulation incorporated in the process. The process was constructed by performing a preliminary hazard analysis of the case study, where this was used to define the safety goals. The process has not yet been defined for a complete safety case, but only a preliminary safety case.

Goal 2: This goal was *partially accomplished.* The process was used in the development of a chosen part of the safety case for an autonomous combine harvester. The co-simulation results that demonstrated the system safety in specific scenarios were used as evidence in the safety case. Section 4.6.1 describes which results were used as evidence, and why these specific co-simulation results could be used to demonstrate the safety of the system. The models used in this thesis project were tested separately, in order to ensure they function correctly, but these models have not been validated with comparison to the physical system. Therefore, when using co-simulation results from models that have not been validated, there is an underlying assumption that the co-simulation results demonstrate the operation of the true system. The next step would be to validate the model by comparing its behaviour with a prototype of the system.

Goal 3: This goal was *partially accomplished.* The process in chapter 5 describes how information from the outcome of a safety analysis of a system can be used to define relevant co-simulation scenarios to use in the safety case. The safety goals are used to define which evidence needs to be constructed, while the fault-tree analysis or research articles related to the safety goals are used to define the co-simulation scenarios. However, again it was not possible to relate any of the evidence to the real physical system.

Goal 4: This goal was *accomplished.* During the process of creating this thesis, it required the author was independent in both planning their time and which path to explore or work further on. It is clear that in the beginning of the thesis, these skills were not well developed, but throughout the process, the author has become better at both planning when and what to spend their time on, but also better at determining which articles are relevant to read and which path to follow. These skills are general research skills that have been improved and can definitely aid the author when working on other projects. The documentation skills of the author have been improved, by iteratively writing this thesis and receiving valuable feedback on the structure and content.

6.4 Future Work

This section describes potential future research that can be conducted in the area of incorporating safety cases with co-simulation.

Extend process: The process has currently only been used in the development of a preliminary safety case. The process could be extended to the complete development process, from a preliminary safety case to the operational safety case of a system, progressing simultaneously with the development of the system itself. The fidelity of the models and how the co-simulation scenarios can be constructed and further developed throughout these safety case phases, could be included in the guidelines of the proposed development process.

For example, figure 6.3 illustrates how the co-simulation scenarios can be applied to the models, with increasing detail, of a system, where it is imagined that the abstraction level of the model in each safety case phase can be explicitly defined. It is also possible to imagine that a definition of how the co-simulation scenarios can be defined can be proposed, in order for the process of reusing these to be as smooth as possible.

Model validation: The models used in this thesis project that represent the system have not been validated against the real physical CPS. This is something that should be performed, where the outcome of this can be added to the guidelines of the proposed development process. It may also be relevant to research different techniques of model validation, and when to use a specific technique rather than another.

The model can be validated by defining a limited number of scenarios, which the physical CPS and model are tested in. The behaviour of the system and model is then compared to determine if the model sufficiently represents the system.

Evaluate process on other systems: Since the development process was only used in a single case study, it is difficult to determine how well the adoption of this process will be when developing other systems. For example there is a difference in developing an autonomous safety-critical system compared to developing a medical device that will be operated by humans. Therefore, it is relevant to use the proposed development process in the development of other systems and determine the suitability of the process.

For example, research in the use of this process in medical devices may prove that other techniques need to be incorporated in the development process, since there is a large focus on how well the operator of the device understands and correctly uses the device. A possible extension to the process could be to use the *Prototype Verification System* for verifying user interfaces of medical devices [47].

6.5 Conclusion

Ensuring the safety of CPSs is a difficult and lengthy process. It is not possible to test these systems in every operational state, due to unethical test setups or difficulties in controlling

the environment around the system. Different techniques need to be included in the development process in order to decrease the amount of time, the cost, and the uncertainty of the system safety. This thesis project studied the feasibility of incorporating co-simulation in the development process of safety-critical systems. The outcome of this work led to defining guidelines that can be followed to determine when and how to use co-simulation when developing safety-critical systems.
The proposed procedure was used in the development of the safety case of an autonomous vehicle. The results proved that co-simulation is convenient to use in cases where complex interactions between subsystems occur, where it is difficult to prove the safety using other means. It is important to note that this development process has been used and determined to be suitable in the development of autonomous systems. It has not been used in the development of other types of safety-critical systems, such as medical devices where the system uncertainty also comes from humans operating the system.

Acronyms

ACH Autonomous Combine Harvester iv, 3, 4, 17, 28–30, 33, 36, 37, 52, 72

AGCO Allis-Gleaner Corporation 18

ALARP As low as reasonably practical 12, 19

CPS Cyber-Physical System 7, 11, 15, 46, 48, 57, 58, 61

FMI Functional Mock-Up Interface 15, 16, 55, 57, 96

FMU Functional Mock-Up Unit iii, iv, vi, 9, 15, 16, 35–37, 49–52, 55, 57–59, 96–101, 103, 104

GSN Goal Structuring Notation iii–vi, 13, 17, 18, 20, 24–28, 30, 31, 43–45, 47, 48, 50, 51, 76, 77

OEM Original Equipment Manufacturer 1, 16, 54

Glossary

co-simulation A simulation of collaborative models, where the models are developed using different tools [24]. iii, vii, 1, 2, 4–10, 14–16, 20, 23, 24, 28, 29, 31–35, 37, 40–43, 46–55, 57–62, 78

cyber-physical system A system which consists of both computational and physical processes that collaborate with each other to form the whole system [48]. 6, 11, 15, 46, 48, 57, 58, 61

mean time to failure The predicted average time between athe occurrence of a failure in a component[4]. 11, 59, 100

reaping Cut or gather (a crop or harvest) [49]. 72

robust design The design of a product, where the product can operate correctly in dynamic environments [50]. 29

threshing Separate grain from (corn or other crops), typically with a flail or by the action of a revolving mechanism [49]. 71, 72

winnowing Blow a current of air through (grain) in order to remove the chaff [49]. 72

[4]`https://en.wikipedia.org/wiki/Mean_time_between_failures`

References

[1] J. C. Knight, “Safety critical systems: Challenges and directions”, in *Proceedings of the 24th International Conference on Software Engineering*, ser. ICSE ’02, Orlando, Florida: Association for Computing Machinery, 2002, pp. 547–550, ISBN: 158113472X. DOI: 10.1145/581339.581406. [Online]. Available: https://doi.org/10.1145/581339.581406.

[2] Linling Sun and T. Kelly, “Safety arguments in aircraft certification”, in *4th IET International Conference on Systems Safety 2009. Incorporating the SaRS Annual Conference*, 2009, pp. 1–6.

[3] H. Martin, K. Tschabuschnig, O. Bridal, and D. Watzenig, “Functional safety of automated driving systems: Does iso 26262 meet the challenges?”, in. Sep. 2017, pp. 387–416, ISBN: 978-3-319-31893-6. DOI: 10.1007/978-3-319-31895-0_16.

[4] “Iso 26262: Road vehicles – functional safety”, International Organization for Standardization, Geneva, CH, Norm, 2011.

[5] A. Maria, “Introduction to modeling and simulation”, in *Proceedings of the 29th Conference on Winter Simulation*, ser. WSC ’97, Atlanta, Georgia, USA: IEEE Computer Society, 1997, pp. 7–13, ISBN: 078034278X. DOI: 10.1145/268437.268440. [Online]. Available: https://doi.org/10.1145/268437.268440.

[6] I. Sommerville, *Software Engineering*, 9th. USA: Addison-Wesley Publishing Company, 2010.

[7] (). Fendt ideal 9t specifications & technical data (2018-2020), [Online]. Available: https://www.lectura-specs.com/en/model/agricultural-machinery/combine-harvesters-fendt/ideal-9t-11724959. (accessed: 11.05.2020).

[8] (). Combine harvester new holland cx840, [Online]. Available: http://speceps.com/spec/agricultural-1538/new-holland/cx840.html. (accessed: 11.05.2020).

[9] AGCO, *Fendt ideal 10t technical specifications*, Accessed: 03-05-2020.

[10] N. Isaac, G. Quick, S. Birrell, W. Edwards, and B. Coers, “Combine harvester econometric model with forward speed optimization”, *Applied Engineering in Agriculture*, vol. 22, Jun. 2006. DOI: 10.13031/2013.20184.

[11] G. Schweiger, C. Gomes, G. Engel, J. Schoeggl, A. Posch, I. Hafner, and T. Nouidu, “An empirical survey on co-simulation: Promising standards, challenges and research needs”, *CoRR*, vol. abs/1901.06262, 2019. arXiv: 1901.06262. [Online]. Available: http://arxiv.org/abs/1901.06262.

[12] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, “Co-simulation: A survey”, *ACM Comput. Surv.*, vol. 51, no. 3, May 2018, ISSN: 0360-0300. DOI: `10.1145/3179993`. [Online]. Available: `https://doi.org/10.1145/3179993`.

[13] P. Bishop and R. Bloomfield, “A methodology for safety case development”, in *SAFETY-CRITICAL SYSTEMS SYMPOSIUM, BIRMINGHAM, UK, FEB 1998*, Springer-Verlag, ISBN 3-540-76189-6, 1998.

[14] T. Kelly and R. Weaver, “The goal structuring notation – a safety argument notation”, in *Proc. of Dependable Systems and Networks 2004 Workshop on Assurance Cases*, 2004.

[15] W. R. Dunn, “Designing safety-critical computer systems”, *Computer*, vol. 36, no. 11, pp. 40–46, Nov. 2003, ISSN: 0018-9162. DOI: `10.1109/MC.2003.1244533`. [Online]. Available: `https://doi.org/10.1109/MC.2003.1244533`.

[16] G. Despotou, R. Alexander, and T. Kelly, “Addressing challenges of hazard analysis in systems of systems”, in *2009 3rd Annual IEEE Systems Conference*, 2009, pp. 167–172.

[17] G. Despotou, S. White, T. Kelly, and M. Ryan, “Introducing safety cases for health IT”, in *Proceedings of the 4th International Workshop on Software Engineering in Health Care, SEHC 2012, Zurich, Switzerland, June 4-5, 2012*, R. Breu and J. Hatcliff, Eds., IEEE Computer Society, 2012, pp. 44–50. DOI: `10.1109/SEHC.2012.6227010`. [Online]. Available: `https://doi.org/10.1109/SEHC.2012.6227010`.

[18] T. A. C. W. Group, *Goal structuring notation community standard*, Accessed: 05-05-2020, Feb. 2018.

[19] T. Kelly, “A systematic approach to safety case management”, Jan. 2003. DOI: `10.4271/2004-01-1779`.

[20] ——, “Evidence based certification: The safety case approach”, 2008. [Online]. Available: `https://www.umsec.umn.edu/sites/www.umsec.umn.edu/files/TimKelly.pdf`.

[21] I. Bate, A. Burns, T. Kelly, and J. McDermid, “Building a preliminary safety case: An example from aerospace”, in *Proceedings of the 1997 Australian Workshop on Industrial Experience with Safety Critical Systems and Software, Sydney, Australia*, 1997, pp. 1–10.

[22] J. Amerongen, “Dynamical systems for creative technology”, Jan. 2010.

[23] M. Neghina, C. Zamfirescu, P. Larsen, K. Lausdahl, and K. PIERCE, “Multi-paradigm discrete-event modelling and co-simulation of cyber-physical systems”, *Studies in Informatics and Control*, vol. 27, Mar. 2018. DOI: `10.24846/v27i1y201804`.

[24] J. Fitzgerald, P. G. Larsen, and M. Verhoef, *Collaborative Design for Embedded Systems: Co-Modelling and Co-Simulation*. Springer Publishing Company, Incorporated, 2014, ISBN: 3642541178.

[25] M. Association, “Functional mockup interface for model exchange and co-simulation”, Aug. 2019.

[26] M. Bijelic, T. Gruber, and W. Ritter, "A benchmark for lidar sensors in fog: Is detection breaking down?", in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2018, pp. 760–767. DOI: `10.1109/IVS.2018.8500543`.

[27] R. Heinzler, P. Schindler, J. Seekircher, W. Ritter, and W. Stork, "Weather influence and classification with automotive lidar sensors", Jun. 2019, pp. 1527–1534. DOI: `10.1109/IVS.2019.8814205`.

[28] Z. Liu, Y. He, C. Wang, and R. Song, "Analysis of the influence of foggy weather environment on the detection effect of machine vision obstacles", *Sensors*, vol. 20, no. 2, p. 349, Jan. 2020, ISSN: 1424-8220. DOI: `10.3390/s20020349`. [Online]. Available: `http://dx.doi.org/10.3390/s20020349`.

[29] T. Koyluoglu and L. Hennicks, "Evaluating rain removal image processing solutions for fast and accurate object detection", Master's thesis, KTH ROYAL INSTITUTE OF TECHNOLOGY, Sweden, 2019.

[30] T. Kelly, *Safety case patterns catalogue.*

[31] "Architecture of safety-critical systems", Aug. 2005. [Online]. Available: `https://www.embedded.com/architecture-of-safety-critical-systems/`, (accessed: 08/05/2020).

[32] K. Amarendra and A. V. Rao, "Safety critical systems analysis", *Global journal of computer science and technology*, 2011.

[33] S. H. Almutairi and N. Aouf, "Aircraft robust flight tracking control against actuator efficiency faults", in *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)*, 2015, pp. 901–906.

[34] R. Clark. (). Selecting a lidar system, [Online]. Available: `http://www.acuitylidar.com/PDFs/Selecting%5C%20a%5C%20LIDAR%5C%20System.pdf`. (accessed: 10.05.2020).

[35] S. Lee, H. Cho, K.-J. Yoon, and J. Lee, *Intelligent Autonomous Systems 12: Volume 1 Proceedings of the 12th International Conference IAS-12, Held June 26-29, 2012, Jeju Island, Korea.* Springer Publishing Company, Incorporated, 2012, ISBN: 3642339255.

[36] M. Bijelic, T. Gruber, F. Mannan, F. Kraus, W. Ritter, K. C. J. Dietmayer, and F. Heide, "Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather.", *arXiv: Computer Vision and Pattern Recognition*, 2020.

[37] D. Lee, S. Kim, C. Kim, and K. Huh, "Development of an autonomous braking system using the predicted stopping distance", *International Journal of Automotive Technology*, vol. 15, pp. 341–346, Mar. 2014. DOI: `10.1007/s12239-014-0035-5`.

[38] R. Hawkins, K. Clegg, R. Alexander, and T. Kelly, "Using a software safety argument pattern catalogue: Two case studies", Sep. 2011, pp. 185–198. DOI: `10.1007/978-3-642-24270-0_14`.

[39] (). Stopping distance formula, [Online]. Available: `https://byjus.com/stopping-distance-formula/`. (accessed: 08.05.2020).

[40] K. Saho, "Kalman filter for moving object tracking: Performance analysis and filter design", in. Feb. 2018, ISBN: 978-953-51-3827-3. DOI: `10.5772/intechopen.71731`.

[41] A. Ejlali, B. M. Al-Hashimi, and P. Eles, “A standby-sparing technique with low energy-overhead for fault-tolerant hard real-time systems”, in *Proceedings of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS ’09, Grenoble, France: Association for Computing Machinery, 2009, pp. 193–202, ISBN: 9781605586281. DOI: `10.1145/1629435.1629463`. [Online]. Available: `https://doi.org/10.1145/1629435.1629463`.

[42] P. Koopman, U. Ferrell, F. Fratrik, and M. Wagner, “A safety standard approach for fully autonomous vehicles”, in. Aug. 2019, pp. 326–332, ISBN: 978-3-030-26249-5. DOI: `10.1007/978-3-030-26250-1_26`.

[43] R. Alexander, N. Herbert, and T. Kelly, “Structuring safety cases for autonomous systems”, Nov. 2008, pp. 1–6. DOI: `10.1049/cp:20080730`.

[44] C. Gamble. (2016). Dse in the into-cps platform. version 1.0, [Online]. Available: `https://projects.au.dk/fileadmin/D5.2d_DSE_in_the_INTO-CPS_Platform.pdf`. (accessed: 11.05.2020).

[45] S. Klikovits, J. Denil, A. Muzy, and R. Salay, “Modeling frames”, in *14th Workshop on Model-Driven Engineering, Verification and Validation (MoDeVVa 2017), 19 September 2017, Austin, TX, USA*, ser. CEUR Workshop Proceedings, vol. 2019, 2017, pp. 315–320.

[46] Jetbrains. (2019). Python developers survey 2019, [Online]. Available: `https://www.jetbrains.com/lp/python-developers-survey-2019/`. (accessed: 23.05.2020).

[47] P. Masci, Y. Zhang, P. Curzon, M. Harrison, P. Jones, and H. Thimbleby, “Verification of software for medical device user interfaces in pvs”, *CHI_ MED Technical Report*, Jan. 2013.

[48] E. A. Lee, “Cps foundations”, in *Proceedings of the 47th Design Automation Conference*, ser. DAC ’10, Anaheim, California: Association for Computing Machinery, 2010, pp. 737–742, ISBN: 9781450300025. DOI: `10.1145/1837274.1837462`. [Online]. Available: `https://doi.org/10.1145/1837274.1837462`.

[49] A. Stevenson, *Oxford dictionary of English*. Oxford University Press, 2011.

[50] T. Eifler. (2018). Robust design, [Online]. Available: `https://www.mek.dtu.dk/english/Sections/KandP/Research/Robust-design`. (accessed: 23.05.2020).

[51] (Feb. 2013). Self-propelled straw walker combine harvesters - function and working, [Online]. Available: `https://hubpages.com/education/Straw-walker-Combine-Harvesters-Function-and-working`. (accessed: 13/03/2020).

[52] Y. Z. Zhong Tang Haotian Zhang and Y. Li, “Effects of stem cutting in rice harvesting by combine harvester front header vibration”, Feb. 2019. [Online]. Available: `https://doi.org/10.1155/2019/6834269`.

[53] Food and A. O. of the United Nations. (Aug. 2019). Global information and early warning system - country brief: Indonesia, [Online]. Available: `http://www.fao.org/giews/countrybrief/country.jsp?code=IDN`. (accessed: 23.03.2020).

[54] "Iso 18497: Agricultural machinery and tractors – safety of highly automated agricultural machines – principles for design", International Organization for Standardization, Geneva, CH, Standard, Nov. 2018.

[55] "Iso 4254: Agricultural machinery – safety – part 7: Combine harvesters, forage harvesters, cotton harvesters and sugar cane harvesters", International Organization for Standardization, Geneva, CH, Standard, 2017. [Online]. Available: `https://www.iso.org/obp/ui/#iso:std:iso:4254:-7:ed-3:v2:en`, (accessed: 09.03.2020).

[56] I. Search. (). Sick - 2d lidar sensors, [Online]. Available: `https://www.industrysearch.com.au/2d-lidar-sensors-sick/p/143239`. (accessed: 27.05.2020).

[57] Baumer. (). Radar sensors, [Online]. Available: `https://www.baumer.com/dk/en/product-overview/distance-measurement/radar-sensors/c/291`. (accessed: 27.05.2020).

[58] J. Val, M. Videgain, P. Martin-Ramos, M. Cortés, A. Boné-Garasa, and F. Ramos, "Fire risks associated with combine harvesters: Analysis of machinery critical points", vol. 9, p. 877, Dec. 2019. DOI: `10.3390/agronomy9120877`.

[59] A. G. R. D̃. C. (GRDC). (Nov. 2017). Reducing harvest fires: The back pocket guide, [Online]. Available: `https://grdc.com.au/ReducingHarvestFiresBPG`. (accessed: 21/03/2020).

[60] P. G. Larsen, J. Fitzgerald, J. Woodcock, P. Fritzson, J. Brauer, C. Kleijn, T. Lecomte, M. Pfeil, O. Green, S. Basagiannis, and A. Sadovykh, "Integrated tool chain for model-based design of cyber-physical systems: The into-cps project", in *2016 2nd International Workshop on Modelling, Analysis, and Control of Complex CPS (CPS Data)*, Apr. 2016, pp. 1–6. DOI: `10.1109/CPSData.2016.7496424`.

[61] F. F. Foldager, P. G. Larsen, and O. Green, "Development of a driverless lawn mower using co-simulation", in *Software Engineering and Formal Methods*, A. Cerone and M. Roveri, Eds., Cham: Springer International Publishing, 2018, pp. 330–344, ISBN: 978-3-319-74781-1.

[62] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm", Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, Jan. 1992.

Appendix A

Safety Analysis

A.1 Preliminary Safety Analysis

The principles of a safety case were introduced in section 2.3, where the process of a phased safety case was described, together with examples of milestones associated with the different phases.
A phased safety case is especially used when developing a novel system or using a novel safety argument [20]. The Autonomous Combine Harvester system is novel technology currently being researched by multiple companies, such as AGCO and John Deere. The development of this system is a large ongoing project, where many uncertainties can occur in the design and ensuring of the system safety. By using a phased safety case development process, the safety can be included in the design from the beginning, and therefore aid in reducing the amount of redesigns further in the project.
The first phase, in a phased safety case development process, consists of developing a Preliminary Safety Case. This is usually developed simultaneously with or after the requirements, conceptual architecture and operational definition of the system is being developed [17, 21]. The Preliminary safety case of the system will be described in the following sections, with a structure adopted from [21, 19].

A.2 System Description

This section will give a high-level system description and describe the operating environment. The overall goal of the Autonomous Combine Harvester project is to fully automate the agricultural vehicle, as described in section 1.2.1.
The main components of a combine harvester is briefly explained in this section, followed by an overview of the functionality and operating environment of the envisioned autonomous combine harvester. It is important to note that an overall design of the Autonomous Combine Harvester is not created at this point.

A.2.0.1 System Components

The physical components in a combine harvester are labelled in figure A.1. The explanations of each of these components are below [51, 52]:

1. **Reel:** used to hold the crops, so that they are positioned vertically while being cut.
2. **Cutter bar:** used to cut/remove the stalks of the crop from the grain and chaff. Different designs of cutter bars are used for different types of crops.
3. **Header auger:** is a cylinder with spiral blades, that is used to move the cut crops towards the grain conveyor.
4. **Grain conveyor:** transports the crop further on in the combine harvester.
5. **Stone trap:** prevents foreign objects, especially stones from entering the *threshing*$^{\tau}$ drum.
6. **Threshing drum:** bashes the crop to separate the grain and chaff from each other and separate the grain and chaff from the straw.
7. **Concave:** contains the straw after the threshing drum has separated it from the grain and chaff.
8. **Straw walker:** transports the straw towards the straw chopper, while left over grain and chaff falls through down to the sieves.
9. **Grain pan:** contains the grain and chaff after passing through the threshing drum.
10. **Fan:** used to separate the grain from the chaff, by blowing the chaff away from the grain. This can be done because the chaff weighs much less than the grain.
11. **Top adjustable sieve:** used for filtering the grain from left over chaff and straw. The sieve performs a reciprocating motion bringing the grain down to the bottom sieve, and leaving the chaff and straw on the top sieve.
12. **Bottom sieve:** used for filtering the grain, before it moves up to the grain tank. The unfiltered grain (tailings) will be moved back to the threshing drum to perform a rethreshing of the grain.
13. **Tailings conveyor:** moves the tailings back to the threshing drum.
14. **Rethreshing of tailings:** the tailings are rethreshed starting at the threshing drum.
15. **Grain auger:** is a cylinder with spiral blades, used to move the grain to the grain tank.
16. **Grain tank:** the tank containing the harvested grain.
17. **Straw chopper:** used to chop the straw before spreading it onto the field.

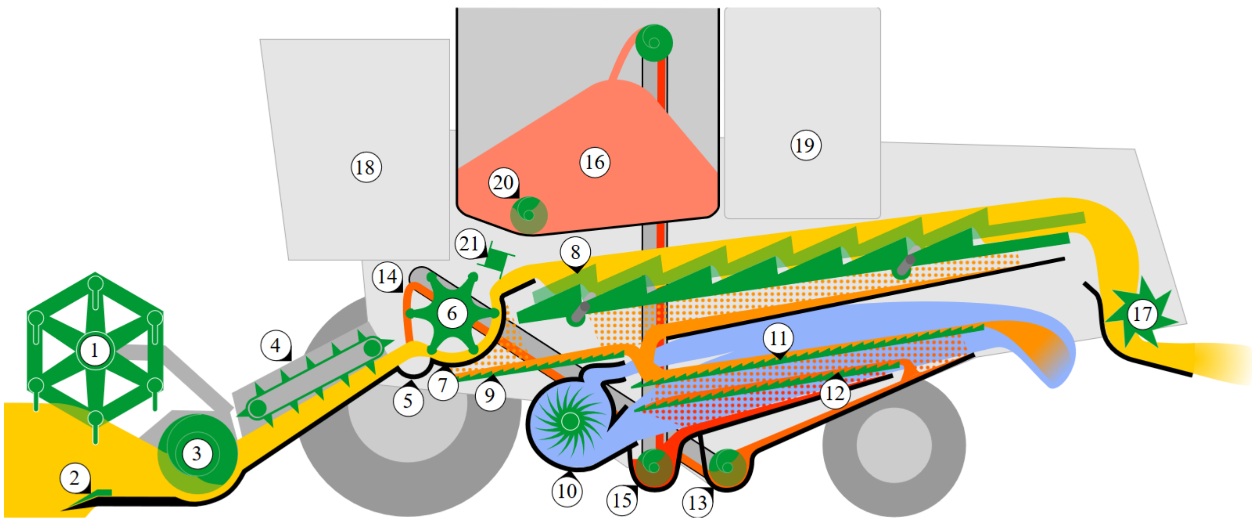


Figure A.1: Structure of a combine harvester. *(By Hans Wastlhuber & Tucvbif, CC BY-SA 2.5, https://commons.wikimedia.org/w/index.php?curid=7568735)*

18. **Driver's cab:** the box where the driver sits.
19. **Engine:** the engine of the combine harvester.
20. **Unloading auger:** used to take the grain out of the grain tank.
21. **Impeller:** moves straw towards straw walkers.

A.2.1 Functionality

The key functions of a combine harvester are the following:

Reaping $^{\tau}$**:** is the process of cutting and gathering ripe crops.

Threshing: is the process of removing the grain from the chaff. This is typically carried out by bashing the grain and chaff using a threshing drum, which loosens the grain from the chaff.

Winnowing $^{\tau}$**:** is the process of separating the grain from the chaff. This is typically accomplished by blowing the lightweight chaff away from the grains.

A typical combine harvester will perform these three actions autonomously, while a human operator is sitting in the driver's cab and controlling the vehicle.
The task of the ACH is to be able to perform these three functionalities while driving in the crop field autonomously, and harvesting chosen crops. There are a lot of system parts that need to collaborate to create a fully autonomous combine harvester. Examples of parts that need to be developed are autonomous path planning, autonomous control, autonomous detection of obstacles and more.

A.2.2 Operating Environment

The amount of harvesting periods per year differs in every country. In Denmark the harvesting season is typically between July and August, while in Indonesia the harvesting season is more spread out during the year, and typically in February-March, July-August and November-December [53]. This means the system must be able to operate in all types of weather, humid, cold, hot, etc.

A.3 Scope of Standards

The ISO standard 18497: *Agricultural machinery and tractors - Safety of highly automated agricultural machines - Principles for design* [54] must be addressed during the development of the system. This standard has its main focus on defining standards for the safety aspects of an automated agricultural vehicle, e.g. safety distances. It briefly describes the possible hazards that can occur during system operation, safety requirements and methods for verification and validation of the safety requirements and risk reduction measures.
This standard gives insight into possible hazards and safety requirements for autonomous agricultural vehicles operating in fields. It does not contain a detailed safety description of specifically a combine harvester, and therefore other relevant standards may be used to extend this.
The ISO standard 4254-7: *Agricultural machinery - Safety - Combine harvesters, forage harvesters, cotton harvesters and sugar cane harvesters* [55] may be used as an extension to the ISO 18497 standard. However this standard is not targeted to autonomous combine harvesters, but it can be used when defining the detailed safety requirements and hazards of an operating combine harvester.
Another relevant standard is the ISO standard 26262: *Road Vehicles - Functional Safety* [4]. This standard has its focus on the safety of road vehicles, such as cars. Although this standard does not focus on the functional safety of agricultural vehicles directly, it may be used as background knowledge.

A.4 Risk assessment

The purpose of performing a risk assessment of hazards was described in section 3.4. A risk assessment was performed on each of the five hazards:

H1: Collision with object

H2: Damage to unharvested crops

H3: Fire ignition

H4: Damage to harvesting machinery

H5: Contamination of harvested crops

Table A.1 describes the reasons why the specific hazard probabilities were chosen for each hazard. These reasons were based on a simple domain analysis, but in reality domain experts should be included in this process, in order to make the best decisions when determining the risk of each hazard.

Identified Hazard	**Hazard probability**	**Argumentation**
H1	Medium	There are many factors that may result in this hazard occurring. Many of these factors are related to the environment that the system will be operating in, meaning it is not possible to control them. The object detection subsystem will be designed to fuse the data from the various sensors used, and therefore decrease the probability of an error occurring from one sensor. Therefore the probability of this hazard is chosen to be medium. There may be unforeseen events in the environment that may lead the hazard occurring.
H2	Low	Most of the events that lead to this hazard are controllable and unlikely to occur. Therefore the hazard probability is chosen to be low.
H3	Low	The events that lead to this hazard are unlikely to occur, and therefore the probability has been decided to be low.
H4	Low	The events that may lead to this hazard are associated with H1 and H3. The probability of the system colliding with an object is medium, but it is determined that the probability of the system colliding with an object that will damage the harvesting machinery is very low. The probability of the machine igniting is also low. The overall probability of this hazard occurring is therefore determined to be low.
H5	Low	The probability of the events that may lead to this hazard occurring are low, and therefore the probability is set to be low.

Table A.1: Description of the arguments for why the hazard probability was chosen as it was.

Table A.2 describes the arguments for the choice of accident severity.

Identified Hazard	**Accident severity**	**Argumentation**
H1	High	A combine harvester is large and heavy, it can weigh around 16 tons. If such a vehicle collides with a living creature, a car, or any other property, then it will damage that object. The worst case would be costing human lives.
H2	Medium	Damaging of unharvested crops can vary in accident severity depending on the amount of unharvested crops that are damaged.
H3	High	If a fire starts in the vehicle, the consequences may be severe. The worst case scenario would be, that the vehicle itself is damaged, and the dry crops on the field burn down.
H4	Medium	This depends on the amount of damage done to the vehicle.
H5	Low	A combine harvester can by itself transport a smaller amount of harvested crops, and therefore they will typically drive with a tractor and a wagon besides them. The amount of harvested crops that can be damaged is not that high, and therefore the severity is set to be low.

Table A.2: Description of the arguments for why the accident severity was chosen as it was.

A.5 Safety Requirements

The top-level safety requirements of the system are derived by studying the standards described in section A.3, and the hazard analysis of the system described in appendix B [21, 16]. The main safety requirements relevant for this thesis project are described below:

- **SR1:** The system shall enter a safe-state if an engine failure occurs [54].
- **SR2:** The system shall enter a safe-state if the communication with the supervisory controller is lost [54].
- **SR3:** An audible alarm shall be initiated and remain on for a specified amount of time, before the engine of the vehicle is turned on [54].
- **SR4:** The operator must be able to stop the system, using a control which is easily accessible [54].
- **SR5:** The system shall detect the length of the header attached to the vehicle, and check if it corresponds to the chosen header by the operator.
- **SR6:** The system shall be able to detect and maintain operation during the failure of one sensor.

- **SR7:** The system shall be able to detect the failure of the lighting mounted on the vehicle during the dark.
- **SR8:** The system shall be able to detect if the sensor range is decreased due to environmental conditions, such as fog.
- **SR9:** The system shall be able to brake in time to avoid collision with an obstacle, in normal operating conditions, and also in foggy surroundings.

A.6 Goal Structuring Notation

Figures A.2 and A.3 together represent the GSN for avoiding collision with obstacles. Table A.3 describes each of the components shown in the two GSN figures.

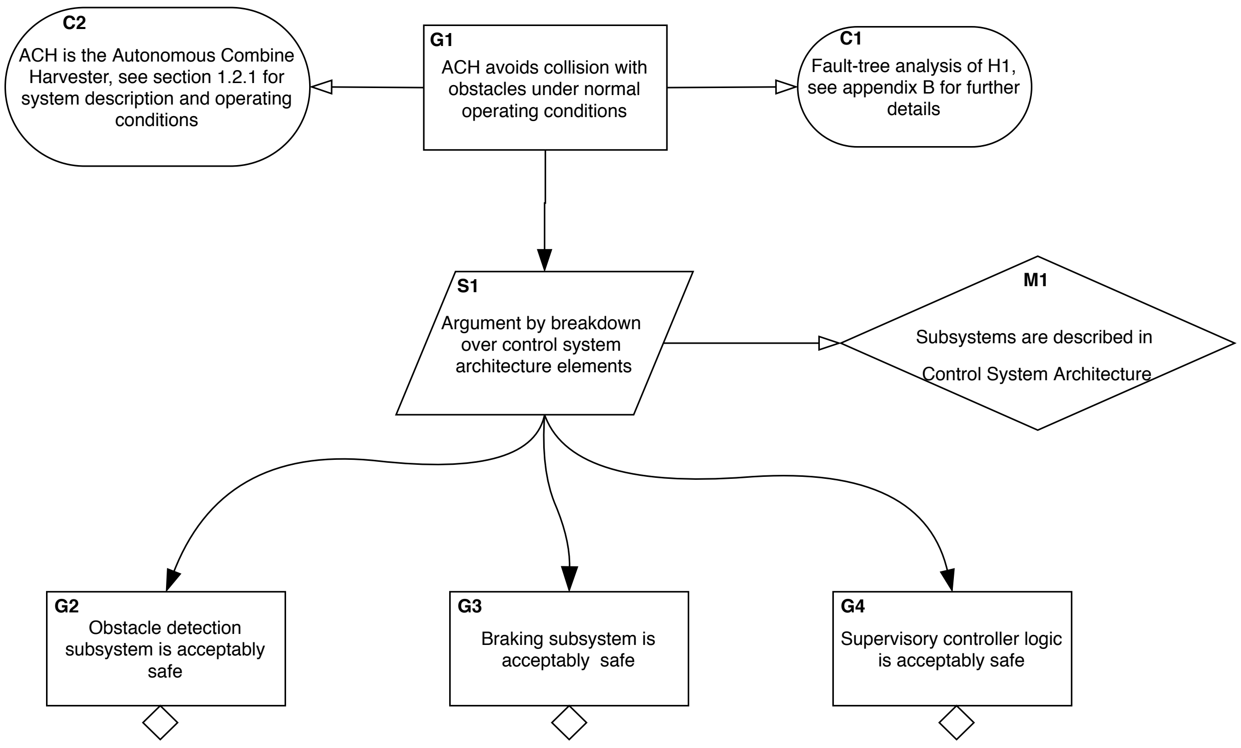

C2
ACH is the Autonomous Combine Harvester, see section 1.2.1 for system description and operating conditions
G1
ACH avoids collision with obstacles under normal operating conditions
C1
Fault-tree analysis of H1, see appendix B for further details
S1
Argument by breakdown over control system architecture elements
M1
Subsystems are described in Control System Architecture
G2
Obstacle detection subsystem is acceptably safe
G3
Braking subsystem is acceptably safe
G4
Supervisory controller logic is acceptably safe

Figure A.2: The GSN diagram with the top-level claim "The Autonomous Combine Harvester avoids collision with obstacles under normal operating conditions", with details of S1 continuing the diagram in figure 3.6.

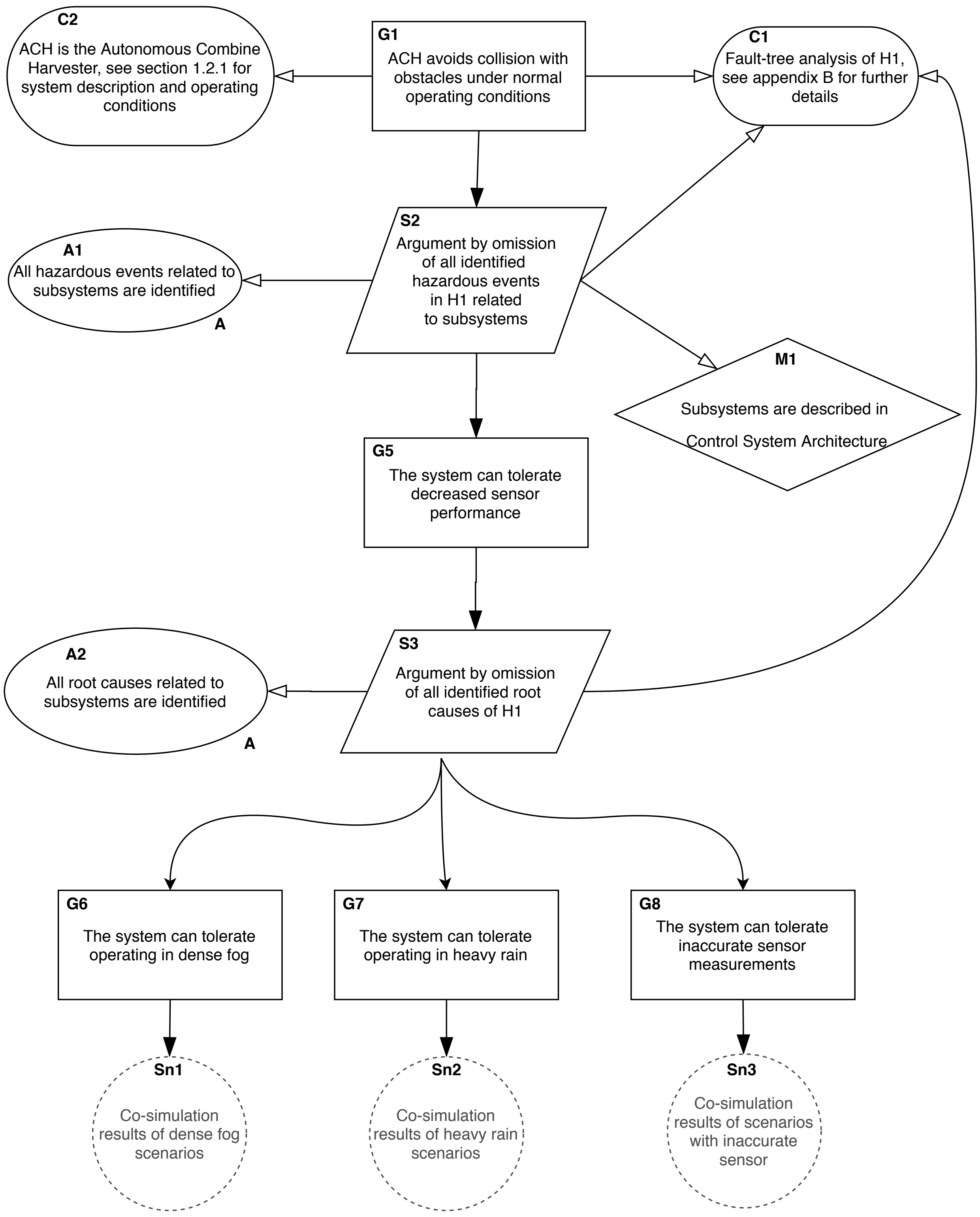

C2
ACH is the Autonomous Combine Harvester, see section 1.2.1 for system description and operating conditions
G1
ACH avoids collision with obstacles under normal operating conditions
C1
Fault-tree analysis of H1, see appendix B for further details
S2
Argument by omission of all identified hazardous events in H1 related to subsystems
A1
All hazardous events related to subsystems are identified
A
M1
Subsystems are described in Control System Architecture
G5
The system can tolerate decreased sensor performance
S3
Argument by omission of all identified root causes of H1
A2
All root causes related to subsystems are identified
A
G6
The system can tolerate operating in dense fog
G7
The system can tolerate operating in heavy rain
G8
The system can tolerate inaccurate sensor measurements
Sn1
Co-simulation results of dense fog scenarios
Sn2
Co-simulation results of heavy rain scenarios
Sn3
Co-simulation results of scenarios with inaccurate sensor

Figure A.3: The GSN diagram with the top-level claim "The Autonomous Combine Harvester avoids collision with obstacles under normal operating conditions", with details of S2 continuing the diagram in figure 3.6.

ID	**Description**
G1	The top-level goal of the GSN, which was arisen from C1, and C2 contains information about the system operating environment.
C1	The fault-tree analysis of *Hazard 1: Collision with object.*
C2	The system description introduced in the early phases of the safety case.
S1	The strategy adopted to fulfill the claim. The argument will be constructed by breaking down the safety goal into safety goals of subsystems. This will be done using the model of the individual elements shown in M1.
M1	A model of the control architecture system, showing the relationship between the subsystems supervisory control logic, object detection, and braking subsystem.
G2	The safety goal of the object detection subsystem, which must be satisfied by supporting evidence.
G3	The safety goal of the braking subsystem, which must be satisfied by supporting evidence.
G4	The safety goal of the supervisory control subsystem which must be satisfied by supporting evidence.
S2	The strategy that will be used to prove the hazardous events that may lead to H1 are mitigated.
S3	The strategy that will be used to prove the root causes that may lead to H1 are mitigated.
A1	The assumption that all the hazardous events related to the system are identified.
A2	The assumption that all the root causes related to the subsystems are identified.
G5	The safety goal illustrating the system can operate safely in conditions with decreased sensor performance.
G6	The safety goal related to the event of the system operating in dense fog conditions.
G7	The safety goal related to the event of the system operating in heavy rain conditions.
G8	The safety goal related to the event where inaccurate sensor measurement occur.
Sn1	The evidence that supports the safety goals G6. The evidence will be constructed using co-simulation results of relevant scenarios.
Sn2	The evidence that supports the safety goals G7. The evidence will be constructed using co-simulation results of relevant scenarios.
Sn3	The evidence that supports the safety goals G8. The evidence will be constructed using co-simulation results of relevant scenarios.

Table A.3: Descriptions of the elements in the Goal Structuring Notation in figures A.2 and A.3.

Appendix B

Hazard Analysis

B.1 Introduction

This chapter describes the hazards defined in section 3.3 in detail, where the root causes of each hazard will are presented. A fault tree is created for each hazard. The fault trees are split up into multiple figures, due to the size of the tree. The figures are connected to each other using dotted arrows, as can be seen in figure B.1, where the root causes of events A and B are continued in following figures.

B.2 Hazard 1: Collision With Object

This section describes the hazard analysis for the system colliding with an object. This hazard describes the system colliding with an object. The object can for example be a living animal, another machine in the field, trees or a hedge.

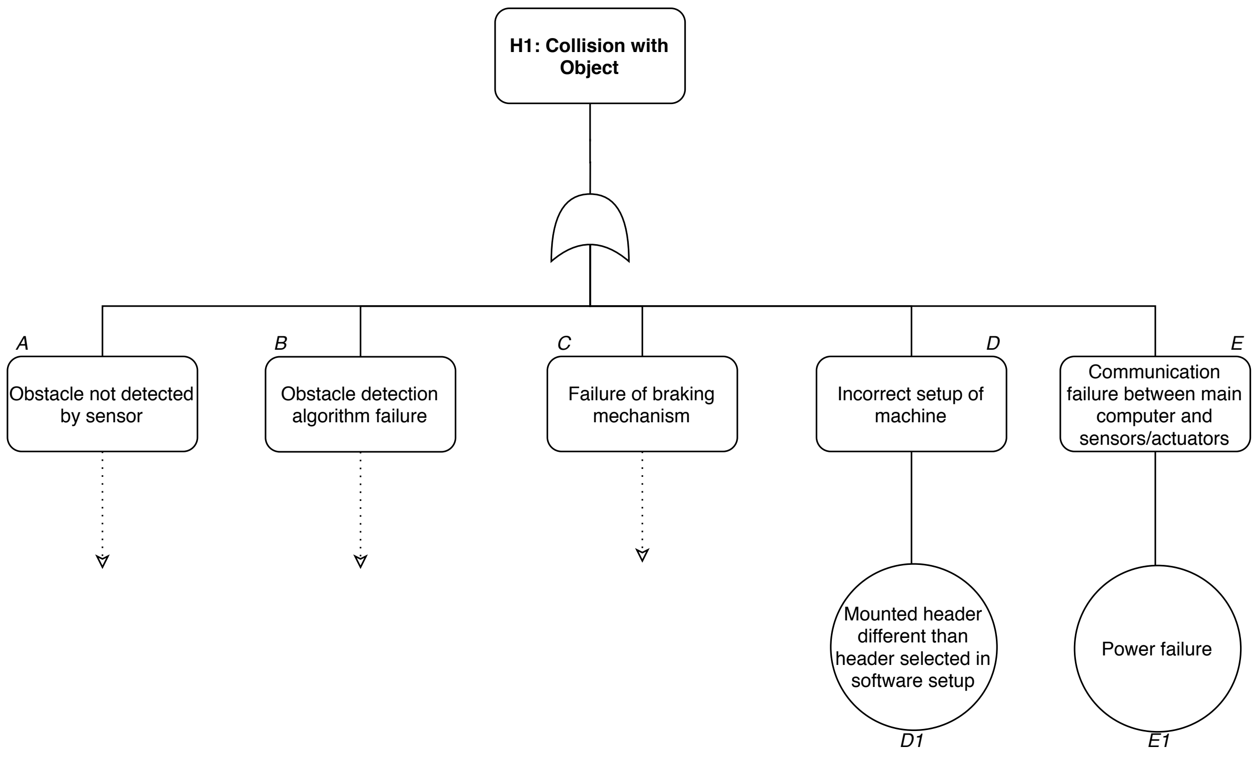

H1: Collision with Object
A
Obstacle not detected by sensor
B
Obstacle detection algorithm failure
C
Failure of braking mechanism
D
Incorrect setup of machine
E
Communication failure between main computer and sensors/actuators
Mounted header different than header selected in software setup
D1
Power failure
E1

Figure B.1: First part of the fault tree of *Hazard 1 - Collision with Object*.

Event ID	**Event**	**Description**
A	Obstacle not detected by sensor.	The sensor(s) does not measure the obstacle.
B	Obstacle detection algorithm failure.	The obstacle detection algorithm fails at detecting a present obstacle.
C	Failure of braking mechanism.	The vehicle fails to brake in time, and therefore collides with object.
D	Incorrect setup of machine.	Fault due to human error, where the setup of the machine is done incorrectly.
D1	Mounted header different than header specified in software setup.	A possible root cause of event *C*. It describes the event where a human operator incorrectly sets up a different header in the user interface, than the actual header mounted on the machine.
E	Communication failure between main computer and sensors/actuators.	The communication between the main computer that is processing the data from the sensors and controlling the actuators fails.
E1	Power failure.	A possible root cause of event *D*. If the power supply fails.

Table B.1: Descriptions of the events in the fault tree in figure B.1.

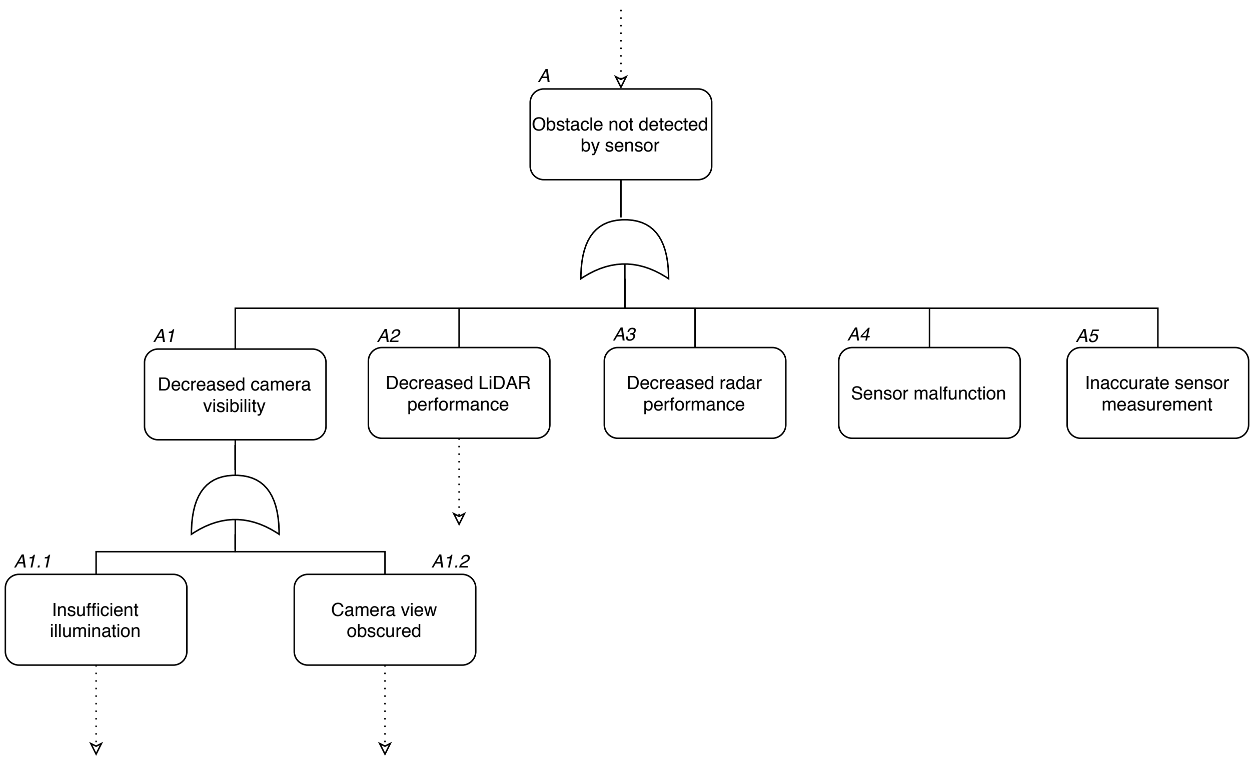

A
Obstacle not detected by sensor
A1
Decreased camera visibility
A2
Decreased LiDAR performance
A3
Decreased radar performance
A4
Sensor malfunction
A5
Inaccurate sensor measurement
A1.1
Insufficient illumination
A1.2
Camera view obscured

Figure B.2: Second part of the fault tree for *Hazard 1 - Collision with Object*, that is shown in figure B.1. This fault tree shows the main causes for the *A: Obstacle not measured by sensor*.

Event ID	**Event**	**Description**
A1	Decreased camera visibility.	The visibility of the camera is too low to be able to properly detect obstacles.
A1.1	Insufficient illumination.	One of the possible causes of event *A1*. If the illumination of the surroundings is too low.
A1.2	Camera view obscured.	One of the possible causes of event *A1*. The obscurant may: - Blur the edges of the obstacle. - Decrease the visibility of the obstacle from the vehicle's point of view.
A2	Decreased LiDAR performance.	If the LiDAR performance, i.e. the range or accuracy, is decreased due to environmental conditions.
A3	Decreased radar performance.	If the radar performance is decreased.
A4	Sensor malfunction.	If the sensor(s) stop functioning as intended.
A5	Inaccurate sensor measurement.	Sensors measurements may drift over time. Some sensors have a minimum detection range when mounted on a dynamic vehicle [34, 56, 57], which may in some cases result in inaccurate sensor measurements.

Table B.2: Descriptions of the events in the fault tree in figure B.2.

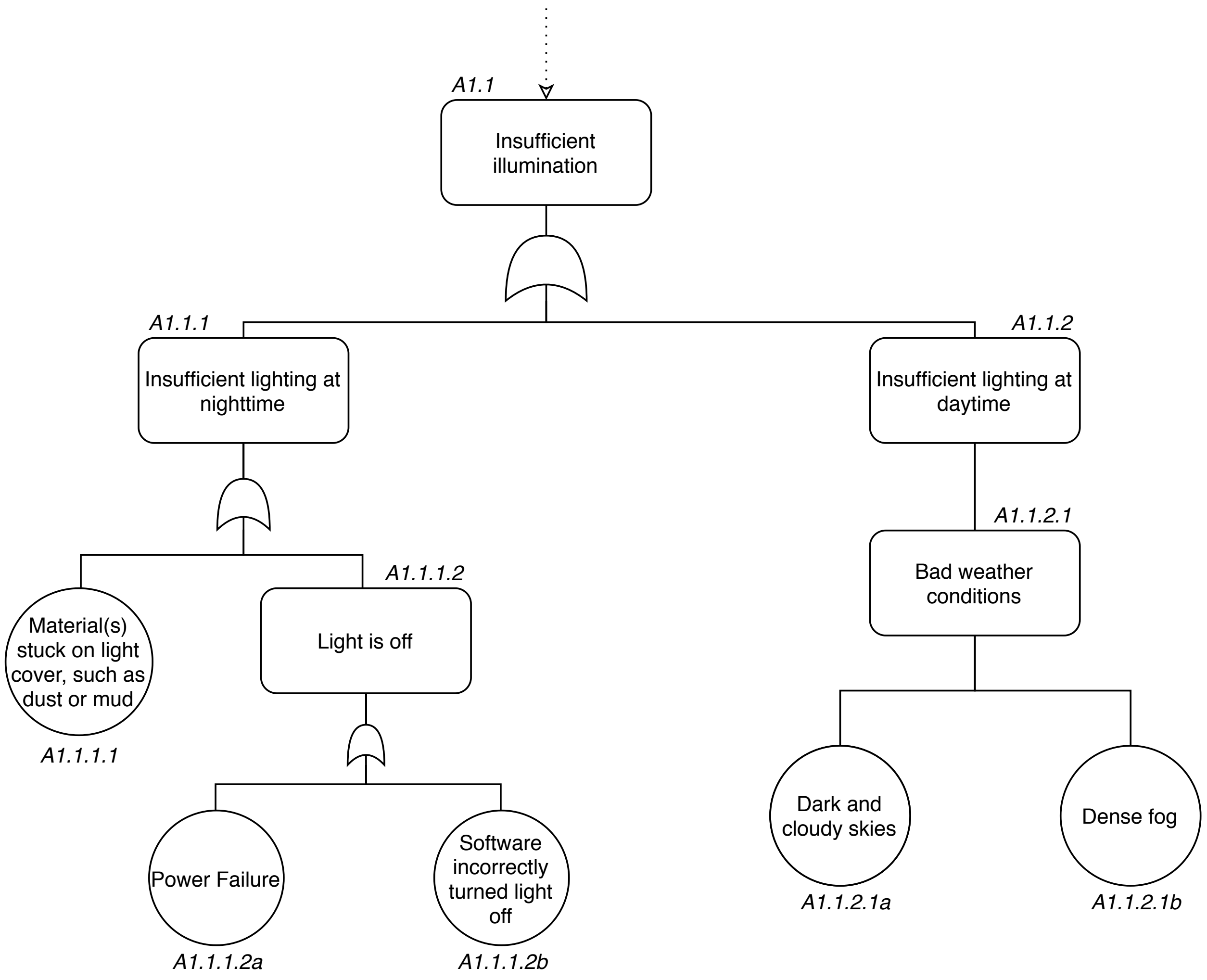

A1.1
Insufficient illumination
A1.1.1
Insufficient lighting at nighttime
A1.1.2
Insufficient lighting at daytime
A1.1.2.1
Bad weather conditions
A1.1.1.2
Material(s) stuck on light cover, such as dust or mud
Light is off
A1.1.1.1
Dark and cloudy skies
Dense fog
A1.1.2.1a
A1.1.2.1b
Power Failure
Software incorrectly turned light off
A1.1.1.2a
A1.1.1.2b

Figure B.3: This fault tree shows the root causes for *A1.1: Insufficient illumination*, and is a continuation of the fault tree in figure B.2.

Event ID	Event	Description
A1.1.1	Insufficient lighting at nighttime.	This event is one of the possible causes of the event *A1: Decreased camera visibility*. If there is not sufficient lighting in front of the vehicle, then the image quality and/or the visibility of the camera may be decreased.
A1.1.1.1	Material(s) stuck on light cover, such as dust or mud.	It is possible that dust or mud may stick to the lights mounted on the vehicle, therefore covering the lights and decreasing the amount of lighting in front of the vehicle.
A1.1.1.2	Light is off.	If the lights mounted on the vehicle are turned off, there will not be any lighting at all, therefore affecting the camera visibility greatly.
A1.1.1.2a	Power Failure.	This is one of the possible root causes of event *A1.1.1.2*.
A1.1.1.2b	Software incorrectly turned light off.	This is one of the possible root causes of event *A1.1.1.2*. If the software is not implemented correctly, it may turn off the lights even though they should be turned on.
A1.1.2	Insufficient lighting at daytime.	This event is one of the possible causes of the event *A1: Decreased camera visibility*.
A1.1.2.1	Bad weather conditions.	This is a possible cause of the event *A1.1.2*. If the weather conditions are bad, they may lead to decreased illumination in front of the vehicle.
A1.1.2.1a	Dark and cloudy skies.	This is a possible root cause of the event *A1.1.2*. During daytime there will be light from the sun, but if the sky is cloudy and dark, then the lighting from the sun will be prevented from fully illuminating the environment in front of the vehicle.
A1.1.2.1b	Dense fog.	A dense fog around the vehicle and camera will decrease the illumination and therefore decrease the visibility of the camera [28].

Table B.3: Descriptions of the events in the fault tree in figure B.3.

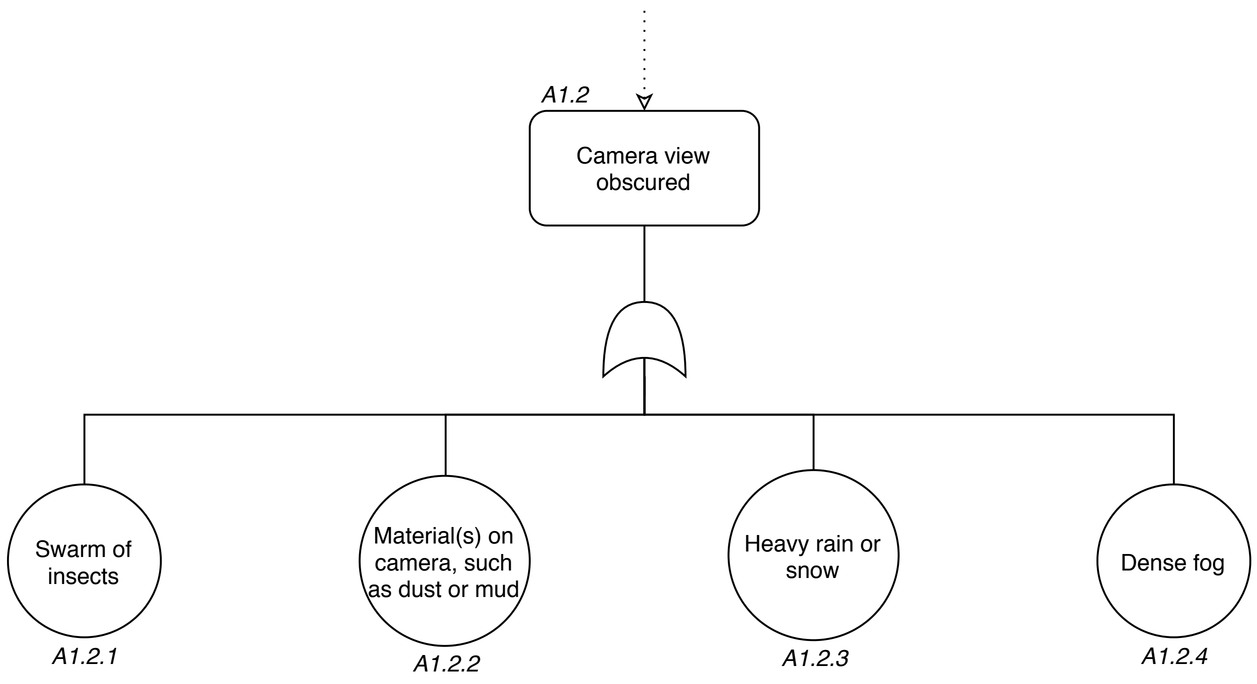


Figure B.4: This fault tree shows the root causes for *A1.2: Camera view obscured*, and is a continuation of the fault tree in figure B.2.

Event ID	**Event**	**Description**
A1.2.1	Swarm of insects.	During nighttime, the lights mounted on the machine will be turned on, in order for the camera to capture images and process them. Insects are typically attracted to light, and therefore it is possible that there will be swarms of insects obscuring parts of the view of the camera.
A1.2.2	Material(s) on camera such as dust or mud.	When the vehicle is driving in the field, materials such as dust or mud may stick to the camera, obscuring parts of it.
A1.2.3	Heavy rain or snow.	If it starts raining or snowing heavily, the rain/snow will be in between the line of sight from the camera to an obstacle, making it difficult to detect objects.
A1.2.4	Dense fog.	A dense fog in the field will decrease the visibility of the camera by decreasing the visible range of the camera [28].

Table B.4: Descriptions of the events in the fault tree in figure B.4.

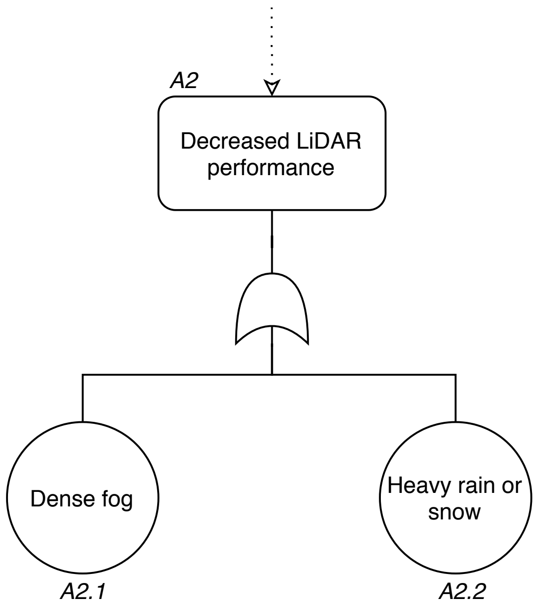


Figure B.5: This fault tree shows the root causes for *A2: Decreased LiDAR performance*, and is a continuation of the fault tree in figure B.2.

Event ID	**Event**	**Description**
A2.1	Dense fog.	Foggy surroundings may decrease the performance of LiDARs [26], leading to a lower accuracy or range.
A2.2	Heavy rain or snow.	Rain or snow may decrease the performance of LiDARs [27], leading to a lower accuracy or range.

Table B.5: Descriptions of the events in the fault tree in figure B.5.

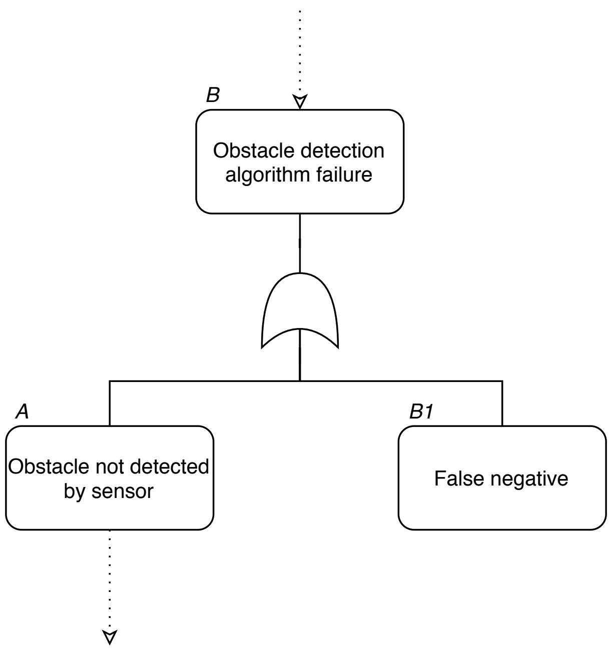

B
Obstacle detection algorithm failure
A
Obstacle not detected by sensor
B1
False negative

Figure B.6: This fault tree shows the root causes for *B: Obstacle detection algorithm malfunction*, and is a continuation of the fault tree in figure B.1.

Event ID	**Event**	**Description**
A	Obstacle not detected by sensor.	This is described in table B.1.
B1	False negative.	The obstacle detection algorithm may incorrectly misclassify an obstacle.

Table B.6: Descriptions of the events in the fault tree in figure B.6.

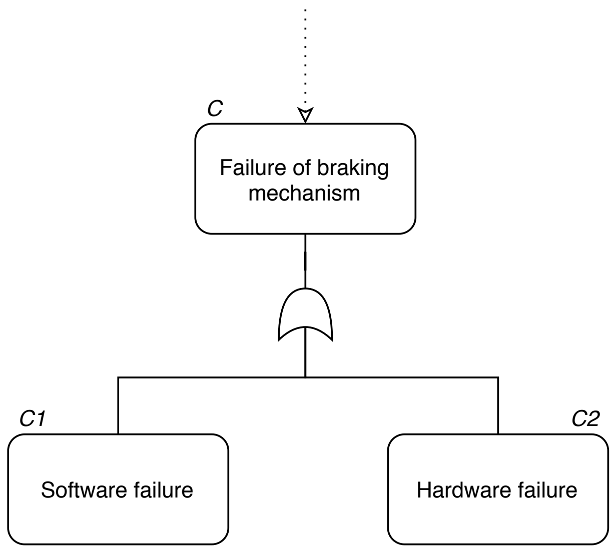


Figure B.7: This fault tree shows the root causes for *C: Failure of braking mechanism*, and is a continuation of the fault tree in figure B.1.

Event ID	**Event**	**Description**
C	Failure of braking mechanism.	This is described in table B.1.
C1	Software failure.	Failure of the software that controls the brakes in the vehicle.
C2	Hardware failure.	Failure of the hardware used for braking in the vehicle.

Table B.7: Descriptions of the events in the fault tree in figure B.7.

B.3 Hazard 2: Damage to Unharvested Crops

This hazard describes the system damaging other crops, which are not ready to harvest. Figure B.8 shows the fault tree of the hazard, and table B.8 describes the components in the fault tree.

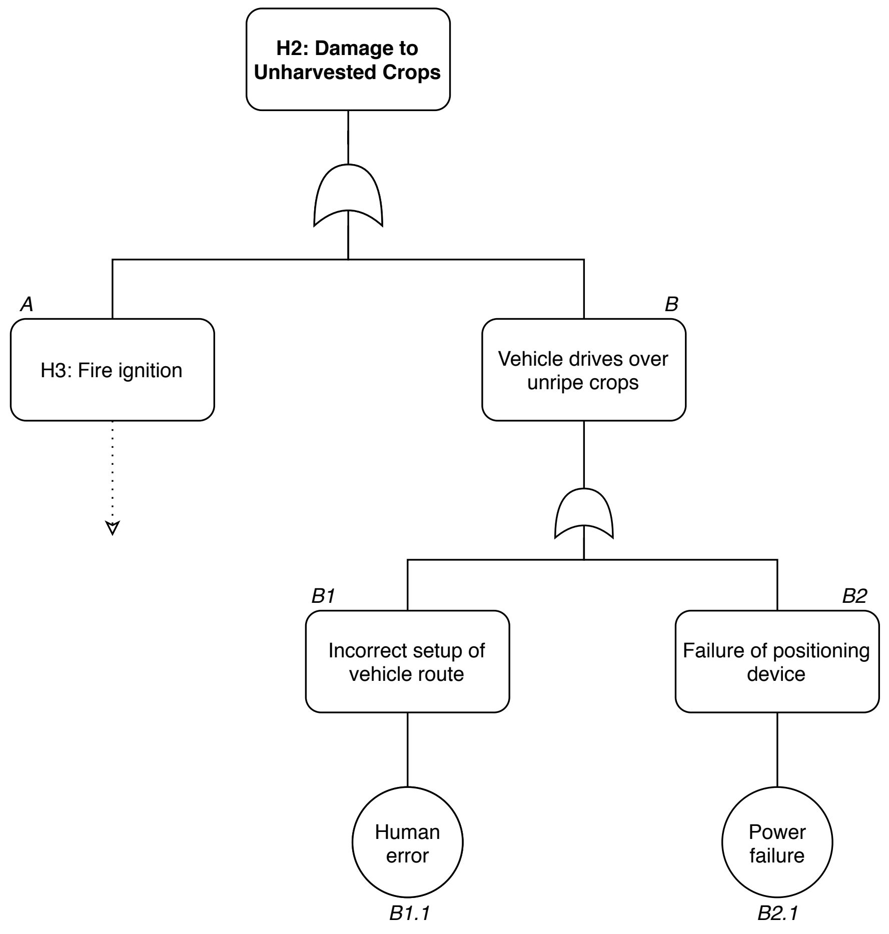

H2: Damage to Unharvested Crops
A
H3: Fire ignition
B
Vehicle drives over unripe crops
B1
Incorrect setup of vehicle route
B2
Failure of positioning device
Human error
B1.1
Power failure
B2.1

Figure B.8: Fault tree of *Hazard 2 - Damage to Unharvested Crops*.

Event ID	**Event**	**Description**
A	H3: Fire ignition.	One of the causes of the vehicle to damage unharvested crops, is if a fire is started in the vehicle. This may spread to the other fields in the farm, and therefore damage unharvested crops.
B	Vehicle drives over unripe crops.	If the vehicle incorrectly drives over crops that are not ready to be harvested.
B1	Incorrect setup of vehicle route.	If the route the vehicle must drive is set up incorrectly, so the vehicle drives over unripe crops that aren't meant to be harvested.
B1.1	Human error	The root cause of this may be a human incorrectly setting the route. This could for example be if the operator forgets to update the route of the vehicle, leading the vehicle to drive an old route.
B2	Failure of positioning device	The device used to retrieve the vehicle's position fails.
B2.1	Power failure	A root cause for the positioning device to fail, is a power failure.

Table B.8: Descriptions of the events in the fault tree in figure B.8.

B.4 Hazard 3: Fire Ignition

This hazard describes different scenarios where fire may be ignited in the machine. Figure B.9 shows the fault tree of the hazard, and table B.9 describes the fault tree components.

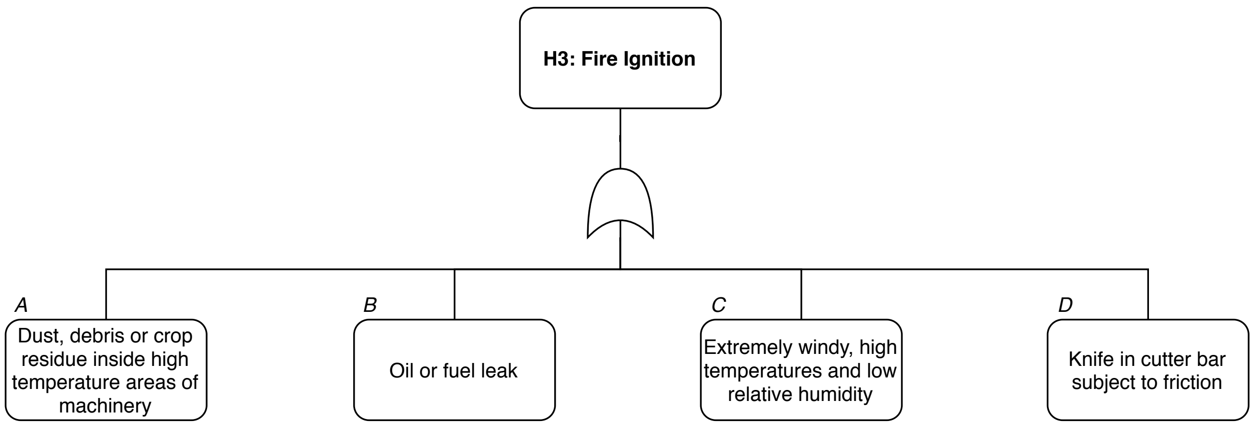

H3: Fire Ignition
A
Dust, debris or crop residue inside high temperature areas of machinery
B
Oil or fuel leak
C
Extremely windy, high temperatures and low relative humidity
D
Knife in cutter bar subject to friction

Figure B.9: Overview of the fault tree of *Hazard 3 - Fire Ignition*.

Event ID	**Event**	**Description**
A	Dust, debris or crop residue inside high temperature areas of machinery.	The combinations of these types of materials accumulating inside the machine, near the engine or areas with very high temperatures may lead to a fire [58].
B	Oil or fuel leak.	An oil or fuel leak near the engine may lead to a fire, if the temperature of the engine is high enough.
C	Extremely windy, high temperature and low relative humidity.	In certain environmental conditions, there may be a high risk of a fire starting, if it is windy, high temperature and a low relative humidity [59].
D	Knife in cutter bar subject to friction.	The reason for friction may be if the knives are poorly maintained. If a knife in the cutter bar is subject to enough friction, the temperature of the metal may increase, and if in contact with dry grain, it may lead to a fire [58].

Table B.9: Descriptions of the events in the fault tree in figure B.9.

B.5 Hazard 4: Damage to Harvesting Machinery

This hazard describes the different objects, or conditions that may lead to damage of the harvesting machinery. Figure B.10 shows the fault tree of the hazard, and table B.10 describes the fault tree components.

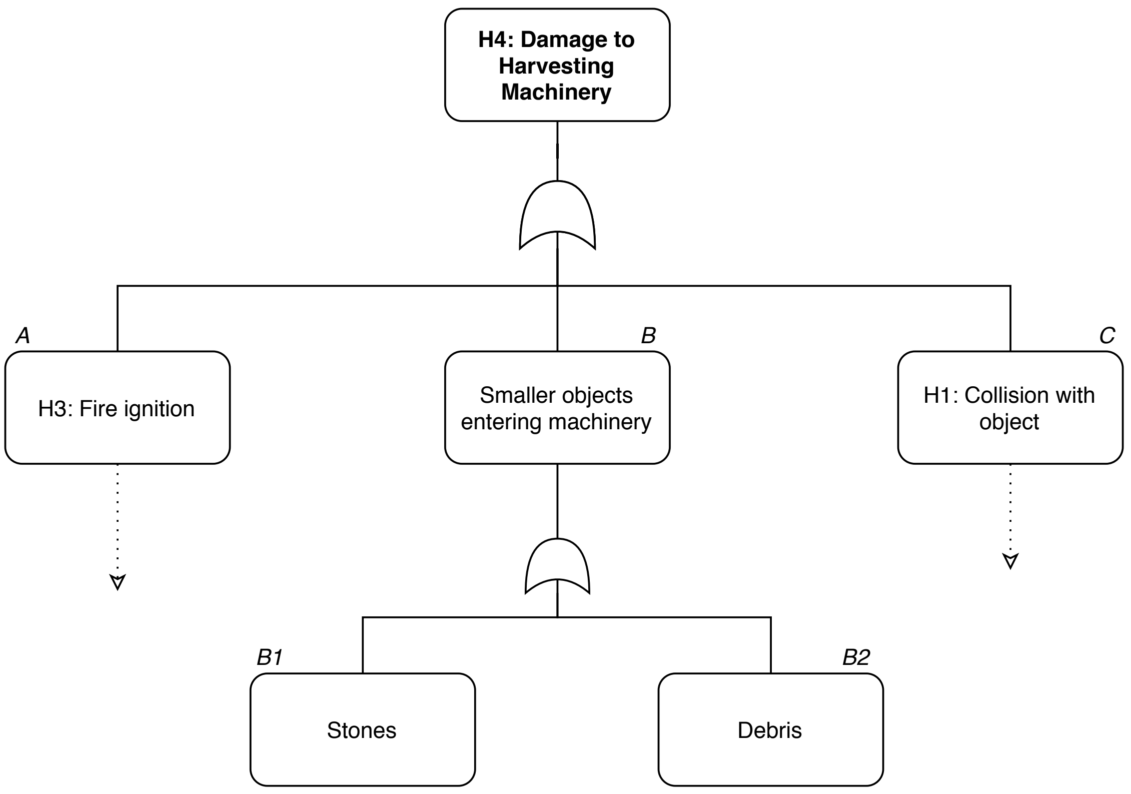

H4: Damage to Harvesting Machinery
A
H3: Fire ignition
B
Smaller objects entering machinery
C
H1: Collision with object
B1
Stones
B2
Debris

Figure B.10: Overview of the fault tree of *Hazard 4 - Damage to Harvesting Machinery*.

Event ID	Event	Description
A	H3: Fire ignition.	This refers to hazard 3. The description of this hazard and it's root causes are described in table B.9. If a fire is ignited in the machine, for example near the engine, then the vehicle will be damaged.
B	Smaller objects entering machinery.	If small objects enter the machinery, they may damage parts of the machine.
B1	Stones.	A stone trap exists in each combine harvester nowadays, but it is still possible that some stones do not get caught by the stone trap and move on through the vehicle.
B2	Debris.	If debris enters the vehicle it may damage parts of it.
C	H1: Collision with object.	If the vehicle collides with a large or heavy object, it may damage the vehicle. A more detailed description of this hazard can be found in table B.1.

Table B.10: Descriptions of the events in the fault tree in figure B.10.

B.6 Hazard 5: Contamination of Harvested Crops

This hazard describes the conditions that may lead to contamination of harvested crops. Figure B.11 shows the fault tree of the hazard, and table B.11 describes the fault tree components.

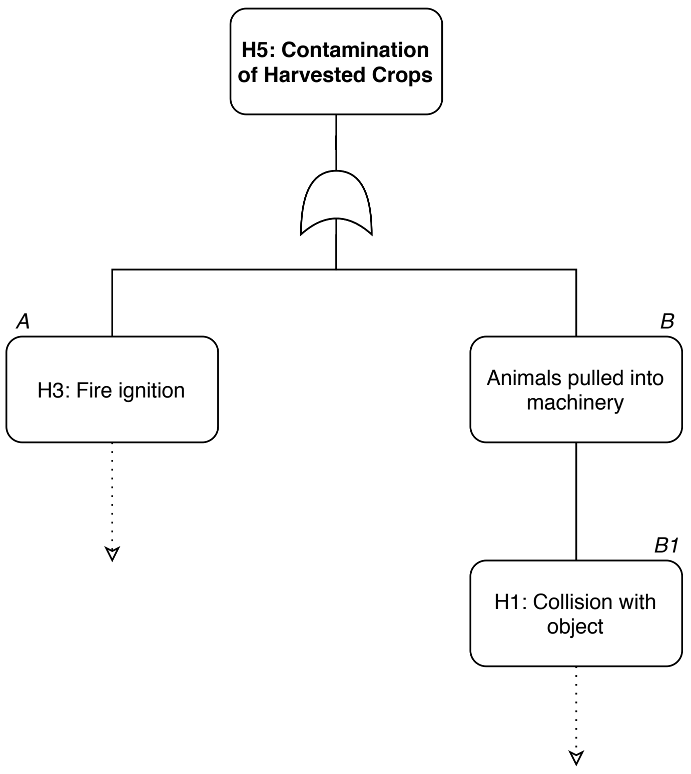

H5: Contamination of Harvested Crops
A
H3: Fire ignition
B
Animals pulled into machinery
B1
H1: Collision with object

Figure B.11: Overview of the fault tree of *Hazard 5- Contamination of Harvested Crops.*

Event ID	**Event**	**Description**
A	H3: Fire ignition.	If the vehicle ignites and burns the harvested crops. This refers to hazard 3. The description of this hazard and it's root causes are described in table B.9.
B	Animals pulled into machinery.	It is common that animals will lay on the field, where it may sometimes be difficult to detect these animals. If the animal gets into the machinery, where their remainings will contaminate the harvested crops.
B1	H1: Collision with object.	A reason why the vehicle drives into an animal. This refers to hazard 1. The description of this hazard and it's root causes are described in table B.1.

Table B.11: Descriptions of the events in the fault tree in figure B.11.

Appendix C

Modelling and Co-simulation

C.1 Modelling Tools

The specific modelling tools that have been used for simulation in this project are described in the following sections.

C.1.1 20-Sim

The tool used for modelling and simulating the dynamics of the vehicle is called *20-sim*[1]. *20-sim* is a program that allows modelling mechatronic systems graphically, using equations, block diagram or physics blocks. The tool contains many elementary models that can be put together to form a larger physical model.

C.1.2 Overture

The tool *Overture* is used for modelling and simulating the discrete-time controller of the system[2]. The language used to realize the controller is VDMRT, which stands for the Vienna Development Method Real-Time, and is a formal modelling language.

C.1.3 PyFMU

The tool *PyFMU* is a Python framework that can be used to create models using Python and export these models to FMUs[3].

C.1.4 INTO-CPS

INTO-CPS stands for *Integrated Toolchain for Cyber-Physical Systems*, which allows collaborative modelling and simulation of cyber-physical systems. The co-simulation engine is FMI 2.0 compliant, making it easy to combine different subsystems into a larger cyber-physical

[1] `https://www.20sim.com/`
[2] `http://overturetool.org/`
[3] `https://github.com/INTO-CPS-Association/pyfmu`

system. INTO-CPS allows performing co-simulations with fixed and variable step sizes, and the ability to perform design-space exploration (DSE). The purpose of performing DSE is to evaluate different designs of a model without the need to physically implement each design in the true system. DSE results are reported in a concise way either using a ranking function or determining the pareto optimal front [60].

C.2 Functional Mock-up Units

This section describes the Functional Mock-Up Units (FMUs) used and developed in this thesis project.

C.2.1 Controller

This FMU was reused from an example project from INTO-CPS[4]. The model is described in the article [61]. The controller is a pure-pursuit controller [62], with a default look-ahead distance of 1 meter.
The purpose of this FMU is to control the vehicle to follow a given route. The route is given as a csv file with x- and y-coordinates that the vehicle must follow. This file must be called "route.csv" and placed in the same folder as the FMU. The vehicle always starts its position at $(0, 0)$. These characteristics of the model were only possible to determine, since the code of model was also given with the FMU of the model. If the code was not given, it would not be possible to change the route of the vehicle, or its starting position. Figure C.1 illustrates which of the relevant characteristics of the Controller model were found in the FMU model description, in this case the speed parameter. The figure also demonstrates the model characteristics that were found by studying the code of the model written in VDMRT, in this case the path file and the start position of the vehicle.

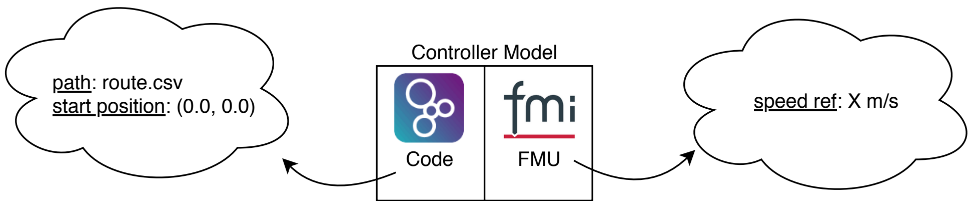


Figure C.1: An illustration of the main characteristics of the Controller model used in this thesis project.

These characteristics of the model defined inside the code, and not included in the FMU parameters, illustrates the challenges of reusing models which are not sufficiently documented.

[4]https://github.com/INTO-CPS-Association/example-autonomous_vehicle

C.2.2 Vehicle

This FMU was reused from an example project from INTO-CPS[5]. The model was developed together with the Controller model and is described in [61].
The model of the vehicle is based on a bicycle model, but with four wheels instead of two. The default weight and dimensions of the vehicle are different than the ones of a combine harvester. It is possible to change these values so the vehicle dynamics are more similar to a combine harvester. The vehicle model is steered on its front wheels, while combine harvesters are steered on their rear wheels.
The weight was changed to 16 tons, and the dimensions were also changed to fit the dimensions of the Fendt Ideal 10T [9], but the experiments with these values didn't show a change in the dynamics of the vehicle. For example the braking distance was observed to be the same. The default values were used when testing the model, since the main purpose was to test the safety of the system regarding the collision avoidance subsystem. It was therefore not necessary for the dynamics of the vehicle to be completely similar to the vehicle dynamics. A simple bicycle model could have also been used in this case.

C.2.3 Environment

This model was implemented in PyFMU. The main purpose of this model is to model the environment of a vehicle. The steps carried out in this model are described below:

1. Upload environment image containing obstacles
2. Detect contours in image
3. Find minimum enclosing circle of contour of each obstacle
4. Define object position and size by center of circle (x,y) and radius (r)

Each of these steps is done using the Python library OpenCV[6]. The first step consists of uploading a plain image which contains coloured objects. Figure C.5 illustrates a possible environment image that can be used. The next step is to detect the contours of each of the objects in the image, followed by finding a minimum enclosing circle of each obstacle. Figure C.2 illustrates the contours of each object, the minimum enclosing circle found, and the center of each circle. The positions and size of each object is defined by the center of the circle, and the radius of the circle. Thus each object id modelled as a circle with a center and radius. The positions and centers of each circle found are the outputs of this FMU.

[5]`https://github.com/INTO-CPS-Association/example-autonomous_vehicle`
[6]`https://opencv.org/`

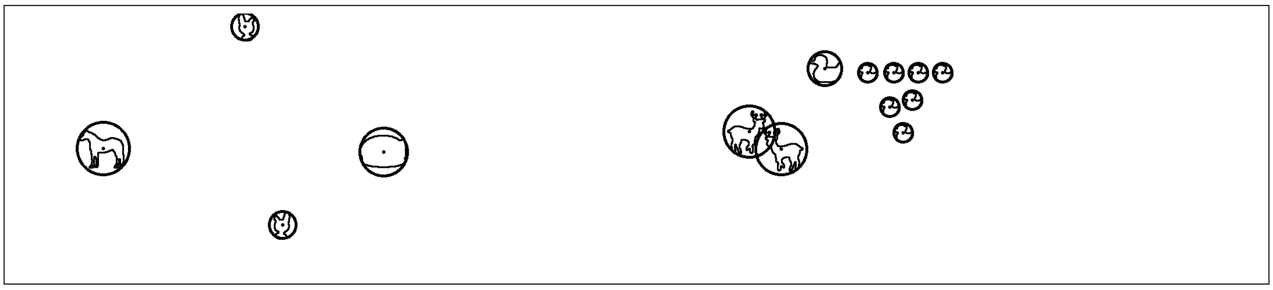


Figure C.2: An illustration of how the environment model finds the contours of each object, then finds the minimum enclosing circle of each of these objects.

The model of the environment is simple, and abstracts the fact that an autonomous vehicle will not be able to determine the complete size of an object, i.e. if viewing an object from one angle, it is not possible to determine how long the object is on the other side.
By representing each object as a circle, the autonomous vehicle will only be able to detect the circular obstacles, meaning that the vehicle may stop before actually being close to the true obstacle. This is chosen an as abstraction to easier model each obstacle and be able to exchange information about obstacles in a more simple manner than defining the complete contour. This also makes it easier to calculate the distance from the vehicle to each object. The inputs of this FMU are an image, and the outputs are the positions and radius of the objects found in the image.
The current limitations of this model are that it only represents static and predefined objects. It would be relevant to extend this model in the future to incorporate dynamic obstacles. This can for example be done using a video and defining a correspondence between time and obstacle position.

C.2.4 Sensor

This model was implemented in PyFMU. The main purpose of this model is to model the object detection system of a vehicle. This model is developed at a high abstraction level, where the results from the fused data of each sensor is modelled. The specific sensors that will be used in the real physical system are yet unknown, but most likely cameras, LiDARs and radars will be used.
In order to model an object detection system which can be adapted to different environmental conditions, two main characteristics were implemented. These characteristics are described below:

Detection range: It is possible to set a minimum range and a maximum range for the sensor to detect an object. This can be used to model a change in the environment, i.e. foggy surroundings. It can also be used to model different sensors with different view ranges.

False positives: It is possible to inject a fault in the sensor, and model false positive objects. It is possible to set the minimum and maximum range in which false positive objects may occur. This is characteristic used to model faults that may occur, where the sensor may incorrectly detect objects. The parameter, mean time to failure, is used to determine the failure rate when generating false positives. The mean time to failure is described in seconds, which is an exaggeration in real life systems. But is set to seconds to model the life time of the system during the co-simulation.

The detection range is set using the minimum and maximum range parameters of the FMU. The distance between the vehicle and object is calculated using formula C.1[7].

$$min\_dist = \left| \sqrt{(x_{point} - x_{circle})^2 + (y_{point} - y_{circle})^2} - r_{circle} \right| \tag{C.1}$$

A comparison is performed checking if the distance is in the range of the sensor. If it is, then the object is added to one of the detected obstacles. The detected obstacles are represented with the center of the circle (x,y) and its radius (r).
The false positives are generated by performing the following steps:

1. Calculate the amount of false positives that occur, by taking a random number from a Poisson distribution [8]. The event rate, λ, is calculated by converting the mean time to failure to a failure rate per second. This is used as the event rate of the Poisson distribution.

2. The position of each false positive is then found by taking random numbers from a Uniform distribution[9].

A Poisson distribution was chosen since the number of false positive events that can occur in a single time frame can vary. It is possible that multiple false positive events occur. It is important to note the assumption made in this case. The assumption is that false positive events occur independently from each other, which is primarily not the case. For example, a false positive event may occur due to reflection of dense fog in the LiDAR or detection of dense fog as an object. In such an event, false positives will in reality occur dependently of each other, where a number of false positives occur on and off in subsequent time frames. The assumption that false positive events are independent was used to create a simple model of false positive occurrences.
A Uniform distribution was chosen, with the assumption that a false positive can occur anywhere around the vehicle.
The inputs of this FMU are the objects in the environment and the current vehicle position. The outputs of this FMU are the detected objects within the sensor range.
Since this model is at a high abstraction level, details such as echos from LiDAR are not modelled, and therefore this model can be used in various systems, which have an object detection subsystem. If more details are needed, then it is possible to model each sensor, and create a model that fuses the data from each of these.

[7]https://www.varsitytutors.com/hotmath/hotmath_help/topics/shortest-distance-between-a-point-and-a-circle

[8]https://www.probabilitycourse.com/chapter3/3_1_5_special_discrete_distr.php

[9]https://www.probabilitycourse.com/chapter4/4_2_1_uniform.php

C.2.5 Monitor

This model was implemented in PyFMU. The main purpose of this model is to perform live plotting of the vehicle driving in the environment with obstacles. The model receives the vehicle position, the environment image, and the positions of the obstacles detected by the sensors. The position of the vehicle is plotted, together with the minimum and maximum range of the sensor, and the objects detected by the sensor.
The live plotting is performed by creating a new thread, and plotting the received data while running on the newly created thread.

C.2.6 Supervisory Controller

This model was implemented in PyFMU. The main purpose of this model is to act as a safety controller, that will stop the vehicle in the case where the system may pose a threat to its surroundings. The main steps carried out in this model are described below:

1. Calculate the braking distance of the vehicle from the current speed.
2. Compare braking distance with maximum sensor range.
3. If braking distance is larger than sensor range, calculate and set new vehicle speed.
4. Calculate distance between vehicle and obstacles detected by sensor FMU. If the obstacle is within the calculated braking distance, perform a safety stop, by setting the vehicle speed to 0 m/s.

The first step consists of calculating the braking distance of the vehicle from the current speed. The formula used to calculated this is simplified with many abstractions, and is defined in equation 4.1. The second step consists of comparing the braking distance and the maximum range of the sensor. In step three, if the braking distance is greater than the maximum sensor range, then a new vehicle speed is calculated and set using equation 4.1. The fourth step consists of calculating the distance from the vehicle to the obstacle, using the formula defined in equation C.1. The braking distance of the vehicle is calculated from the current speed of the vehicle. If the obstacle is within the braking distance calculated, then a safety stop will be performed, by setting the speed of the system to 0 m/s.
The inputs to this FMU are the vehicle position, speed set point from the controller, and the obstacles detected by the sensor. The output of this FMU is the speed set point of the vehicle.

C.2.6.1 Test Results

Table C.1 contains the results of testing different values of the safety margin used. The chosen safety margin value was 0.5 meters, since this was the only value that stopped at least 10cm away from the obstacle at all the tested speeds.

Speed $[\frac{m}{s}]$	**Safety margin** $[m]$	**Distance to obstacle** $[m]$
1	0	-0.01
1	0.3	0.07
1	0.4	0.142
1	0.5	0.213
2	0	-0.01
2	0.3	0.028
2	0.4	0.028
2	0.5	0.14
3	0	-0.013
3	0.3	0.027
3	0.4	0.027
3	0.5	0.211

Table C.1: The results of testing how well the system brakes with different values of the safety margin. The chosen safety margin was 0.5 meters, since at all speeds the vehicle stops at least 10cm away from the obstacles.

To test the model functioned as expected, the following tests were performed and succeeded:

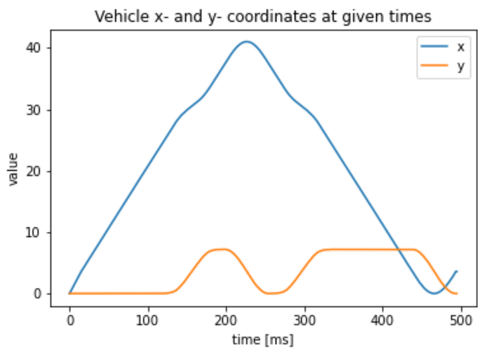
- Test the system in map1, with the safety of the supervisory controller turned on, where the system performs a safety stop before colliding with an obstacle. The sensor range is set to $[0.5; 2]$ meters, the vehicle speed is $2m/s$.
- Test the system in map1, with the safety of the supervisory controller turned off, where the system does not perform a safety stop and therefore collides with an obstacle. The sensor range is set to $[0.5; 2]$ meters, the vehicle speed is $2m/s$.
- Test the system in map1, with the safety of the supervisory controller turned on, where the system decreases the speed of the vehicle to decrease the braking distance to be within the maximum sensor range. The sensor range is set to $[0.5; 1]$ meters, the initial vehicle speed is $3m/s$. The initial braking distance was calculated to be 1.3 meters, and therefore must be reduced. The supervisory controller decreased the speed to $2.3m/s$ to decrease the braking distance to 1 meter.
- Test the system in map1, with the safety of the supervisory controller turned on, where the system decreases the speed of the vehicle to decrease the braking distance to be within the maximum sensor range. The sensor range is set to $[0.5; 1]$ meters, the vehicle speed is $2m/s$. The initial braking distance was calculated to be 0.87 meters, and therefore the supervisory controller did not change the speed of the vehicle.

C.3 General Characteristics

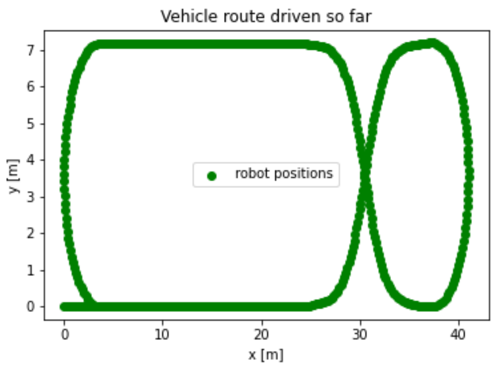
This section describes some general characteristics of the methods used when implementing the models.

C.3.1 Visualizing Results

When running a co-simulation it is common practice to plot the values of the variables exchanged between models throughout time. These plots contain the time represented in the x-axis, and the value of the variable represented in the y-axis. In many cases this is a sufficient visualization method for the developer to easily follow the progress of the co-simulation while it is running. In other cases, such as in this thesis, it would be of more value to the developer, if the visualization showed the current x and y positions of the vehicle.
Figure C.3 shows the two techniques for visualizing the vehicle x- and y-positions. The strength of visualization method *a*, is that other values with different units may be plotted at the same time. The strength of visualization method *b*, is that a human can easily understand what is going on with the vehicle, and if it is operating as it should. This especially aids the developer, when this plot is made live during the co-simulation, where obstacles detected by the sensor are also plotted. This allows the developer to observe how the vehicle operates when it approaches an obstacle, if the obstacle is detected by the vehicle, and if the vehicle stops in time to avoid a collision with the obstacle.



(a) Plot of the x- and y-position values of the vehicle so far during the co-simulation, where the x- and y-positions are displayed on the y-axis, and the current time is displayed on the x-axis.



(b) Plot of the route that the vehicle has driven so far during the co-simulation, where the x- and y-positions are each displayed on their own axis.

Figure C.3: Two different techniques for visualizing the x- and y-position of the vehicle during a co-simulation.

C.3.2 Connecting Vectors between FMUs

The FMUs implemented in this thesis adhere to the latest defined FMI standard, FMI 2.0 [25]. The standard does not yet contain a definition for exchanging vectors as a datatype, which was problematic for some of the FMU connections. Therefore a work-around was implemented, as shown in figure C.4, where a predefined number of connections were set up, including a parameter indicating the number of connections that were used.

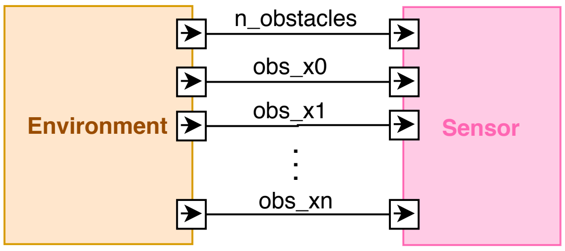


Figure C.4: A demonstration of how to connect vectorized data between FMUs by using a parameter defining the number of, in this case, obstacles that are set.

C.4 Environment Maps

The environment maps used in the test of this system are shown in figures C.5 and C.6, which are map 1 and 2 respectively.

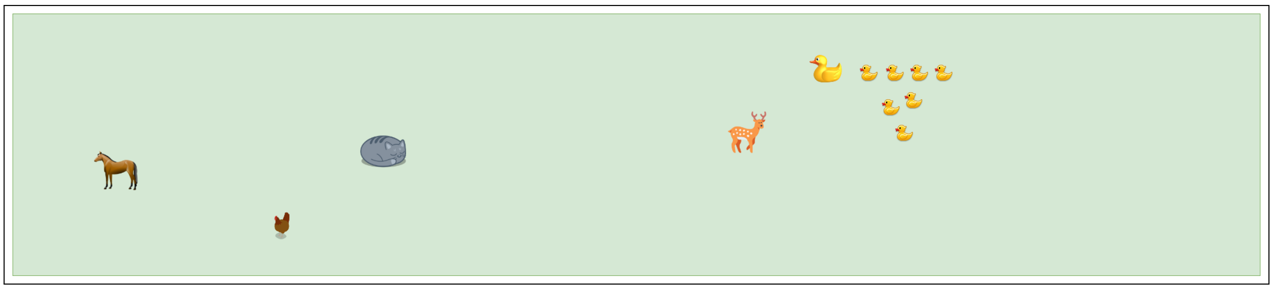


Figure C.5: Map 1 used as the environment for the co-simulation scenarios. Each object on the map is an represented as an obstacle in the environment.

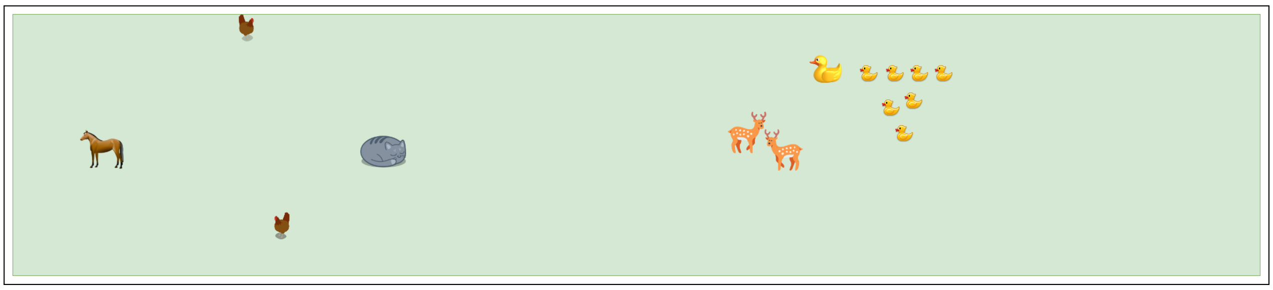


Figure C.6: Map 2 used as the environment for the co-simulation scenarios. Each object on the map is an represented as an obstacle in the environment.